Weekly report

Maire AIMAR-BEURTON and LE Van Linh

November 2016

Abstract

This document contains the summary about the morphometry and deep learning studied. Besides, it also contains the algorithms to apply for image processing such as segmentation or detection the dominant points.

Part I

Morphometry

Chapter 1

Segmentation

1.1 Canny algorithm

In 1986, **John F.Canny** had proposed a method to determine the edge in image. This is a technique to detect the useful structure of the object in digital image. Until now, the Canny algorithm[1] is used widely for the segmentation in computer vision. The process of Canny algorithm can be described in 4 steps as follows:

1. Smoothing the image to reduce the noises by using Gaussian filter
2. Finding the intensity and direction gradient of each pixel in image
3. Eliminating the weak edge by using the edge thinning technique.
4. Applying double threshold to determine the potential edges

1.1.1 Gaussian filter

To smooth the image, a Gaussian filter is applied to convolve with the image. This step will help to reduce the effects of the noises on the edge detector. Normally, the equation of a Gaussian kernel with size $(2k+1)$ x $(2k+1)$ is computed as:

$$H_{ij} = \frac{1}{2\pi\sigma^2} exp(-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}); 1 \leq i, j \leq (2k+1) \tag{1.1}$$

where k is the size of kernel, and it should be a odd number.
For example, a 3x3 Gaussian filter with $\sigma = 1$ as followed:

$$G = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{1.2}$$

The selection of the size of the Gaussian kernel is important, it will affect the performance of the detector. If the size of the kernel is large, the detector can be sensitive to noise; otherwise, if the kernel's size is small, the detector can be destroy many strong edge. In the practice, this step is combined into Sobel convolution with a 3x3 kernel, which used to finding the intensity and direction gradients at each pixels of image.

1.1.2 Sobel convolution

The points belong to the edge in an image can stay in any direction, so the Canny algorithm uses four filters to detect the edges (vertical, horizontal and two diagonal edges) in the image. And the Sobel operator is used to detect the edges. This operator returns a value for the first derivative in horizontal direction (G_x) and the vertical direction (G_y). From these values, the gradient and direction of edge at each pixel are determined:

$$G = \sqrt{{G_x}^2 + {G_y}^2} \tag{1.3}$$

$$\phi = atan2(G_y, G_x) \tag{1.4}$$

In this case, the kernel of Sobel convolution is 3x3, and it is also combined the Gaussian filter to smooth the image. The kernels are used to convolute the horizontal direction and vertical direction as follows:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{1.5}$$

The edge direction angle is rounded to one of four angles which were presented for four directions: vertical, horizontal, and two diagonals $0^o, 45^o, 90^o$ and 135^o.

1.1.3 Non-maximum suppression

Non-maximum suppression is applied to thin the edge in an image. Thus, this operation is used to suppress all the gradient values to 0 except the local maximal. At every pixel, it suppress the gradient value of the center pixels if its magnitude is smaller than the magnitude of one out of two neighbors in the gradient direction. In details:

- If the gradient direction angle is **0** degree, the point will be considered to be on the edge if the gradient magnitude is greater than the magnitude at pixels in the **east** and **west** directions.
- If the gradient direction angle is **45** degree, the point will be considered to be on the edge if the gradient magnitude is greater than the magnitude at pixels in the **north east** and **south west** directions.
- If the gradient direction angle is **90** degree, the point will be considered to be on the edge if the gradient magnitude is greater than the magnitude at pixels in the **north** and **south** directions.
- If the gradient direction angle is **135(-45)** degree, the point will be considered to be on the edge if the gradient magnitude is greater than the magnitude at pixels in the **north east** and **south west** directions.

1.1.4 Double threshold

After applying the non-maximum suppression, the edges pixels are presented. However, there are still some edge pixels effected by noise. Double threshold will filter out the edge pixels with the weak gradient value and preserve the edge with the hight gradient value.

- A pixel called strong pixel (hence, it belong to the edge), if the edge pixel's gradient value is higher than the high threshold value.

- A pixel will be suppressed, if the edge pixel's gradient value is smaller than the low threshold value.
- A pixel called weak pixel (can be belong to the edge or not), if the edge pixel's gradient value is larger than low threshold value and smaller than high threshold value. A weak pixel can be belong to the edge if it connected with a strong pixel in 8-connected; else, it will be suppressed.

Thus, the accuracy of algorithm is depended on two parameters: the kernel of Gaussian filter and thresholds value. As said before, if we choose incorrect the kernel size of Gaussian filter, we can not reduce the noise or we can remove the real edge. Besides, the values of double threshold is also important to filter out the edge pixels. In practice, 1:3 is the good ratio between lower threshold and upper threshold in Canny.

1.1.5 Summary

With applying double thresholding in the last stage, Canny had provied a strict condition to consider the weak edge as well as remove the pixels which were not belong to the edge. So far, Canny algorithm is good method to determined the edge in image, it is used by many application in image processing.

1.2 Suzuki algorithm

The Canny algorithm had detected the edge in the image. We can apply many difference methods to track the edge. **S. Suzuki** and **K. Abe**[2] had proprosed a method to get the border of object in image. This method is based on the topological structure analysis on binary image.

Following this method, it detects two kinds of border in image. The first is outer border, which is defined by a set of border points between an arbitrary 1-component and the 0-component which surrounds it directly; another type is hole border which refers to the set of border points between a hole and the 1-componet which surrounds the hole directly. In this case, the 1-component (or 0-component) is connected component of 1-pixels (or 0-pixels).

