Some Practical Applications of Soft Computing and Data Mining

Hung T. Nguyen[1], Nadipuram R. Prasad[2],
and Vladik Kreinovich[3]

[1]Department of Mathematical Sciences
[2]Klipsch School of Electrical and
Computer Engineering
New Mexico State University
Las Cruces, NM 88003-8001, USA
emails {hunguyen,rprasad}@nmsu.edu

[3]Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
email vladik@cs.utep.edu

Abstract

Traditional data mining techniques mainly deal with a search for patterns in traditional-type databases, where data consists of numbers and words. In many application areas, however, data is more complicated: real-life data often comes not as a result of measuring a few quantities, but rather as an image from a camera, and this image can also change dynamically. In this paper, we present several examples of how soft computing is related to mining such data.

1 Introduction

Traditional data mining techniques mainly deal with a search for patterns in traditional-type databases, where data consists of numbers and words. In many application areas, however, data is more complicated:

- Real-life data often comes not as a result of measuring a few quantities, but rather as an image from a camera. Each image contains much more information than a typical data record, so we cannot simply use traditional methods to mine through image database.
- Another reason why real-life data is more complex than the data stored in traditional databases is that traditional databases (such as databases describing business transactions, purchases, etc.) deal mainly with rather slowly changing processes. For such processes, it is sufficient, for each quantity, to store its current value and to update this value periodically. In many real-life situations, however, we deal with a fast changing process. A numerical characteristic of such a process can no longer be characterized by a single number, it can only be described by a speedily changing function of time.
- In some practical cases, both these problems are present, i.e., we need to mine the database of dynamically changing images.

In this paper, we present several case studies of how soft computing is related to mining such data:

- Our first two case studies are analysis of images of cotton, both cotton in the field and ginned cotton. We use fuzzy, neural, and more traditional geometric techniques to detect and classify different types of trash in ginned cotton and different insects in the cotton field.

- Our third case study is an automated tracking system for tracking high-speed targets such as airplanes, missiles, etc. Here, we have an example of highly dynamical data, and moreover, data which is represented largely by images.

2 First Case Study: Classification of Trash in Ginned Cotton

2.1 A practical problem: brief description

The main use of cotton is in textile industry; for that purpose, we only need cotton fiber called *lint*. Mechanical harvesters collect fiber together with the seeds. To separate lint from the seeds and from other non-lint material, a special process called *ginning* is used. Ginned cotton consists primarily of lint, but some non-lint material (*trash*) is left. For the further textile processing, it is important to know how much trash of what type is left.

In principle, it is possible to detect the amount and type of trash by visual inspection, because trash is usually of different color than the whitish lint and is thus clearly visible. The problem with visual inspection is that the visual inspection of all 15 to 19 million bales of cotton annually produced in the USA is a very time-consuming and expensive process. It is therefore desirable to develop an automatic system for the analysis of trash in ginned cotton (see, e.g., [6, 7]).

2.2 The need for soft computing

In general, to automate expert skills, we can do two things:

- first, we can ask experts how they perform their tasks, and then try to formalize the resulting rules;
- second, we can record the expert classification, and then try to teach the automated system to follow the expert decisions on the given examples.

Expert usually formulate their rules by using words from a natural language like "small", "large", etc. Transforming such "fuzzy" rules into precise formulas is one of the main objectives of a special formalism called *fuzzy logic*; so, we must use fuzzy logic techniques. To make a system learn from examples of expert classification, we must use a universal learning technique, e.g., the technique of *neural networks*.

In many real-life situations, we have both expert rules *and* samples of expert decisions; we must therefore be able to combine the fuzzy technique of handling rules and the neural techniques of handling examples. To facilitate the combination of fuzzy and neural techniques (as well as several other intelligent techniques), an umbrella approach has been developed called *soft computing*. So, soft computing is necessary for automating expert skills.

