Pattern Recognition

Dimitris Gangas, Theodoros Lymperopoulos

December 17, 2019

Contents

1 Pattern recognition

1.1 Introduction

Pattern recognition is concerned with the automatic identification of regularities in data through the use of computer algorithms. With the use of these regularities, pattern recognition classifies the data into different categories. The main goal in this current project is to study a few mathematical tools and algorithms used in an application; the recognition of shapes into an image, commonly known as image recognition. The same techniques can be also applied in other problems.

The two fundamental problems in a pattern recognition system are feature extraction (shape measurement) and classification. The problem of extracting a vector of shape measurements from a digital image can be further decomposed into three subproblems:

- **The first** is the image segmentation problem, i.e., the separation of objects of interest from their background.
- **The second** subproblem is that of finding the objects in the segmented image.
- **The final** subproblem is extracting the shape information from the detected objects.

The main interest of this study is the third subproblem of the shape measurement that seems as the most challenging aspect of the shape recognition problem. In this area, there are many tools available depending on the properties of the objects that are to be classified.Two of them will be described: **The Hough transform** and the **polygonal approximation**. Some new techniques that are applied at this problem are based on neural networks an deep geometric learning. That will be the final section.

Any additional information to make this study more complete, will be also included.

1.2 Shape measurement

First of all, it is essential to separate objects of interest from their background. Clustering is one of the most powerful approaches to this problem.

In this approach each pixel in the $N \times N$ image is treated as a complicated object by associating it with a local neighborhood. For example,someone may define the 5×5 neighborhood of pixel p_{ij}, denoted by $N_5[p_{ij}]$, as $\{p_{mn} | i-2 \leq m \leq i+2, j-2 \leq n \leq j+2\}$. The following step is the measurement of k properties of p_{ij} by making k measurements in $N_5[p_{ij}]$. Such measurements may include various moments of the intensity values (grey levels) found in $N_5[p_{ij}]$, etc. Thus each pixel is mapped into a point in k-dimensional pixel-space. Performing a cluster analysis of all the resulting $N \times N$ points in pixel-space yields the desired partitioning of the pixels into categories.

1.3 Cluster Analysis

A fundamental problem in pattern recognition of images is the segmentation problem: distinguishing the figure from the background. Clustering as discussed above is one of the most powerful approaches to image segmentation, applicable even to complicated images such as those of outdoor scenes.

Definition of Cluster analysis problem: Partitioning a collection of n points in some fixed-dimensional space into m < n groups that are "natural" in some sense. Here m is usually much smaller than n.

Classical cluster analysis requires partitioning points into natural clumps. "Natural" may mean that the clustering agrees with human perception, or it may simply optimize some natural mathematical measure of similarity or distance so that points that belong to one cluster are similar to each other and points far away from each other are assigned to different clusters.

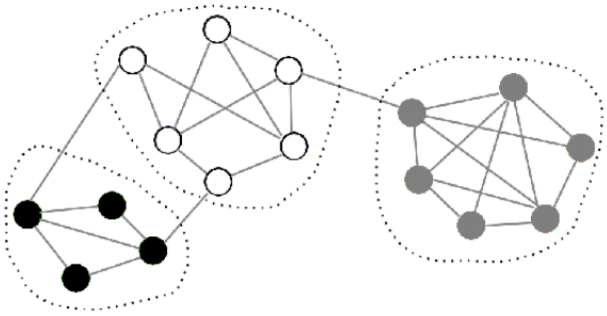
One common application is the determination of the number of classes, and the description of them in a pattern recognition problem where the classes are not known a priori (e.g. disease classification in particular or taxonomy in general). In this case **m** is not known beforehand and the cluster analysis reveals it.

Some of the most famous clustering methods that solve the segmentation problem of an image are presented below.

- **Graph-Theoritic Clustering**

  The algorithm is based on a simple idea:

  1. Compute some proximity graph (such as the minimum spanning tree) of the original points.
  2. Then delete (in parallel) any edge in the graph that is much longer (according to some criterion) than its neighbors.



The resulting forest is the clustering (each tree in this forest is a cluster).

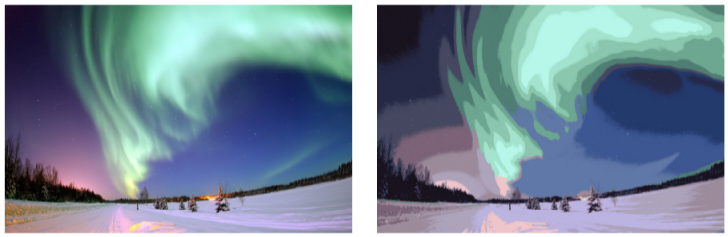
- **Hierarchical Clustering**

  There is no special number m that needs to be discovered; rather that, the main goal is the production of a dendrogram (tree) that grows all the way from one cluster to n clusters. This method contributes to an better understanding of how a partitioning is obtained for any number of clusters between one and n. Such methods are referred as hierarchical methods. They fall into two groups: agglomerative (bottom-up, merging) and divisive (top-down, splitting).

Traditional Hierarchical Clustering | Traditional Dendrogram

- **K-means Clustering**

  The k-means algorithm searches for k cluster centroids in R^d with the property that the mean squared (Euclidean) distance between each of the n points and its nearest centroid ("mean") is minimized.
  A typical heuristic starts with an initial partition, computes centers, assigns data points to their nearest center, recomputes the centroids, and iterates until convergence is achieved according to some criterion.