In our case, our purpose is getting the edges which were detectecd by Canny[1]. An edge is consider as an outer border or a hole border does not important. So, the Suzuki algorithm could make some changes to fit with our aim. The processes of algorithm is desribed as follows:

Let an input binary image is $F = \{f_{ij}\}$. Set initially $NBD = 1$ (denoted the sequence number of border.)

1. Select one of the following:
   (a) If $f_{ij} = 1$ and $f_{i,j-1} = 0$, increment NBD, $(i_2, j_2) \leftarrow (i, j-1)$ (pixel (i, j) is the starting point of an outer border).
   (b) If $f_{ij} \geq 1$ and $f_{i,j+1} = 0$, increment NBD, $(i_2, j_2) \leftarrow (i, j+1)$ (pixel (i, j) is the starting point of an hole border).
   (c) Otherwise, go to step (3)
2. From the starting point **(i,j)**, the process to trace the edge is done by substeps following:
   2.1 Starting from point (i_2, j_2), look around clockwise the pixels in the neighborhood (8-connected) of (i, j) and find the first non-zero pixel (i_1, j_1). If no non-zero pixel is found, assign -NBD to f_{ij} and go to step (3)

2.2 $(i_2, j_2) \leftarrow (i_1, j_1)$ and $(i_3, j_3) \leftarrow (i, j)$

2.3 Starting from the **next element of the pixel** (i_2, j_2) in the counterclock-wise order, check the pixels neighborhood of current pixel (i_3, j_3) to find the first non-zero pixel (i_4, j_4).

2.4 Chang the value f_{i_3,j_3} of the pixel (i_3, j_3) as follows:

(a) If the pixels $(i_3, j_3 + 1)$ is a 0-pixel examined in the substep (2.3) then $f_{i_3,j_3} \leftarrow -NBD$. Else, $f_{i_3,j_3} \leftarrow NBD$ unless (i_3, j_3) is on an already border.

(b) If the pixels (i_3, j_3+1) is not a 0-pixel examined in the substep (2.3) and $f_{i_3,j_3} = 1$ then $f_{i_3,j_3} \leftarrow NBD$

(c) Otherwise, do not change f_{i_3,j_3}.

2.5 If $(i_4, j_4) = (i, j)$ and $(i_3, j_3) = (i_1, j_1)$ (coming back to the starting point), then go to step (3); otherwise, $(i_2, j_2) \leftarrow (i_3, j3), (i_3, j_3) \leftarrow (i_4, j_4)$ and go back to step (2.3)

3. Resume the scan from the pixel $(i, j + 1)$. The algorithm is stop when the scan reaches the lower right corner of the image.

The obtained results from Suzuki algorithm are list of the edges of the object in the image. Each edge is list of connected points. This result is very important for the next steps of automatic detection landmarks.

1.3 Approximated lines

The list of points from Suzuki algorithm can used to present the object. But in some case to consider the feature of the object, the information from the list of points is not perfect. Instead of, we use another kind of the geometric object to represent the object. This is the line. In this section, we will describe the duration to get the approximated lines of object from the list of edge (each edge is represented by the list of points).

The method is used to fragment the edge into a list of approximated line is a recursive algorithm[3], which is a new improved version with the method proposed by Lowe[4] except the stop condition is considered as a parameter λ. The steps of algorithm are followed:

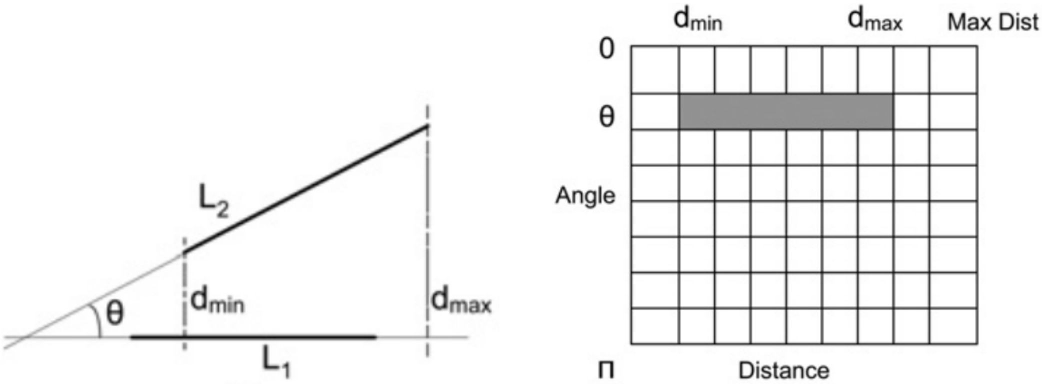
1. Create a straight line l between two endpoints of the edge
2. Calculate the perpendicular distance from each point on edge to line l, and identifying the maximum point p_m.
3. If the perpendicular distance from maximum point p_m is greater than the stop condition($d(p_m, l) > \lambda$), then the edge is split at this point and both parts are reprocessed. Otherwise, if we do not have any p_i that the perpendicular distance from they to l greater than λ then the edge can be represented by l.

1.4 Pairwise Geometric Histogram

In image processing, we have many techniques to describe the features of the image. With expect represent the image's features into the variant information for compare and representation, pairwise geometric histogram (PGH) method is chosen. PGH is constructed based on the geometry relative of the image's features. With an image is presented by the list of lines, the angle and perpendicular distance between two lines are importance characteristic to consider for re-constructor the image. Moreover, PGH is also fitwell when we apply some variants on the image such as translation or rotation because the angle and perpendicular distance between two lines are invariant. The process to calculate the PGH of an image which is represented by a list of lines as follows:

- Choose arbitrary line as reference lines l_f
- For each other lines l in image, calculate the angle between l and l_f, and perpendicular distance from two endpoints of l to l_f.
- The process will stop when all line in the image are considered as reference line.