2.3 The need for data mining

Traditional soft computing techniques for formalizing and automating expert skills work well only when the number n of input values $x_1, \ldots, x_n$ which determine the expert's decision is small. Indeed:

- In *fuzzy logic techniques*, to elicit the expert rules, we usually select a few words like "small", "medium", "large" which describe each of the input quantities, and then ask the expert to describe his decision for all possible combination of such words, such as "x_1 is small, x_2 is medium, ...". This elicitation procedure is feasible when n is small. For large n, however, this procedure is no longer feasible: even for three possible states of each variable, we need 3^n combinations of input values. When n is large (e.g., if the input is a picture described by $n \approx 10^6$ pixel values), this number of combinations is unrealistically large.
- Similarly, to train a *neural network*, we normally need a number of examples to be at least as large as the square n^2 of the number n of input variables (see, e.g., [2] and references therein). This empirical fact can be easily understood:

- In the simplest possible case when the dependence of the decision y on the inputs x_i is linear, i.e., when $y = c_0 + c_1 \cdot x_1 + \ldots + c_n \cdot x_n$, we need at least $n+1$ examples to determine all $n+1$ coefficients c_i which describe the desired dependence.
- Neural networks describe, in general, non-linear dependencies. The simplest possible non-linear dependence is *quadratic*, when

$$y = c_0 + \sum_{i=1}^{n} c_i \cdot x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \cdot x_i \cdot x_j.$$

  To describe the general quadratic dependence, we need $\approx n^2$ different coefficients c_i; therefore, to determine all these coefficients, we need at least $\approx n^2$ different examples.

When n is small, it is easy to record n^2 examples of the experts using their expertise. Alas, when n becomes large, e.g., when the input is an image with $n \approx 10^6$, the successful application of a neural network requires an unrealistic number of $n^2 \approx 10^{12}$ examples.

Summarizing: when the number n of input variables is large, we cannot directly use traditional soft computing techniques. Therefore, instead of simply using all n input variables, we must find a smaller number of relevant *combinations* $y_1, \ldots, y_m$ of these input variables, and then use these combinations y_j as the new inputs. In other words, to be able to use soft computing techniques for automating expert skills, we must first use *data mining* to find relevant combinations of input variables.

In this section (and in the following section), we will show how this data mining can be performed for the problems of cotton analysis.

2.4 Towards formalization of the corresponding data mining problem

Since trash is clearly visible on the lint background, it is natural to take a photo of a cotton bale, and then run a computer program to analyze this photo. Our goal is to separate trash from lint; since trash is of different color than the lint, we can ignore the details about the intensities of different pixels and use a threshold on intensity to transform the original image into a black-and-white one: points in which the intensity is above the threshold are treated as white (i.e., as lint), and points in which the intensity is below the threshold are treated as black (i.e., as trash).

As a result, we get a black-and-white picture in which several pieces of trash are present on the white background. Pieces of trash can have complicated shapes. The user needs a simple classification of these shapes. A natural way of classifying different shapes is to describe several simple approximate shapes, and then to classify a given piece of trash based on which simple shape it resembles most. So, to develop a good classification of trash in cotton, we need to find a good approximating family of sets.

Because of the large volume of cotton processing, even a small gain in classification quality can lead to a large economic benefit. It is therefore desirable to look not simply for a *good* approximating family of sets, but rather for a family which is *optimal* in some reasonable sense.

Of course, the more parameters we allow, the better the approximation. So, the question can be reformulated as follows: for a given number of parameters (i.e., for a given dimension of approximating family), which is the best family? In this section, we use a geometric formalism developed in [5] and [15] to formalize and solve this problem.

2.5 Formalizing the problem

In this formalization, we will, in effect, follow [5] and [15].

The pieces of trash are usually smooth lines or areas with smooth boundaries, so it is reasonable to restrict ourselves to families of sets with analytical boundaries. By definition, when we say that a piece of a boundary is analytical, we mean that it can be described by an equation $F(x, y) = 0$ for some analytical function

$$F(x, y) = a + b \cdot x + c \cdot y + d \cdot x^2 + e \cdot x \cdot y + f \cdot y^2 + \ldots$$

So, in order to describe a family, we must describe the corresponding class of analytical functions $F(x, y)$.