- **Left Image:** Source image.
- **Right Image:** Image after running k-means with k = 16. Note that a common technique to improve performance for large images is to downsample the image, compute the clusters, and then reassign the values to the larger image if necessary.

1.4 Hough transform

1.4.1 Overview

The Hough transform was originally proposed (and patented) as an algorithm whose objective is to detect straight lines in digital images. The method may be used to detect any parametrizable pattern, and has been generalized to locate arbitrary shapes in images. The basic idea is to let each above-threshold pixel in the image vote for the points in the parameter space that could generate it. The votes are accumulated in a quantized version of the parameter space, with high vote tallies indicating detection.

1.4.2 An example of a Hough transform

It is widely known that a line in the image space can be expressed by using two variables. For example:

1. In the Cartesian coordinate system: Parameters: (a, b)
2. In the polar coordinate system: Parameters: (r, θ)

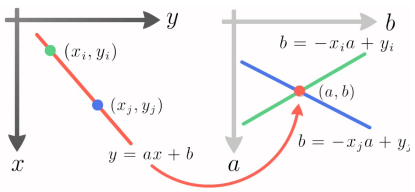


Figure 1: Dual: b= -ax+y

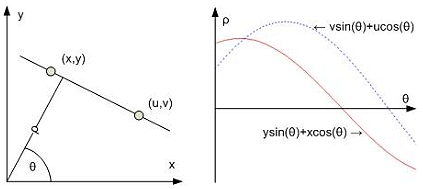


Figure 2: polar representation $r = x\cos(\theta) + y\sin(\theta)$

For Hough Transforms, the line equations are expressed in polar coordinates. Hence, a line equation can be written as:

$$y = (\frac{-\cos\theta}{\sin})x + (\frac{r}{\sin\theta})$$

By rearranging the terms the following equation can be obtained.

$$r = x\cos(\theta) + y\sin(\theta)$$

1. In general for each point (x_0, y_0), the family of lines that goes through that point is defined as:

   $$r_\theta = x_0\cos(\theta) + y_0\sin(\theta)$$

   Meaning that each pair (r_θ, θ) represents each line that passes by (x_0, y_0).

2. For a given point (x_0, y_0), the plot of the family of lines that goes through it is a sinusoidal curve. For instance, for $x_0 = 8, y_0 = 6$ we get the following plot (in a plane $\theta - r$):

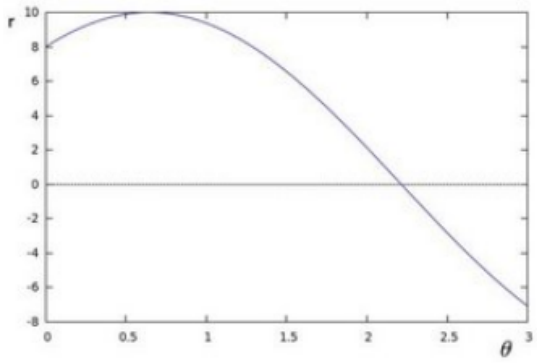
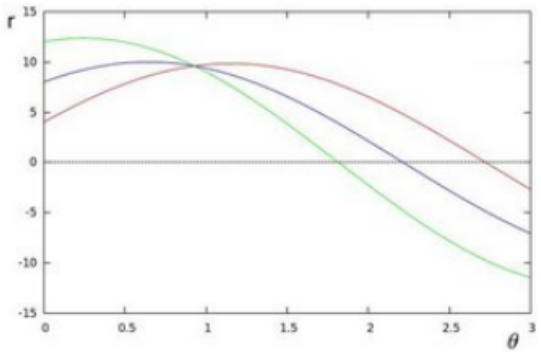


Figure 3: $r = 8 * \cos(\theta) + 6 * \sin(\theta)$

   We consider only points such that $r > 0$ and $0 < \theta < 2\pi$

3. The same process can be repeated for all the points in an image. If the curves of two different points intersect at the plane $\theta - r$, then the two points belong to the same line (in the cartesian coordinate system). For instance, following the example above and by adding the curves for two move points $x_1 = 4, y_1 = 9$ and $x_2 = 12, y_2 = 3$, the following plot is obtained.

   The three curves intersect at one point $(0.925, 9.6)$, which correspond to the parameters (θ, r) of the line in which (x_0, y_0), (x_1, y_1) and (x_2, y_2) lie.



4. The above result can be generalized for any given line, which means that any line can be detected by finding the number of intersections between curves. If a given number of curves intersect at one point, then the same number of points lie on the corresponding line. In general, we can define a threshold for the minimum number of intersections required to detect a line.

5. This is the main function of the Hough Line Transform. For every pair of points of the image it keeps track of the intersection between the corresponding curves. If the number of intersections is above some predetermined threshold, then it is declared as a line with the parameters (θ, r_θ) of the intersection point.

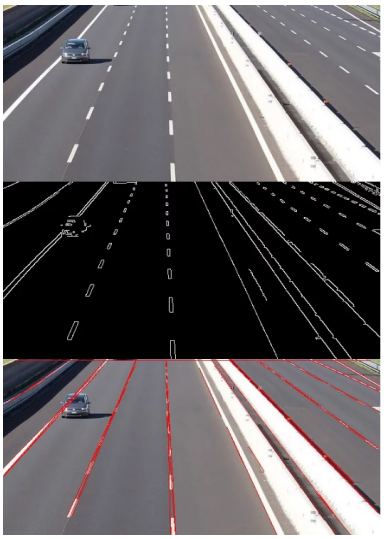


Figure 4: Recently developed applications of the Hough transform and its variants include the detection of lane lines in traffic and transportation science.