An importance note, during the process calculate the PGH, we need to keep the information of PGH to reconstruct the image or compare with other image.



(a) The geometric relationship between two lines

(b) The information of PGH is kept into a matrix

Figure 1.1: Example about geometric features and the pairwise geometric histogram

To detect the similarity between two images, we can use the pairwise geometric histogram matching via the similar distance of their probability distribution on histogram. The Bhattacharyya metric is common distance to compare two models. The form of Bhattacharyya is:

$$d_{Bhattacharyya}(H_iH_j) = \sum_{\theta}^{\pi}\sum_{d}^{d_{max}}\sqrt{H_i(\theta,d)H_j(\theta,d)} \tag{1.6}$$

The significance of parameters in the formula 1.6, as follows:

- θ: angle value, range of θ in angle axis from 0 to π.
- d: the perpendicular distance, range of d in perpendicular distance from 0 to the maximum distance of arbitrary lines of shape.
- $H_i(\theta, d)$ is an entry at row θ and column d in PGH of image i
- $H_j(\theta, d)$ is an entry at row θ and column d in PGH of image j

1.5 Probabilistic Hough Transform

Probabilistic Hough Transform (PHT) is used to detect the presence of a model image in a scene image based on the group of features. The hypothesised location of the model image in the scene image is indicated based on the conditional probability that any pair scene lines agreement about a position in model image. Applying PHT can be separated into two steps: firstly, recording the information of model image and try to find the presence of the model image in scene image (called training process); secondly, predicting the pose of model image in the scene image (called estimating process).

During training process, choose an arbitrary point in the model image, called reference point. For each pair of lines in model image, the perpendicular distance and angle from each line to reference point is recording (angle is calculated as angle between line and a horizontal line begin from reference point). The presence of model image in scene image is detected by PHT with *"vote"* procedure. Finally, we choose the similar pair lines between model image and scene image. The chosen pair is obtained from best *vote* when we consider each pair of line in scene image with each pair of lines in model image.

In estimating process, the reference point in model image is estimated in scene image by extending the perpendicular lines of the pair of scene lines at the appropriate position. There, we can estimate the pose of the model in the scene image.

1.6 Template matching

Template matching is a technique for finding areas of an image that match to a template image (template) by sliding the template over each pixel on the image (commonly cross-correlation). At each position, the sum of products between two images is calculated. The position is considered similar if the sum value at this position is maximal. The equation of cross-correlation is as follows:

$$R_{ccorr}(x,y) = \sum_{x',y'} [T(x'.y').I(x+x',y+y')] \tag{1.7}$$

Where:

- T is template which use to slide and find the exist in other image.
- I is image which we expect to find the template image
- (x', y') are coordinates in template where we get the value to compute.
- $(x+x', y+y')$ are coordinates in image where we get the value to compute when template T sliding.

However, if we use the original image to compute and find the similarity, the brightness of the template and the image might change the conditions and the result. So, we can normalize the image before applying the cross-correlation to reduce the effect of lighting difference between them. The normalization coefficient is:

$$Z(x,y) = \sqrt{\sum_{x',y'} T(x'.y')^2 . \sum_{x',y'} I(x+x',y+y')^2} \tag{1.8}$$

The value of this method when we normalized computation as below:

$$R_{ccorr\_norm}(x,y) = \frac{R_{ccorr}(x,y)}{Z(x,y)} = \frac{\sum_{x',y'} [T(x'.y').I(x+x',y+y')]}{\sqrt{\sum_{x',y'} T(x'.y')^2 . \sum_{x',y'} I(x+x',y+y')^2}} \tag{1.9}$$

Chapter 2

Dominant points

In shape analysis, extracting features from the curves is an important step because in another way, we can re-construct the shape from the features. The term dominant points, also called as siginficant points, points of interest, corner points or landmarks is assigned to the points which have the high effect on boundary of object; their dectection is a very important aspect in contours methods because these concentrate the information of a curve on the shape.

Dominant points can be used to produce a presentation of a shape contour for futher processing. The representation ... In the content of this chapter, we will discuss about the methods to determine the dominant in digital image.

There are many approaches developed for detecting dominant points and the methods can be classified into three groups follows:

- Dectermine the dominant points using some significat measure other than curvature
- Evaluate the curvature by transforming the contour to the Gaussian scale space.
- Search for dominant points by estimating directly the curvature in the original image space.

2.1 Method 1

2.2 Method 2

2.3 Method 3

Chapter 3

Software

3.1 The software architecture

The architecture of program is followed 3-tier model. There-tier architecture is an architecture that each tier is designed, developed and maintained as independent. The advantage of this architecture is intended to allow any upgraded or replaced independent between the tiers. When user want to change the requirements or technology of a tier, it will non-affect to other tiers.

The architecture of three-tiers includes:

- **Data tier**: includes the classes which were designed for the data structure of program. It also provides the persistence mechanism to access the data.
- **Logic tier**: controls the functionality of application by performing detailed processing.
- **Presentation tier**: displays information related to user. It is a layer which received the require from user to program or return the result from program to user.