Since we are interested in families of sets which are characterized by finitely many parameters (i.e., in finite-dimensional families of sets), it is natural to consider finite-dimensional families of functions, i.e., families of the type

$$\{C_1 \cdot F_1(x,y) + \ldots + C_d \cdot F_d(x,y)\},$$

where $F_i(z)$ are given analytical functions, and $C_1, \ldots, C_d$ are arbitrary (real) constants. So, the question becomes: which of such families is the best?

When we say "the best", we mean that on the set of all such families, there must be a relation $\succeq$ describing which family is better or equal in quality. This relation must be transitive (if A is better than B, and B is better than C, then A is better than C).

This relation is not necessarily asymmetric, because we can have two approximating families of the same quality. However, we would like to require that this relation be *final* in the sense that it should define a unique *best* family $A_{\rm opt}$ (i.e., the unique family for which $\forall B\,(A_{\rm opt} \succeq B)$. Indeed:

- If none of the families is the best, then this criterion is of no use, so there should be *at least one* optimal family.
- If *several* different families are equally best, then we can use this ambiguity to optimize something else: e.g., if we have two families with the same approximating quality, then we choose the one which is easier to compute. As a result, the original criterion was not final: we get a new criterion ($A \succeq_{\rm new} B$ if either A gives a better approximation, or if $A \sim_{\rm old} B$ and A is easier to compute), for which the class of optimal families is narrower. We can repeat this procedure until we get a final criterion for which there is only one optimal family.

The exact shape depends on the choice of a starting point, on the orientation of the camera, and on the choice of the zoom. It is reasonable to require that if we change the starting point, the orientation, or the zoom, the relative quality of different approximating families should not change. In other words, it is reasonable to require that the relation $A \succeq B$ should not change if shift, rotate, or scale the image; i.e., the relation $A \succeq B$ should be shift-, rotation- and scale-invariant.

These requirements can be formalized as follows:

Definition 1. *Let $d > 0$ be an integer. By a d-dimensional family, we mean a family A of all functions of the type*

$$\{C_1 \cdot F_1(x,y) + \ldots + C_d \cdot F_d(x,y)\},$$

where $F_i(z)$ are given analytical functions, and $C_1, \ldots, C_d$ are arbitrary (real) constants. We say that a set S is defined by this family A if for some function $F \in A$, all points (x,y) from the border ∂S of the set S satisfy the equation $F(x,y) = 0$.

Definition 2.

- *By an optimality criterion, we mean a transitive relation $\succeq$ on the set of all d-dimensional families.*
- *We say that a criterion is final if there exists one and only one optimal family, i.e., a family $A_{\rm opt}$ for which $\forall B\,(A_{\rm opt} \succeq B)$.*
- *We say that a criterion $\succeq$ is shift- (corr., rotation- and scale-invariant) if for every two families A and B, $A \succeq B$ implies $TA \succeq TB$, where TA is a shift (rotation, scaling) of the family A.*

Proposition 1. *Let $d \le 4$, let $\succeq$ be a final optimality criterion which is shift-, rotation- and scale-invariant, and let $A_{\rm opt}$ be the corresponding optimal family. Then, the border of every set defined by this family $A_{\rm opt}$ is either a straight line interval, a circle, or a circular arc.*

Proposition 1 was first proven in [5, 15]. For the convenience of the readers who are mainly interested in practical results, the proofs of all the propositions are given in the special Appendix.

2.6 Discussion

The only shape which actually bounds a 2-D set is a circle which bounds a disk. So, as a result of this proposition, we have the following trash shapes:

- straight line intervals,
- circular arcs, and
- disks.

When the disk is small, we can view it as a point, which leads us to the fourth possible approximate shape of cotton trash:

- points.