1.5 Polygonal approximation

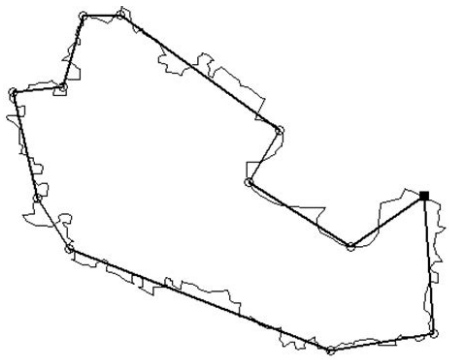


Figure 5: a polygonal approximation of a random shape

Let $P = (p_1, p_2, ..., p_n)$ be a polygonal curve or chain in the plane, consisting of n points p_i joined by line segments $p_i p_{i+1}$. In general P may be closed and self-intersecting. Polygonal curves occur frequently in pattern recognition either as representations of the boundaries of figures or as functions of time representing, e.g., speech.

In order to reduce the complexity of costly processing operations, it is often desirable to approximate a curve P with one that is composed of far fewer segments, yet is a close enough replica of P for the intended application.Following this approach, only the information necessary for the corresponding problem is taken into consideration, while any other information which only adds to the complexity of the problem is neglected. For instance, in facial recognition systems, it is not necessary to process the exact shape of a face, while some details (e.g. the distance between the eyes and the nose) are sufficient to detect differences between two faces.

An important instance of the problem is to determine a new curve $Q = (q_1, q_2, ..., q_m)$ such that

- m is significantly smaller than n
- the q_j are selected from among the p_i
- any segment $q_j q_{j+1}$ that replaces the chain $q_j = p_r, ..., p_s = q_{j+1}$ is such that the distance between $q_j q_{j+1}$ and each p_k, $r \leq k \leq s$, is less than some predetermined error tolerance ω

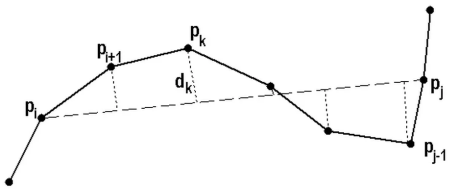


Figure 6: the ϵ criterion

Different notions of distance, or error criteria, lead to different algorithmic issues. Moreover, for each distance definition, there are two constrained optimization problems that are of interest, Min-# and Min-ϵ.

Definition.1.1.

1. **Min-#**. Given the error tolerance ω, find a curve $Q = (q_1, q_2, ..., q_m)$ satisfying the constraint such that m is minimum.

2. **Min-ϵ**. Given m, find a curve $Q = (q_1, q_2, ..., q_m)$ satisfying the constraint such that the error tolerance is minimized.

3. **Parallel-strip criterion**. All the vertices $p_i, ..., p_t$ lie in a parallel strip of width 2ϵ whose center line is collinear with $q_j q_{j+1}$.

4. Given an error measure m and a pair of indices $1 \leq i < j \leq n$, let $\delta_m(p_i p_j, P)$ denote the error of the segment $p_i p_j$ with respect to P under the error measure m. Intuitively, $\delta_m(p_i p_j, P)$ measures how well $p_i p_j$ approximates the portion of P between p_i and p_j . The error of a simplification Q where $p_{i_1} = p_1, ..., p_{i_k} = p_n$ of P is defined as $\delta_m(P, Q) = max_{1 \leq j < k} \delta_m(p_{i_j} p_{i_{j+1}}, P)$.

The main research focus has been the Min-# problem under the Hausdor error measure, also called ϵ-simplication. Near quadratic-time, $O(n^2 \log n)$, algorithms have been achieved for arbitrary polygonal curves. The quadratic-time barrier seems dicult to break, so approximation algorithms have been explored, which achieve an error no more than ϵ but compromise on m, the min-#. A popular algorithm is the Douglas-Peucker algorithm (iterative endpoint fitting), which, although easily implemented, does not guarantee performance and could be quadratic in the worst case. Now, near-linear performance with a guarantee has been achieved for simplifying monotone curves.

1.5.1 Douglas-Pecker Algorithm

A pseudocode for the Douglas-Peucker algorithm is presented below. Other, better performing algorithms will be presented later on.

Algorithm **DOUGLAS-PEUCKER**(P, ϵ)
Input: An ordered set of points P and the distance dimension ϵ
Output: An ordered set $Q \subseteq P$ with the minimum number of points such that the ϵ-criterion is satisfied

1. $Q \leftarrow \emptyset$
2. Assume p_1, p_n is the first and the last point of P, respectively. Insert p_1, p_n in Q.
3. **find** in P the farthest point from the line segment $\overline{p_1 p_n}$. Let that point be p_i.
4. — **if** $dist(p_i, \overline{p_1 p_n}) \geq \epsilon$:
   — — $P_1 \leftarrow \{p_1, ..., p_i\}, P_2 \leftarrow \{p_i, ..., p_n\}$
   — — $Q_1 \leftarrow$**DOUGLAS-PEUCKER**(P_1, ϵ)
   — — $Q_2 \leftarrow$**DOUGLAS-PEUCKER**(P_2, ϵ)
   — — Insert the points of Q_1, Q_2 in the right position of Q (between p_1 and p_n).
5. Return Q