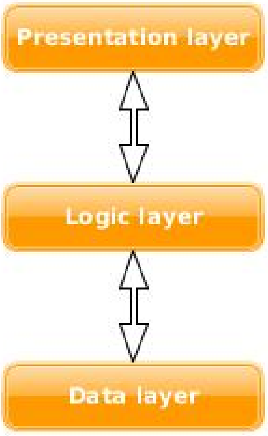


Figure 3.1: Three-tiers model

3.2 The modules

The MAELab software mainly includes four modules: **segmentation**, **histograms**, **pht** and **correlation**. Besides, the software also includes the other modules to support for the main

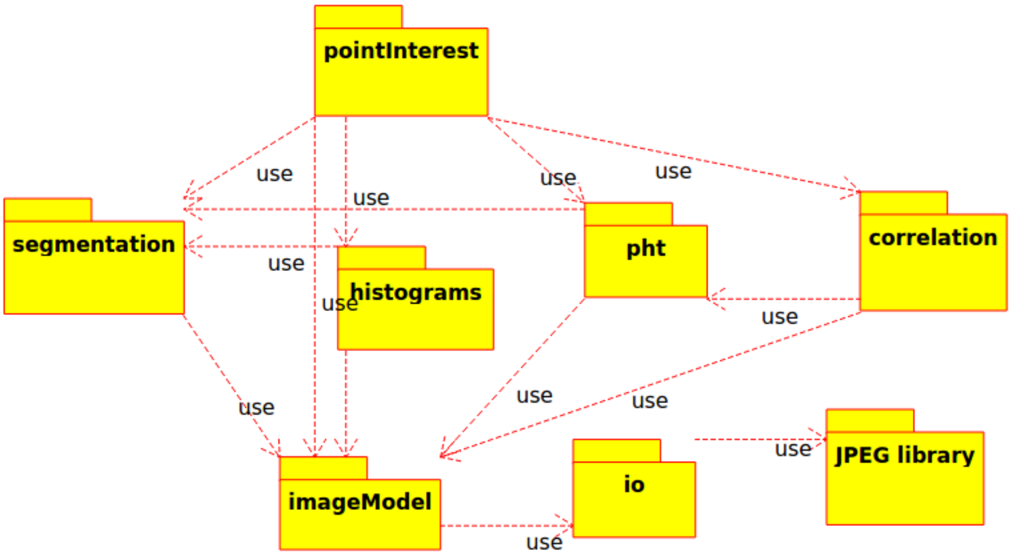


Figure 3.2: Three-tiers model

modules. The relation between the modules in the software is shown in figure 3.2. The functions of each modules is describing as followed:

- **io** module: Implement the functions to read and write file. It includes the **JPEG library** that used to decode and encode the JPEG image.
- **imageModel** module: Represent the data structure of the image.
- **segmentation** module: Implement the segmentation methods on image.
- **histograms** module: Contains the methods to compute the geometric histogram of the image.
- **pht** module: Describe the probabilistic hough transform duration.
- **correlation** module: Includes the template matching methods.
- **pointInterest** module: Combine the result of the modules such as segementation, histograms,... to provide the adapter to other module or other software.

3.3 The modules

3.4 The classes architecture

Part II

Deep learning

Chapter 4

Machine Learning

Machine learning is a norm refer to teach the computer the abilities which are only done by the humans. A machine learning algorithm is an algorithm that is able to learn from data. Most of machine learning algorithms can be divided into two categories: supervised learning and unsupervised learning algorithms.
A machine learning algorithm is built based on the tasked for a machine learning system. We have many kinds of task can be solved with machine learning. Some of common machine learning tasks include the following:

- *Classification*: In this type of task, the computer is asked to indicate a category in k category which the input belongs to. To solve this task, the learning algorithm uses a function $y = f(x)$, the model assigns the input described by vector x to a category identified by score y.
- *Classification without input*: A challenge of classification is missing the input vectors. In this case, to solve the classification task, the learning algorithm only has to define a single function mapping from a vector input to a category output. When some of inputs are missing, instead of providing a single classification function, the learning algorithm must learn a set of functions. Each function corresponds to classifying x with different subset of its inputs missing.
- *Regression*: the computer program is asked to predict a numerical value given some input.
- *Transcription*: machine learning system is asked to observe a relatively unstructured representation of some kind of data and transcribe it into discrete, textual form.
- *Translation*: The input already contains the sequence of symbols in some languages, the computer program must convert it into the sequence of symbols of other languages.
- *Structure output*: involve any task where the output is a vector with important relationships between the different elements.
- *Anomaly detection*
- *Synthesis and sampling*: The program is asked to generate the new example that are similar with the training data.
- *Imputation of missing value*: The algorithm mus provide a prediction of the values of the missing entries in a new example.
- *Denoising*
- *Density estimation or probability mass function estimation*

4.1 Supervised learning algorithms

4.2 Unsupervised learning algorithms

4.3 Stochastic Gradient Descent

Chapter 5

Classification

Classification is a most of important task in machine learning. In classification, a function is constructed to determine the category of the input. Generally, the model of classification as following:

The process of classification includes two steps:

1. **Training**: Use the **training set** to learn what every object of a class looks like. This duration is called training a classifier or learning a model. The training set is a set with the objects which have labeled with specific category.

2. **Evaluation**: To evaluate the quality of the classifier. We use a new set **(test set)** of the objects and try to ask the classifier predict the category of the object in the test set.

In the content of this chapter, we will discuss about the classification techniques, especially, linear classification which technique has used more in neural network and deep learning.

5.1 Nearest Neighbour Classifier

The first approach to Classifier, we will develop Nearest Neighbour Classifier. This classifier do not have any relation with deep learning or convolutional networks, but it will help us to have an overview about classification problem.