This classification is in perfect agreement with the existing empirical classification of trash into:

- *bark1* (approximately circular arcs),
- *bark2* (straight line segments),
- *leaf* (disks), and
- *pepper trash* (points).

The names of these types of trash come from their physical meaning, with the only exception of *pepper trash* which refers to broken or crushed pieces of leaf.

2.7 Implementation details: creating an image

We have used this geometric classification to develop a prototype system for classifying trash. In our system, images (640×480) are acquired using a 3-chip CCD Sony color camera. The imaging hardware consists of a Matrox IM-1280 imaging board and CLD acquisition board. The pixel resolution is 0.13 mm (0.005 inches).

The acquired images are flat field corrected for spatial illumination non-uniformity. Each acquired color image (RBG) is converted into hue, luma (intensity), and saturation (HLS) color space (see, e.g., [10]), and a threshold on intensity is used to create a black-and-white image.

2.8 Implementation details: selecting image characteristics

To classify trash, we selected several reasonable geometric characteristics from the list of standard characteristics of a black-and-white image described, e.g., in [10].

2.8.1 First image characteristic – solidity – measures the image's convexity

First, we noticed that some of our shapes are convex sets (disks – leaves, points – pepper, and straight line segments – bark2), while some are not (circular arcs – bark1). By definition, a convex set is a set S whose convex hull $\mathrm{co}(S)$ coincides with itself (i.e., $\mathrm{co}(S) = S$); the closer the convex hull $\mathrm{co}(S)$ to the set itself S, the more convex is this set S. Therefore, as a characteristic of convexity, one can use the ratio between the area A of the original set S (measured, e.g., by the total number of pixels in the set S) and the area of its convex hull $\mathrm{co}(S)$. This ratio is equal to 1 for a convex set and is smaller than 1 for non-convex sets.

In computer imaging, the area of a convex hull is called the *convex area*, and the ratio of the area and the convex area is called the *solidity* of the set S. So, we expect that:

- for non-linear shapes such as bark1, solidity is much smaller than 1;
- while linear shapes such as bark2, leaf, and pepper trash, should have solidity close to 1.0.

The experimental analysis shows that, indeed, for bark1, solidity is typically less than 0.5, while for other types of trash, it is typically close to 1. Thus, solidity enables us to distinguish between bark1 and other trash types.

2.8.2 Second image characteristic – difference – measures the image's rotation invariance

Using solidity, we can distinguish between bark1 and other types of trash. To further distinguish between the three remaining types of trash, we can use the fact that our classification was based on invariance with respect to geometric transformations: shift, rotation, and scaling. It is therefore reasonable to check the invariance of the resulting shapes. Let us check these invariances one by one.

None of our trash shapes are exactly shift-invariant, so checking for this invariance does not help in distinguishing between different types of trash.

Let us now consider rotation invariance. Bark2 (straight line segment) is not rotation-invariant, while leaf (circle) and pepper trash (point) are rotation-invariant. It is therefore desirable to find an image characteristic which will enable us to tell whether a given image is rotation-invariant; based on this characteristic, we will then be able to distinguish between bark2 and the remaining trash type (pepper and leaf).

In selecting the first characteristic, we used the area of the original image and the area of its convex hull. Neither of these two characteristics can distinguish between rotation-invariant and rotation-non-invariant shapes, because both the area A and the area of the convex hull are rotation-invariant. Instead, we can use a similar standard image characteristic which is *not* rotation-invariant: the area of the *bounding box*. The bounding box is defined as the smallest box (= rectangle parallel to coordinate axes) which contains the desired image. Its area is equal to the product $X_f \times Y_f$, where X_f and Y_f are *ferrets* – lengths of the image's projections on the corresponding axes.

In general, the area of the bounding box changes when we rotate the coordinate axes. It is therefore reasonable to take, as the second image characteristic, the difference E^{dif} between the original bounding box area and the bounding box area corresponding to the rotated coordinate system.