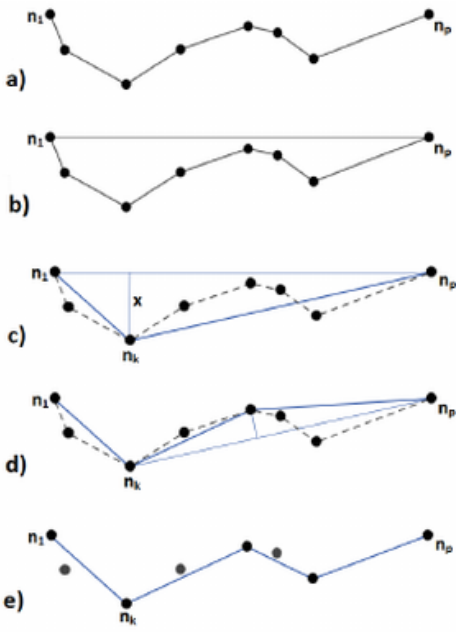


Figure 7: The Douglas-Peucker algorithm [4]

Now near-linear performance with a guarantee has been achieved for simplifying monotone curves. Such an algorithm was presented by Pankaj K. Agarwal,

Sariel Har-Peled, Nabil H. Mustafa, and Yusu Wang [1]. Their goal was to minimize the number of vertices of Q while ensuring that the error between Q and P is below a certain threshold. Their work is presented below.

Note. Not all proofs are included. Some proofs follow the same techniques as the ones presented in detail in this work, while some others are trivial.

1.5.2 The error measurement

We consider two error measures:

1. **Hausdorff error measure.** Let $\rho : R^d \times R^d \to R$ be a distance function. The distance between a point p and a line segment e is defined as $d(p, e) = min_{q \in e} \rho(p, q)$. The Hausdorff error under the metric ρ, also referred to as the ρ-Hausdorff error measure, is defined as $\delta_H(p_i p_j, P) = max_{i \leq k \leq j} d(p_k, p_i p_j)$.
   We consider the Hausdorff error measure under the so-called uniform metric defined as
   $$\rho(p, q) = \begin{cases} |p_y - q_y|, \text{ if } p_x = q_x \\ \infty, \text{ otherwise} \end{cases}$$

2. **Frechet error measure**. Given two curves $f : [a, a'] \to R^d$ and $g : [b, b'] \to R^d$ and a distance function $\rho : R^d \times R^d \to R$, the Frechet distance $\mathcal{F}(f, g)$ between them is defined as
   $$\mathcal{F}(f, g) = \inf_{\substack{\alpha:[0,1]\to[a,a'] \\ \beta:[0,1]\to[b,b']}} \max_{t \in [0,1]} \rho(f(\alpha(t)), g(\beta(t))), \tag{1}$$
   where α and β range over continuous and monotonically nondecreasing functions with $\alpha(0) = a, \alpha(1) = a', \beta(0) = b$, and $\beta(1) = b'$. For a pair of indices $1 \leq i < j \leq n$, the Frechet error of a line segment $p_i p_j$ is defined to be
   $$\delta_{\mathcal{F}}(p_i p_j, P) = \mathcal{F}(p_i p_j, P[p_i, p_j]),$$
   where $P[p, q]$ denotes the portion of P from p to q. Given curves P and Q (its vertices need not be a subset of the vertices of P), we say that Q is a weak ϵ-simplification of P if $\mathcal{F}(P, Q) \leq \epsilon$. Let $\hat{\kappa}_{\mathcal{F}}(\epsilon, P)$ denote the minimum number of vertices in a weak ϵ-simplification of P.

1.5.3 Hausdorff error measure algorithm

Theorem.1.2. Let P be an x-monotone polygonal curve in R^2 with n vertices, and let $\epsilon > 0$ be a parameter. We can compute in $O(n)$ time a simplification Q of P of size at most $\kappa_H(\epsilon/2, P)$ so that $\delta_H(P, Q) \leq \epsilon$, assuming that the distance between points is measured in uniform metric.

Definition.1.3. Let $P = p_1, ..., p_n$ be an x-monotone polygonal curve in R^2, and let $\epsilon > 0$ be a parameter. Let $D(p, r)$ be the disk of radius r centered at p

under uniform metric, i.e., $D(p, r)$ is a vertical segment of length $2r$ with p as its midpoint. A segment $p_j p_k$ is a valid segment, i.e., $\delta_H(p_j p_k, P) \leq \epsilon$, if and only if $p_j p_k$ intersects the segments $D(p_i, \epsilon)$ for all $j \leq i \leq k$.

That is a different (and more helpful) mathematical expression of the problem.

Lemma.1.4. Given an x-monotone polygonal curve P in R^2 and four indices $1 \leq i \leq l < r \leq j \leq n$,

$$\delta_H(p_l p_r, P) \leq 2\delta_H(p_i p_j, P).$$

Proof. Let $\epsilon' = \delta_H(p_i p_j, P)$. Let q_k (resp. q'_k) be the intersection point of $p_i p_j$ (resp. $p_l p_r$) with the vertical line passing through p_k if such an intersection point exists. By definition, $q_k \in \mathrm{D}(p_k, \epsilon')$ for all $i \leq k \leq j$. Hence $|p_l q_l| \leq \epsilon'$ and $|p_r q_r| \leq \epsilon'$, and it follows that $|q_k q'_k| \leq \epsilon'$ since P is an x-monotone curve. Therefore, by triangle inequality, we have that $|p_k q'_k| \leq |p_k q_k| + |q_k q'_k| \leq 2\epsilon'$.