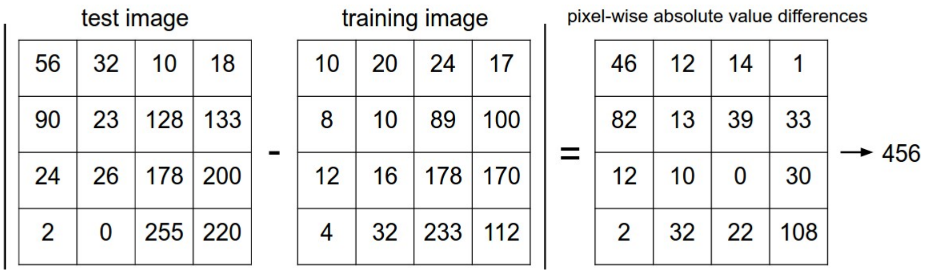
The idea of Nearest Neightbour Classifier is comparing each image in test data set with all image in training data set and predict the label of closet training image. And one of simplest methods to compare two images is comparing each pixels of two images and sum of all the differences. Assum that we have two vector I_1, I_2 presented for two images, the equation to compare two images is following (called **L1 distance**):

$$d_1(I_1, I, 2) = \sum_p |{I_1}^p - {I_2}^p| \tag{5.1}$$

Actually, we have many ways to compute the distances between two image. Instead of using L1 distance, we can use **L2 distance**, which has indicated by square root of euclidean distance between two vectors. The form of L2 distance as:

$$d_2(I_1, I, 2) = \sqrt{\sum_p ({I_1}^p - {I_2}^p)^2} \tag{5.2}$$

For example, this is a way to compute distace between two images (fig. 5.1):


test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

Figure 5.1: An example used **L1 distance** to compare two images

5.2 K-Nearest Neighbour Classifier

In the case of Nearest Neighbour Classifier, we just determine only one closest image in the training data with the test image when we wish to make a prediction. It means that we need some images in training data set that closest with the test image. In this case, we can use the **k-Nearest Neighbour Classifier**. The idea of this method is finding top **k** closest images instead of singel closest image (hence, when k = 1, we recover the Nearest Neighbour Classifier).

In practice, what is the best value of k that we should to use? Besides that, we have many choices for compute the distance between two images different with L1 distance, L2 distance. The method called **hyperparameters** is vary for this work. This method comes in the design of many Machine Learning algorithsm, and it is used to choose the setting values. We should try out many different values and see what works best. This is the idea, but we must be done vary carefully. Another noticed that, we do not try to evaluate on test data set with each k. After having the k, we evaluate on the test set only a single time, at the end of procedure.

The idea is spitting the training data set in two subsets: the first subset is used to training, the other subset is used to validate (called **validation set**). The validation set is used as the test set to indicate the value of k. At the end of procedure, we could determine values of k work best. We would then use this value and evaluate once on the actual test set.

In summary, split the training set into training set and validation set. use validation ton tune all hyperparameters. At the end run a single time on the test set and evaluate the result.

5.3 Linear Classification

The (k-)Nearest Neighbour Classifier had introduced about the problem of Image classification, which is predicting the label to an image from a fixed set of labels. But with the these methods, we must spend more time with large datasets an the cost for classifying is expensive. Another classification methods is known as **linear classification** which is the core of neural networks. The linear classification has two main components:

- **Score function**: which used to map the raw data to score of a category.
- **Loss function**: that quantifies the agreement between predict score and the truth category of the data.

The simplest function of a linear mapping is:

$$y = f(x_i, W, b) = Wx_i + b \tag{5.3}$$

Where:

- x_i is the raw data, *example: an image.*

- W: a matrix parameter, called **weight** matrix
- b: vector, called **bias** vector
- y: score when consider the data x_i belongs to a category.

In equation above, the input image x_i is fixed but we can control the setting of parameters **W and b**. Our goal is setting the parameters that the computing score match with the truth labels of image.

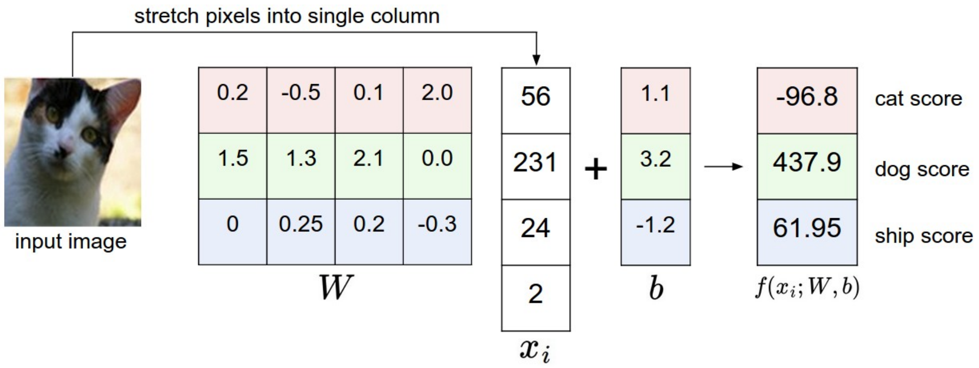


Figure 5.2: An example of mapping an image to a class scores

In training process, it is a little cumbersome to keep two sets of parameters (W,b) separately. A commonly trick is used to combine two sets of parameters into a single matrix that holds both of them by extending a vector x_i with one additional dimension and keep the constant defaut 1. Now, the new score function will be:

$$y = f(x_i, W, b) = Wx_i \tag{5.4}$$

Visualation of new score function:

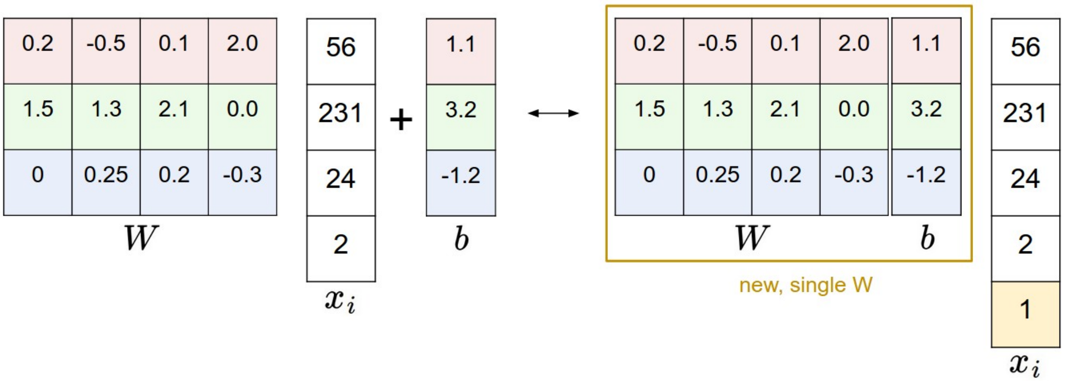


Figure 5.3: An example of bias trick

As described, we defined a score functions from a pixels value of an image to class scores with set of parameters **W**. Moreover, we need to control over parameters W such that the class scores are consistent with the ground truth labels in the training data. But not all cases are perfect, the class scores just near with the score of truth labels. So, we are going to measure the wrong with a **loss function**. Intuitively, the loss will be high if we are doing a poor classifier, ant it will be low if we are doing well. Multicalss Support Vector Machine (SVM) and Softmax function are two commonly methods for this purpose.

5.3.1 Multiclass Support Vector Machine loss

A commonly way to define the loss function called the **Multiclass Support Vector Machine** (SVM) loss. SVM loss is set up a margin Δ for the incorrect class scores. It means that SVM loss function wants the score of the correct class to be greater than the incorrect class (predict score) by at leat Δ. If this is not the case, we will accumulate the loss.

The SVM loss for the i-th is formalized as follows:

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + \Delta) \tag{5.5}$$

Where:

- s_j is the score of x_i for j-th class
- s_{y_i} is the score of correct class
- Δ is margin
- $max(0, -)$ is thresholding to zero, called hinge loss.

For example, we have three predict scores of an image x_i like $s = [14, -9, 11]$, and the first class is true class of x_i. Assume that Δ is 10. The SVM loss of this case is following:

$$L_i = max(0, -9 - 14 + 10) + max(0, 11 - 14 + 10) = 0 + 7 = 7$$

5.3.2 Softmax classifer

The other popular choice to define the loss function is the **Softmax classifier**. Unlike SVM which treats the output of score function for each class, the Softmax classifier give a sightly more intuitive output and use the probabilistic description. Instead using theshold zero function as SVM, Softmax is using a **cross-entropy loss** for *hingle loss*, which has the form.

$$L_i = -log(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}) \tag{5.6}$$

Where: f_j is the j-th element of vector of class scores f.

In formula 5.6, the function $f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$ is called the softmax function. This formula turns the predict scores into probabilistic values (Noticed that sum of all $f_j(z)$ is 1).

The cross-entropy between a correct distribution **p** and an estimated distribution **q** is defined as:

$$H(p, q) = -\sum_x p(x)log(q(x)) \tag{5.7}$$

The Softmax classifier is minimizing the cross-entropy between the estimated socre and true score. At the end, the loss of training process is the average of cross-entropy.

$$L = \frac{1}{N}\sum_i (H(p, q)) \tag{5.8}$$

The image 5.4 describes an example for a comparison between SVM and Softmax:

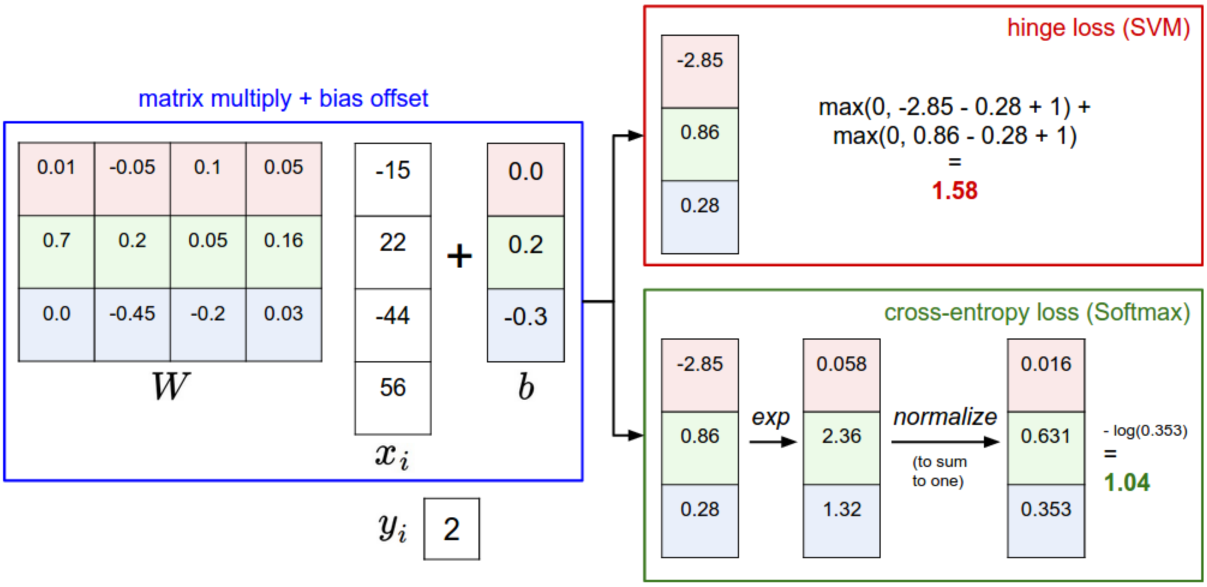


Figure 5.4: An example about SVM and Softmax classifiers

Both SVM and Softwax compute the same score of vector f. The difference is the way to present the score f: SVM uses the margin and Softmax uses probabilistic. In practice, the SVM and Softmax are usually used and compared in the machine learning systems.