To finalize the selection of this characteristic, we must select the rotation angle. Some angle are not very useful:

- This angle should not be too large: e.g., rotation by $90°$ simply swaps x and y axes without changing the bounding box and its area, so the corresponding difference is always equal to 0.
- Similarly, this angle cannot be too small: Indeed, real-life leaf and pepper trash shapes are only approximately rotation-invariant, so for these types, the difference E^{dif} is close t0 0 (i.e., small) but, most probably, different from 0. If the rotation angle is small, then even for bark2, the rotated bounding box is close to the original one; therefore, the two areas are close, and the difference E^{dif} between these two areas is small. Hence, for a small rotation angle, the difference E^{dif} will be small for all trash types, and we will not be able to use this characteristic to distinguish between different trash types.

Therefore, for the difference characteristic to be useful, it is important to select an appropriate rotation angle. Similarly to the above text, we can formulate the problem of choosing an appropriate rotation angle as an optimization problem under an (arbitrary) reasonable optimality criterion. Before we formulate the result, let us make two comments:

- Since rotation by $90°$ leaves the bounding box area unchanged, it is sufficient to only consider *acute* angles, i.e., angles from 0 to $90°$ ($= \pi/2$ radians).
- It is reasonable to assume that the criterion does not change if we simply swap x and y axes. In geometric terms, this "swap" can be described as follows: the rotation angle can be defined, e.g., as the angle α between the original x-axis $0x$ and the new x-axes $0x'$. The result of swapping the original x axis $0x$ is the original y axis $0y$; so, the angle between the new x axis $0x'$ and the swapped original x axis is simply the angle between $0x'$ and $0y$, which is equal to $90° - \alpha$. Thus, in geometric terms, the swap means replacing an angle α by its complement $90 - \alpha$.

Now, we are ready to formulate the result:

Definition 3.

- *By an optimality criterion, we mean a transitive relation* $\succeq$ *on the set* $[0,90]$ *of all acute angles.*
- *We say that a criterion is final if there exists one and only one optimal angle, i.e., an angle* $\alpha_{\rm opt}$ *for which* $\forall\beta\,(\alpha_{\rm opt}\succeq\beta)$.
- *We say that a criterion* $\succeq$ *is swap-invariant if for every two angles* α *and* β, $\alpha\succeq\beta$ *implies* $T(\alpha)\succeq T(\beta)$, *where* $T(\alpha)=90-\alpha$.

Proposition 2. *Let* $\succeq$ *be an arbitrary final optimality criterion which is swap-invariant. Then, the optimal angle* $\alpha_{\rm opt}$ *is equal to* $45°$.

So, the optimal choice of the difference characteristic is the difference between the original bounding box area and the area of the bounding box after the rotation by $45°$.

Comments.

- We are checking rotation-invariance by using only one rotation. It is therefore quite possible that the image is not rotation-invariant (i.e., it is a straight line segment), but for the chosen angle, the bounding box areas are actually equal. However, this is only possible for a single rotation angle, and since the orientation of trash is random, this accidental coincidence will happen with probability 0. So, with probability 1, the difference $E^{\rm dif}$ does enable us to distinguish between the shapes which are rotation-invariant (pepper and leaf) and which are not (bark2).
- In general, the checking of rotation invariance is intended for distinguishing between bark2 and leaf or pepper; we assume that bark1 have already been classified. However, in reality, we may have intermediate situations in which a circular arc (bark1) is almost linear and so, bark1 (which is normally characterized by small solidity) is not easily distinguishable from bark2 (which is normally characterized by large solidity). For these situations of medium (intermediate) solidity, we can use the new difference characteristic $E^{\rm dif}$ to distinguish between bark1 which is more rotation-invariant and bark2 which is less rotation invariant.

2.8.3 Third image characteristic – area – distinguishes between points (pepper trash) and circles (leaves)

Using the first image characteristic (solidity), we can distinguish between bark1 and other types of trash (bark2, pepper, and leaf):

- low solidity means bark1, while
- larger values of solidity can mean one of the remaining three trash types.