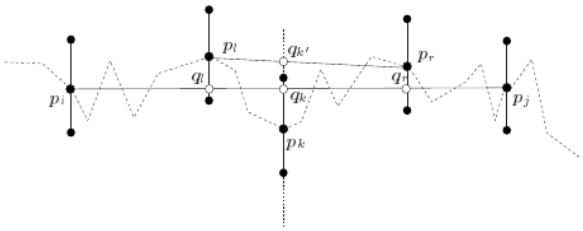


Figure 8: A geometric visualization of the theorem's proof

Algorithm.1.5. A greedy approach. Suppose we have added $p_{i_1}, ..., p_{i_{j-1}}$ to Q.

1. We find the smallest index $k > i_{j-1}$ so that $\delta_H(p_{i_{j-1}} p_{k+1}, P) > \epsilon$. We set $i_j = k$ and add p_{i_j} to Q.
2. We repeat this process until p_n is encountered.
3. We add p_n to the sequence and return the resulting sequence of vertices Q as the desired simplification.
4. Given the vertex $p_{i_{j-1}} = p_l$, we find p_k in time $O(k-l+1)$ as follows.Let $\mathrm{Cone}(p_i, p_j)$ be the set of rays emanating from p_i that intersect the vertical segment $D(p_j, \epsilon)$. $\mathrm{Cone}(p_i, p_j)$ is the cone bounded by rays emanating from p_i and passing through the endpoints of $D(p_j, \epsilon)$. Set

$$C(i, j) = \bigcap_{r=i+1}^{j} \mathrm{Cone}(p_i, p_r).$$

Then $\delta_H(p_ip_j, P) \leq \epsilon$ if and only if $p_j \in C(i,j)$. $C(i,r) = C(i,r-1) \cap$ Cone(p_i, p_r) can be computed in O(1) from $C(i,r1)$. To compute p_k , we visit the vertices p_r of P one by one, starting from p_{l+1} and maintain $C(l,r)$ by spending O(1) time at p_r. We stop as soon as we reach a vertex p_{k+1} such that $p_{k+1} \notin C(l, k+1)$. The total time spent in computing p_k is $O(k-l+1)$. Hence the total running time is $O(n)$.

Lemma.1.6. Q is an ϵ-simplification of P whose size is at most $\kappa_H(\epsilon/2, P)$.

The proof of the lemma is by induction on the number of vertices in Q.

1.5.4 Frechet error measure algorithm

Frechet Simplification. Let $P = \{p_1, ..., p_n\}$ be a polygonal curve in R^d , and let $\epsilon > 0$ be a parameter. Unlike before, we do not assume P to be x-monotone.

Let $\mathcal{F}(.,.)$ be as defined in (1), and let $d(.,.)$ denote the Euclidean distance between two points in R^d.

Lemma.1.7. Given two directed segments uv and xy in R^d, $\mathcal{F}(uv, xy) = \max\{d(u,x), d(v,y)\}$.

Lemma.1.8. Given a polygonal curve P in R^d and two directed segments uv and xy,

$$|F(uv, P) - F(xy, P)| \leq F(uv, xy)$$

.

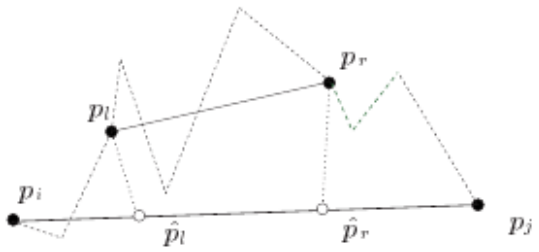


Figure 9: A geometric vizualisation of the theorem's basic mathematic properties.

Lemma.1.9. Let $P = \{p_1, p_2, ..., p_n\}$ be a polygonal curve in R^d . For $i \leq l \leq r \leq j$,

$$\delta_{\mathcal{F}}(p_lp_r, P) \leq 2\delta_{\mathcal{F}}(p_ip_j, P).$$

Algorithm.1.10. We use a greedy algorithm to compute a simplification Q of P, but with one twist. Suppose we have added $p_1 = p_{i_1}, ..., p_{i_j}$ to Q.

1. We find an index $k > i_j$ such that (i)$\delta_{\mathcal{F}}(p_{i_j}p_k, P) \leq \epsilon$, (ii) $\delta_{\mathcal{F}}(p_{i_j}p_{k+1}, P) > \epsilon$.

2. We set $i_{j+1} = k$ and add $p_{i_{j+1}}$ to Q.

3. We repeat this process until we encounter p_n. We then add p_n to Q.

Alt and Godau [2] have developed an algorithm that, given a pair $1 \leq i < j \leq n$, can determine in $O(j-i)$ time whether $\delta_{\mathcal{F}}(p_ip_j, P) \leq \epsilon$. Therefore, a first approach would be to add vertices greedily one by one, starting with the first vertex p_{i_j}, and testing each edge $p_{i_j}p_l$, for $l > i_j$, by invoking the Alt–Godau algorithm, until we find the index k. However, the overall algorithm could take $O(n^2)$ time. To limit the number of times that the Alt–Godau algorithm is invoked when computing the index k, we proceed as follows.

1. Using an exponential search, we determine an integer $l \geq 0$ such that $\delta_{\mathcal{F}}(p_{i_j}p_{i_j+2^l}, P) \leq \epsilon$ and $\delta_{\mathcal{F}}(p_{i_j}p_{i_j+2^{l+1}}, P) > \epsilon$.