5.4 How to determine the value of W and b?

As we have seen in previous section, the loss function is used to quantify the quality of any set of weights **W**. So, the choosing (or optimizing) the weights W is really important to obtain a good prediction. The goal of this process is to find W that minimize the loss function. In this section, we will describe some strategy to optimize the W.

5.4.1 Random search and random local search

The core idea is finding the best set of weights W. We begin with the random weights W, try out many different random weights and keep the W what works best.

Another way to try with random search is to try to extend the random direction . We also start with a random **W**, generate a random noise $\delta\mathbf{W}$ to it and if the loss at the concern **W** + $\delta\mathbf{W}$ is lower, we will perform an update. Following that method, the accuracy of classifier is getting on **21.4%** (by experiment).

5.4.2 Following the Grandient

Another method is usually used to optimize the W that following the best direction which we should change our weight (steepest increase or descend). This direction is related to the gradient of the loss function.

Gradient Descent is the procedure of repeatedly evaluating the gradient and then performing a parameter updated. In an application with the training data set is large, we must wasteful to compute the full loss function to perform the parameter. A very common way is addressing this challenge with a batches (a subset) of the training data. The extreme case of this is a setting where the mini-batch contains only a single example. This process is called **Stochastic Gradient Descent (SGD)**

5.5 Backpropagation

With a classifier, we can use SVM or Softmax to compute the loss of classifier, and the input are both the training data and the parameter weights **W** and biases **b**. Clearly, the value of training data is fixed, so we can control the value of loss function via controling the weight parameters.

In this section, we will discuss about a method to compute the gradient of a function **f(x)** at **x** (i.e $\nabla f(x)$), called backprobagation. The core of backprobagation is computing gradients of function through recursive application of **chain rule**.

Let consider a simple function: $f(x, y) = xy$. The partial derivative for this function as followed:

$$f(x, y) = xy \rightarrow \frac{\partial f}{\partial x} = y, \frac{\partial f}{\partial y} = x \tag{5.9}$$

The purpose of derivative is indicated the rate of change of the function with respect to that variable surrounding a small region near a particular point. Example, if **x = 4, y = -3** then $f(x, y) = -12$. The derivate on x is $\frac{\partial f}{\partial x} = -3$, it means if we increase the value of x by a tiny amount, the value of this funciton will be to decrease it. Otherwise, the derivate on y is $\frac{\partial f}{\partial y} = 4$, if we increase the value of y, the function also increase the output.

The derivaties for the addition operation:

$$f(x, y) = x + y \rightarrow \frac{\partial f}{\partial x} = 1, \frac{\partial f}{\partial y} = 1 \tag{5.10}$$

And for *max* operation, the gradient is 1 on the input that was larger and 0 on the orther input:

$$f(x, y) = max(x, y) \rightarrow \frac{\partial f}{\partial x} = 1(x \geq y), \frac{\partial f}{\partial y} = 1(y \geq x) \tag{5.11}$$

The image 5.5 show an example about applying derivative to calculate the backward pass of the fucntion $f(x, y, x) = (x + y)z$:

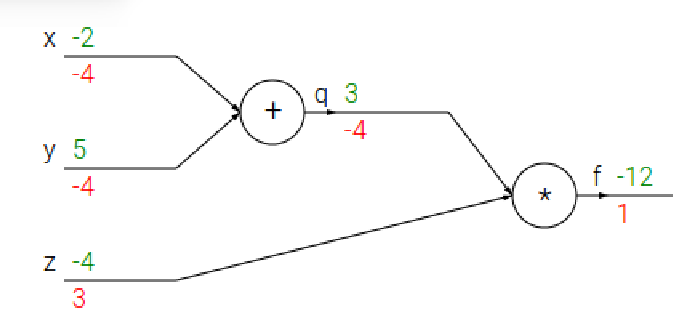


Figure 5.5: A backpropagation example

The backpropagation is good local process. Each gate of the circuit gets some inputs and compute two things (1) its output value and (2) the local gradient of its inputs with respect to its output value. Morever, the process at each gate can do independent. However, one the forward pass is over, during backpropagation the gate will eventually learn about the gradient of its output value on the final output of the entire circuit. Chain rule says that the gate should take that gradient and multiply it into every gradient it normally computes for all of its inputs.

Another kind of function at gate of circuit which we use to compute the gradient of function is *sigmoid activation* function. The form of sigmod function as follows:

$$\sigma(x) = \frac{1}{1+e^{-x}} \rightarrow \frac{d\sigma(x)}{dx} = (1-\sigma(x))\sigma(x) \tag{5.12}$$

Chapter 6

Deep Network

6.1 Neural network

6.1.1 Neural

The basic components of the brain is a neuron. For the ordinary man, we have billion neurons in the human nervous system, and they are connected by the billion of synapses. Each neuron receives input signals from its dendrites and procedures output signals along its axon.