So, if the solidity is low, we know that the trash is of type bark1. If the solidity is high, we can use the second image characteristic (difference) to distinguish between bark2 and pepper or leaf:

- large value of the difference means bark2, while
- small values of the difference mean that the trash is either pepper or leaf.

Hence, to complete the classification of trash type, the only remaining task is to separate pepper trash from leaf trash. From the invariance viewpoint, they are both rotation-invariant, and the difference between these two types is that pepper is scale-invariant, while leaf is not. Therefore, to distinguish between these two types, we can use the difference between, e.g., the area A of the original image and the area of the scaled image.

If we use scaling with a coefficient λ, then the area of the scaled image is equal to $\lambda^2\cdot A$ and therefore, the desired difference is equal to $C\cdot A$, where we denoted $C=\lambda^2-1$. Thus:

- if the value of $C\cdot A$ is small, it is most probably pepper;
- if the value of $C\cdot A$ is large, then it is most probably leaf.

By appropriately changing which values we consider small and which values we consider large, we can always select $C = 1$. For this selection, the new difference characteristic is simply the area of the image. Therefore, as our third image characteristic, we select the image's area A.

This selection can be explained in common sense geometric terms, without using invariance:

- pepper trash is – approximately – a point, while
- leaf trash is – approximately – a circle.

A point is a degenerate circle, of radius 0 and of area 0. So, to distinguish between pepper and leaf trash, we can use the area A of the trash image. In other words, if we already know, from the values of the first two characteristics (solidity and difference), that the trash is either of pepper type or of leaf type, then we can use the third characteristic – area A – to distinguish between pepper trash and leaf trash:

- if the area is small ($A \approx 0$), then the trash type is most probably pepper trash;
- if the area is not small, the trash is most probably a leaf.

2.9 Trash classification in terms of fuzzy rules

While describing the three characteristics, we showed natural rules which use the values of these characteristics to determine the trash type. These rules, however, cannot be directly implemented because they use words from natural language like "small", "large", etc. Specifically, we use two words "small" and "large" to describe the difference and the area, and three words "small", "medium", and "large" to describe solidity.

We therefore need fuzzy logic to formalize these rules. In this formalization, we used the Adaptive Network-Based Fuzzy Inference System ANFIS. In our case, we have 3 values for the first characteristic, and 2 values each for the second and the third; thus, we have a total of $3 \times 2 \times 2 = 12$ possible combinations of characteristics, for each of which we know the trash type. We therefore formulated the corresponding 12 rules of the type "if solidity is large, difference is small, and area is small, then bark1", and used a neural network to adjust the shapes of the corresponding $3 + 2 + 2$ membership functions so as to achieve the best possible trash classification.

2.10 Practical results

The resulting systems achieves a 98% correct classification of trash – a much higher percentage than the previously known methods; for further details see, e.g., [11, 12].

3 Second Case Study: Classification of Insects in the Cotton Field

3.1 A practical problem: brief description

In addition to trash, cotton contains insects. Some of these insects destroy the cotton crop; to preserve the crop, farmers use insecticides. Among other crops, cotton is especially vulnerable to insects; as a result, world-wide, more insecticides are used on cotton than on any other crop.

The problem is that it is often difficult to distinguish between harmful and harmless insects; as a result, insecticides are used even when only harmless insects are present, thus destroying the (often useful) insects and, in general, polluting the environment. It is therefore desirable to be able to distinguish between useful and harmful insects.

Expert entomologists can easily distinguish between harmful and harmless insects, but there are not enough experts to monitor every cotton field. It is therefore necessary to design an automated system which would analyze the image of a cotton sample and tell whether it contains harmful insects.

Similarly to the first case study, we need methods of *soft computing* to formalize expert rules, and methods of *data mining* to find the relevant combinations of input variables. Let us describe how these methods can be used in classifying insects.