2. By performing a binary search in the interval $[2l, 2l+1]$, we determine an integer $r \in [2l, 2l+1]$ such that $\delta_{\mathcal{F}}(p_{i_j}p_{i_j+r}, P) \leq \epsilon$ and $\delta_{\mathcal{F}}(p_{i_j}p_{i_j+r+1}, P) > \epsilon$.

Note that in the worst case, the asymptotic costs of the exponential and binary searches are the same. Set $k = i_j + r$. Since computing the value of k requires invoking the Alt– Godau algorithm $O(l) = O(\log n)$ times, each with a pair (α, β) such that $\beta - \alpha \leq 2r$,the total time spent in computing the value of k is $O((k-i_j+1)\log n)$. Hence, the overall running time of the algorithm is $O(n\log n)$.

```
ALGORITHM FS (P, ε)
  Input: P = ⟨p_1, ..., p_n⟩; ε > 0.
  Output: P' ⊆ P such that δ_F(P', P) ≤ ε.
begin
  j = 1; i_j = 1; P' = ⟨p_{i_j}⟩;
  while ( i_j < n ) do
    l = 0;
    while (δ_F(p_{i_j} p_{i_j + 2^{l+1}}, P) ≤ ε) do
      l = l + 1;
    end while
    low = 2^l; high = 2^{l+1};
    while (low < (high − 1)) do
      mid = (low + high)/2;
      if (δ_F(p_{i_j} p_{i_j + mid}, P) ≤ ε)
        low = mid;
      else high = mid;
    end while
    i_{j+1} = i_j + low; P' = P' · ⟨p_{i_{j+1}}⟩; j = j + 1;
  end while
end
```

Figure 10: A pseudocode for the Frechet error measure algorithm

Theorem.1.11. Given a polygonal curve $P = \{p_1, ..., p_n\}$ in R^d and a parameter $\epsilon > 0$, we can compute in $O(n\log n)$ time an ϵ-simplification Q of P under the Frechet error measure so that $|Q| \leq \kappa_{\mathcal{F}}(\epsilon/2)$.

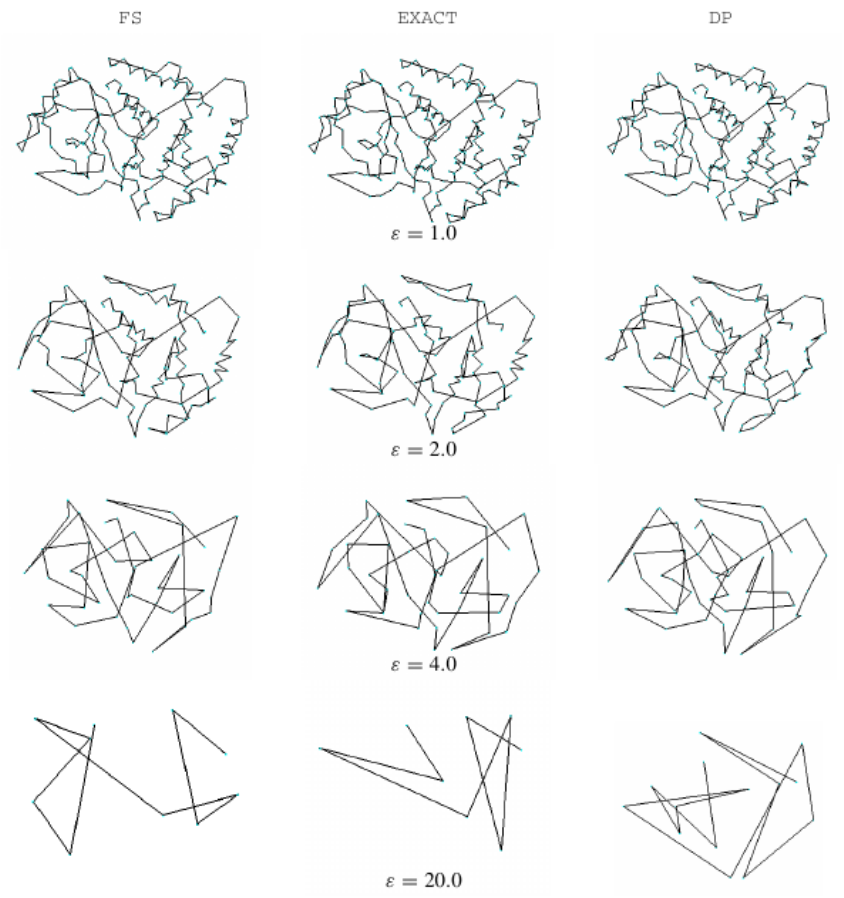


Figure 11: The algorithm in action. DP=Douglas-Pecker algorithm, FS=Frechet error measure algorithm under L_2-metric.

2 Deep learning

2.1 Introduction

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer.

Conventional machine-learning techniques were limited in their ability to

process natural data in their raw form. **For decades, constructing a pattern-recognition or machine-learning system required careful engineering and considerable domain expertise to design a feature extractor that transformed the raw data** (such as the pixel values of an image) **into a suitable internal representation or feature vector from which the learning subsystem**, often a classifier, could detect or classify patterns in the input.

Representation learning is a set of methods that allows a machine to be fed with raw **data and to automatically discover** the representations needed for detection or classification. Deep-learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple **but non-linear** modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned.