In the computational model of a neuron, the signals travel along the axons interact multiplicatively with the dendrites of the other neuron based on the synaptic strength at the synapse. The synaptic strength are learnable and control the strength at influence or inhibitory of one neuron on another. In basic mode, the input signals are summed and compared with a threshold value. If the sum is greater than threshold value, the neuron can fire, sending a spike along its axon. Actually, we have many firing rate (called activation function) at a neuron, and the common choice of activation function is the **sigmoid funciont** σ, because it take a real-valued input and squashes it to range between 0 and 1. The image () show the model of a neuron: Some activation functions which we can use:

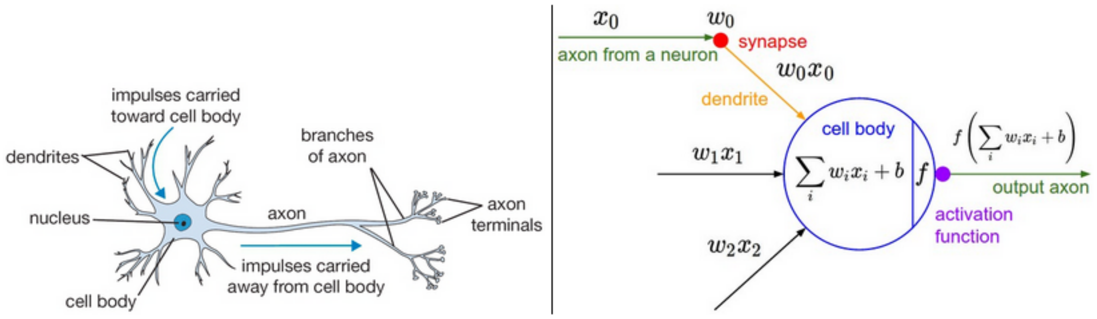


Figure 6.1: A drawing of a biological neuron and its mathematical model

- **Sigmoid function**:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{6.1}$$

- **Tanh**

$$tanh(x) = 2\sigma(2x) - 1 \tag{6.2}$$

- **ReLU**

$$f(x) = max(0, x) \tag{6.3}$$

- **Maxout**:

$$f(w^T x + b) = max(w_1{}^T x + b_1, w_2{}^T x + b_2) \tag{6.4}$$

6.2 The architecture of neural networks

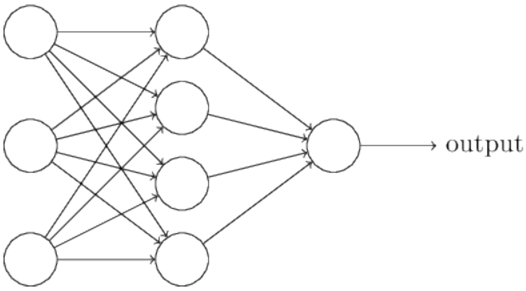


Figure 6.2: A model of neural networks

The image 6.2 show a simple model of neural networks. The leftmost layer in this network is called the input layer, the rightmost layer is called the output layer. The neurons within the input layer are called input neurons, the neurons from output layer are called output neurons. The middle layer is called a hidden layer. The network in example 6.2 has just a single hidden layer, but many networks have multiple hidden layers. When design the network, the input and the output are often straightforward. It means that the neural networks is designed where the output form one layer is used as the input to the next layer, there are no loops in the network, it always feed forward, never feed back (called feedforward networks).

So, the neural network includes many layers are designed as an directred acyclic graph from the intput to the output layer. The output of previous layer is used as the input of the next layer. At each layer excepts the output layer, the output is indicated by a activation function (i.e loss, tanh,...). The size of a neural network can be to compute as the number of neurons, or the number of parameters.

6.3 Deep network

Chapter 7

Convolutional Neural Network

Convolutional Neural Networks (CNNs) are similar with the original of Neural Networks. Neural Networks receive an input and pass it through a series of hidden layer. Each hidden layer is made from a set of neurons, where each neuron is full connected with all neurons of previous layer. Actually, the neural networks do not scale well to full images...

7.1 Architecture

A CNN is made from the layers. The common layers in CNN are convolutional, nonlinear, pooling and full connected layers. CNN takes image as an input, pass it through the series of layers and get an ouput. Each layer has a difference function to transform the input to another layer.

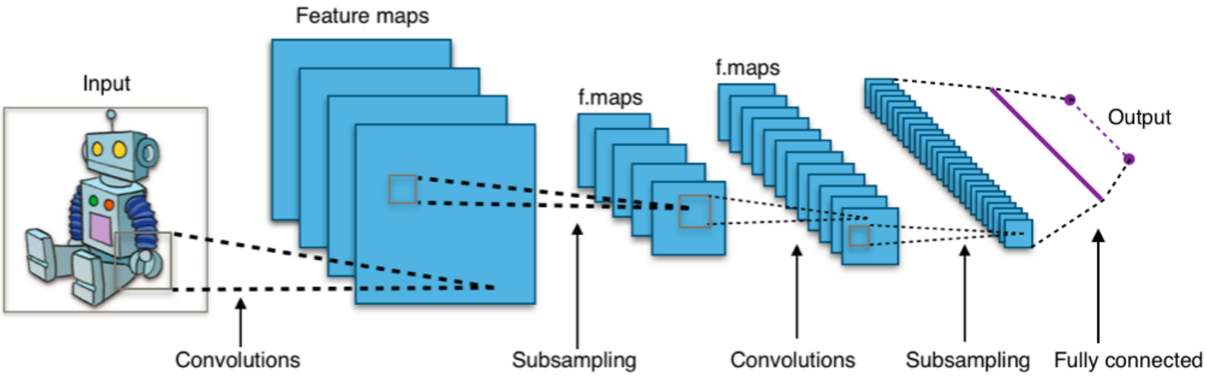


Figure 7.1: An architecture of convolutional neural network

7.1.1 Convolutional layer

7.1.2 Pooling layer

7.1.3 Full connected layer

7.2 Caffe framework

Part III

Conclusion

Chapter 8

Discussion

Chapter 9

Conclusion

Bibliography

[1] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[2] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[3] Neil A Thacker, PA Riocreux, and RB Yates. Assessing the completeness properties of pairwise geometric histograms. *Image and Vision Computing*, 13(5):423–429, 1995.

[4] David G Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial intelligence*, 31(3):355–395, 1987.