3.2 Formalization of the corresponding data mining problem. 1: We should use ellipses to approximate insect shapes

Similarly to the first case study, we can use black-and-white images, and approximate the desired images (sets) by sets from a certain family. There are, however, two differences between the problem of classifying trash and the problem of classifying insects:

- The first difference is that to classify trash, it was sufficient to use a very crude approximation by sets from a 4-parametric family. To classify insects, we need a more accurate approximation and thus, we need a larger family of approximating sets.
- The second difference is that trash, by definition, may contain *pieces* of leaves, bark, etc., while the insects are usually viewed *whole*. Therefore, when classifying trash, we could use shapes for which the boundaries satisfied the equation $F(x,y)=0$ but which contained only a part of all the points (x,y) which satisfy this equation: e.g., we considered a straight line segment, which is only a *piece* of a straight line $F(x,y)=a\cdot x+b\cdot y+c=0$. For insects, we must consider only the shapes for which the corresponding equation $F(x,y)=0$ bounds the whole image.

Definition 4. *Let A be an d-dimensional family, i.e., a family A of all functions of the type*

$$\{C_1\cdot F_1(x,y)+\ldots+C_d\cdot F_d(x,y)\},$$

where $F_i(z)$ are given analytical functions, and $C_1,\ldots,C_d$ are arbitrary (real) constants. We say that a bounded set S is defined as a whole by a family A if for some function $F\in A$, the border ∂S of the set S coincides with the set of all points (x,y) for which $F(x,y)=0$.

Proposition 3. *Let $d\le 6$, let $\succeq$ be a final optimality criterion which is shift-, rotation- and scale-invariant, and let $A_{\rm opt}$ be the corresponding optimal family. Then, every bounded set defined as a whole by this family $A_{\rm opt}$ is an ellipse.*

Thus, we should use ellipses to approximate the insect shapes.

3.3 Formalization of the corresponding data mining problem. 2: The use of aspect ratio

According to our result, we must approximate the insect's shape by an ellipse. Since insects can destroy the crop, we better err on the side of caution, and use an ellipse which contains the actual insect shape S. To classify an insect, we should therefore use a characteristic of this approximating ellipse. What characteristic should we choose?

The type of an insect does not change if we simply shift or rotate the insect; thus, the characteristics used to classify the insect should not change if we simply shift or rotate the insect's image (and hence, shift or rotate the corresponding ellipse).

Similarly, the classification of an insect should not change if the insect simply *grows*. This growth can be described as *scaling* $(x,y)\to(\lambda\cdot x,\lambda\cdot y)$, so our characteristic should not change with scaling.

So, we want a characteristic of an ellipse which does not change with shift, rotation, or scaling.

Definition 5.

- *By a characteristic of an ellipse, we mean a function $J:\mathcal{E}\to R$ from the set $\mathcal{E}$ of all ellipses to the set R of real numbers.*
- *We say that a characteristic J is shift- (corr., rotation- and scale-invariant) if for every ellipse E, $J(E)=J(T(E))$, where $T(E)$ denotes a shift (rotation, scaling) of the ellipse E.*
- *An aspect ratio $a(E)$ is a ratio $D_{\max}/D_{\min}$ of the lengths of the major and minor axes of an ellipse.*

It is easy to check that the aspect ratio is a shift-, rotation-, and scale-invariant characteristic of an ellipse. It turns out that it is, in effect, the only such characteristic:

Proposition 4. *Let J be a characteristic of an ellipse which is shift-, rotation- and scale-invariant. Then there exists a function $f : R \to R$ for which $J(E) = f(a(E))$ for every ellipse E.*

Thus, if we know the aspect ratio, we can compute an arbitrary invariant characteristic of an ellipse. So, to classify an insect, we should use the aspect ratio of the approximating ellipse.