An image, for example, comes in the form of an array of pixel values, and the learned features **in the first layer** of representation typically represent the presence or absence of edges at particular orientations and locations in the image.
The second layer typically detects motifs by spotting particular arrangements of edges, regardless of small variations in the edge positions.
The third layer may assemble motifs into larger combinations that correspond to parts of familiar objects, and subsequent layers would detect objects as combinations of these parts.
The key aspect of deep learning is that these layers of features are not designed by human engineers: they are learned from data using a general-purpose learning procedure

2.2 Supervised learning

The most common form of machine learning, deep or not, is supervised learning. Imagine that we want to build a system that can classify images as containing, say, a house, a car, a person or a pet. We first collect a large data set of images of houses, cars, people and pets, each labelled with its category. During training, the machine is shown an image and produces an output in the form of a vector of scores, one for each category. We want the desired category to have the highest score of all categories, but this is unlikely to happen before training. We compute an objective function that measures the error (or distance) between the output scores and the desired pattern of scores. The machine then modifies its internal adjustable parameters to reduce this error. These adjustable parameters, often called weights, are real numbers that can be seen as 'knobs' that define the input–output function of the machine. In a typical deep-learning system, there may be hundreds of millions of these adjustable weights, and hundreds of millions of labelled examples with which to train the machine.

2.3 The Threshold of Linear Classifiers

Many of the current practical applications of machine learning use linear classifiers on top of hand-engineered features. A two-class linear classifier computes a weighted sum of the feature vector components. If the weighted sum is above a threshold, the input is classified as belonging to a particular category.
Since the 1960s **we have known that linear classifiers can only carve their input space into very simple regions, namely half-spaces separated by a hyperplane. But problems such as image and speech recognition require the input–output function to be insensitive to irrelevant variations of the input**, such as variations in position, orientation or illumination of an object, or variations in the pitch or accent of speech, while being very sensitive to particular minute variations (for example, the difference between a white wolf and a breed of wolf-like white dog called a Samoyed).

At the pixel level, images of two Samoyeds in different poses and in different environments may be very different from each other, whereas two images of a Samoyed and a wolf in the same position and on similar backgrounds may be very similar to each other.

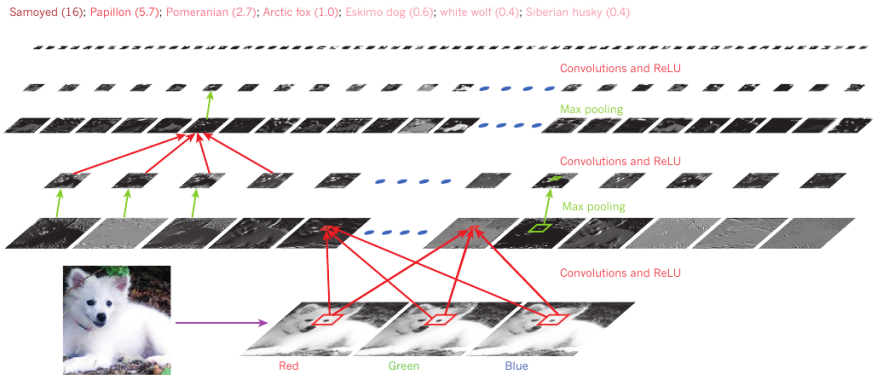


Figure 12: The outputs (not the filters) of each layer (horizontally) of a typical convolutional network architecture applied to the image of a Samoyed dog.

A linear classifier, or any other 'shallow' classifier operating on raw pixels could not possibly distinguish the latter two, while putting the former two in the same category. This is why shallow classifiers require a good feature extractor that solves the selectivity–invariance dilemma — one that produces representations that are selective to the aspects of the image that are important for discrimination, but that are invariant to irrelevant aspects such as the pose of the animal. To make classifiers more powerful, one can use generic non-linear features, as with kernel methods, but generic features such as those arising with the Gaussian kernel do not allow the learner to generalize well far from the training examples. The conventional option is to hand design

good feature extractors, which requires a considerable amount of engineering skill and domain expertise. But this can all be avoided if good features can be learned automatically using a general-purpose learning procedure. This is the key advantage of deep learning.

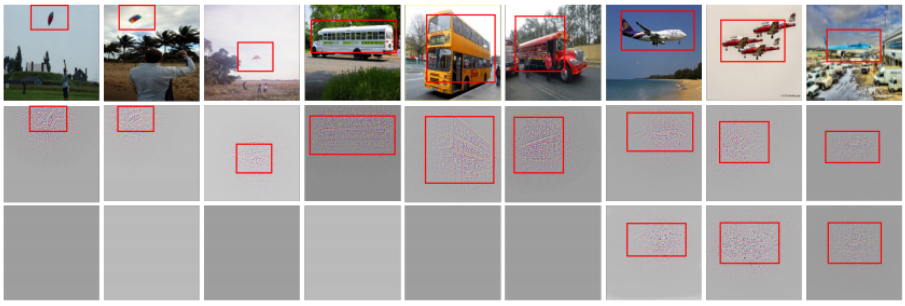


Figure 13: Visualization of the local response region of filters in the last convolutional layer of CNN on different images. The second row images show the local response region of filter α. Filter α is activated by all the images. The third row images show the local response region of filter β. Filter β is only activated by the "flight" images

2.4 Convolutional neural networks-CNN Intuition

2.4.1 Overview

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

There are four key ideas behind ConvNets that take advantage of the properties of natural signals: local connections, shared weights, pooling and the use of many layers. The architecture of a typical ConvNet (see Samoyeds image) is structured as a series of stages. The first few stages are composed of two types of layers: convolutional layers and pooling layers. In the following subsection we will give a brief explanation of these 2 layers which are also the most important ones.

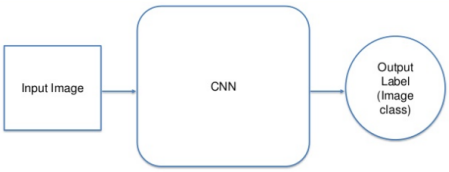


Figure 14: CNN

2.4.2 Convolution, Pooling Steps in CNN - Brief explanation

- **The Convolution Step** ConvNets derive their name from the "convolution" operator. The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data.

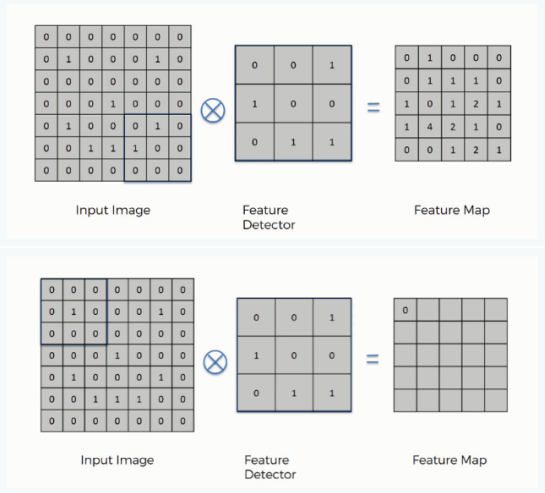


Figure 15: The Convolution operation. The output matrix is called Convolved Feature or Feature Map

- **The Pooling Step**

  Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc.

  In case of Max Pooling, we define a spatial neighborhood (for example, a 22 window) and take the largest element from the rectified feature map within that window. Instead of taking the largest element we could also

take the average (Average Pooling) or sum of all elements in that window. In practice, Max Pooling has been shown to work better.

An **example** that illustrates these steps for handwritten digits can be found in this link

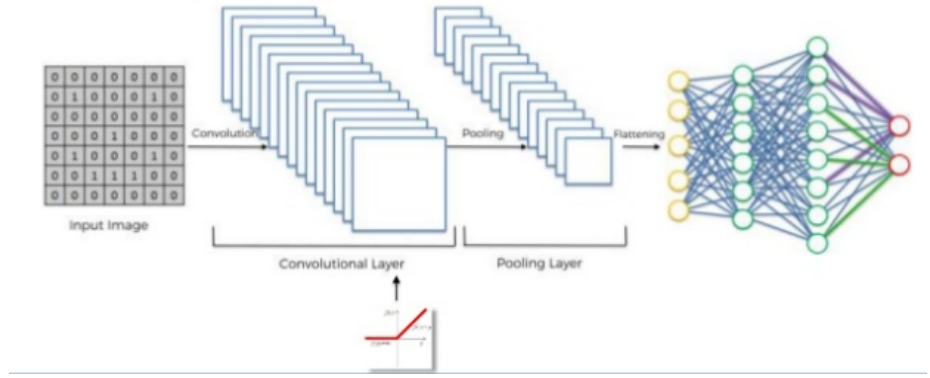


Figure 16: CNN-abstract representation (1)

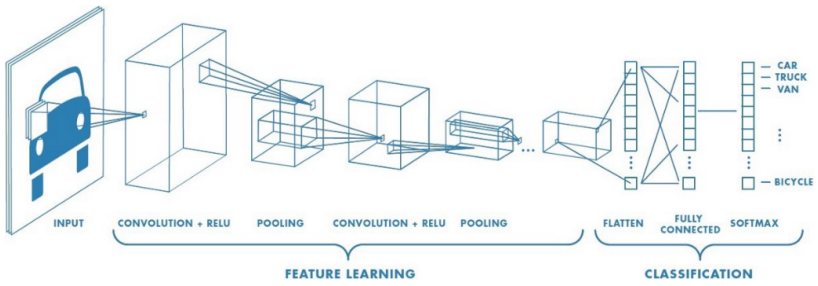


Figure 17: CNN-abstract representation (2)

3 Conclusion

In the above sections, in order to investigate the pattern recognition problem a number of different approaches were discussed. Considering all of the above, solving the pattern recognition problem by using CNN techniques is much simpler than following the mathematical and algorithmic analysis. Since the development of artificial intelligence algorithms were required for the development of the CNN techniques, they have only been recently studied. Ever since their development, CNN techniques are used almost exclusively, whilst other techniques have been neglected. However, it is worthwhile to study the mathematical and

algorithmic approach, because, as a result of this study, many useful tools and techniques have been explored and are nowadays commonly used in other fields of computer science.

We acknowledge that our project is only an overview of the entire pattern recognition problem. In addition, there is a number of issues regarding the problem that were not discussed in this current study, which is expected given the nature of this problem. When the entire pattern recognition problem is investigated, various sub-problems occur which require distinct different solutions, and, thus, cannot be studied in such a short period of time. In order for our project to be as complete as possible, while all the fundamental parts of the problem were presented, only the most challenging ones were studied in depth. In conclusion, this brief introduction to the pattern recognition problem has substantially enhanced our knowledge in the subject and has sparked an interest for future investigation.

References

[1] Pankaj K Agarwal et al. "Near-Linear Time Approximation Algorithms for Curve Simplification1". In: *Algorithmica* 42 (2005), pp. 203–219.

[2] Michael M Bronstein et al. "Geometric deep learning: going beyond euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[3] J Oseph O'ROURKE and Godfried T. Toussaint. "54 PATTERN RECOGNITION". In: 2017.

[4] https://en.wikipedia.org/wiki/Ramer–Douglas–Peucker