3.4 Formalization of the corresponding data mining problem. 3: Selected values of aspect ratio

3.4.1 Theoretical result

In the previous text, we took into consideration the fact that an insect grows; we represented this growth by a scaling transformation, i.e., a transformation which simply "blows up" the insect without changing its shape. In reality, the life of an insect can be divided into several stages. On each stage, the growth can be reasonably well described as scaling; however, the transition from one stage to another changes the shape. The harmlessness of an insect may drastically change from stage to stage, so, it is important not only to classify insects on a cotton field, but also to find out on what stage these insects are.

The transition from one stage to another can be described as a transformation $\vec{x} \to \vec{f}(\vec{x})$ which transforms the location $\vec{x} = (x, y)$ of the original point on a body (e.g., eye, leg, etc.), into its position $\vec{f}(\vec{x})$ on the next stage. In general, this transformation $\vec{f}(\vec{x})$ can be non-linear. To simplify this general expression, let us expand this function into Taylor series. Insects are small, so the coordinates $\vec{x}$ are small, hence quadratic terms in this expansion are also relatively small in comparison with the linear terms. Since we are approximating the insect's shape by an ellipse anyway (thus drastically distorting its shape), there is no big sense in keeping these small quadratic terms. Therefore, it makes sense to assume that the transformation $\vec{x} \to \vec{f}(\vec{x})$ from one stage to another is linear.

A generic linear transformation can be described as a rotation and shift followed by contractions and dilatations along appropriately chosen (orthogonal) coordinates (see, e.g., Chapter 11, p. 49 from [1] or Theorem 5.42 from [16]). Since shifts and rotations do not change the shape, we can therefore assume, without losing generality, that the change in shape from one stage to another can be described by contractions and dilatations along the axes, i.e., by a transformation $x \to x' = \lambda_x \cdot x$, $y \to y' = \lambda_y \cdot y$ for some real numbers $\lambda_x > \lambda_y > 0$.

An insect starts as an extremely small point-size embryo; we can assume that the embryo is a small circle, i.e., an ellipse with an aspect ratio equal to 1, in which the minor and the major axes have the same length: $D_{\max} = D_{\min}$. To find the shape at the next stage, we must apply the above transformation. As a result, we get a new ellipse, with an aspect ratio $r = \lambda_x / \lambda_y$. One more application of the above transformation leads to the new ellipse with an aspect ratio r^2, then r^3, etc.

We can therefore conclude the aspect ratios of the ellipses approximating to the actual insect shapes form a geometric progression $1, r, r^2, r^3, \ldots$

3.4.2 Experimental testing of the theoretical result

To test this theoretical conclusion, we took the average aspect ratios r_i of the insects inhabiting cotton and alfalfa fields; if the above conclusion is correct, then each aspect ratio r_i has the form r^k for some integer k and therefore $\ln(r_i) = k \cdot \ln(r)$. Indeed, the experimental data shows that all the values $\ln(r_i)$ are approximately proportional to whole multiples of some real number which can be therefore taken as $\ln(r)$. As a result, we got $r \approx 1.2$. The aspect ratios of different insects can be approximately described as q^k for integer values k:

- $k = 2$, aspect ratio $r^2 \approx 1.4$:
  - Stingbug Adult (harmless) and
  - Hippodamia Lady Beetle Adult (harmless);

- $k = 3$, aspect ratio $r^3 \approx 1.7$:
  - Three-Corned Alfalfa Hopper (destructive);
  - Cucumber Beetle (destructive);
  - Collops Beetle (harmless);
- $k = 4$, aspect ratio $r^4 \approx 2.0$:
  - Big-Eyed Bug (harmless) and
  - Hippodamia Ladybug Larva (harmless);
- $k = 5$, aspect ratio $r^5 \approx 2.4$:
  - Assassin Bug (harmful to humans);
  - Lygus Adult (destructive);
  - Lacewing Larva (harmless);
- $k = 6$, aspect ratio $r^6 \approx 2.9$:
  - Nabid Adult (harmless);
  - Leaf Hopper (can be destructive);
- $k = 8$, aspect ratio $r^8 \approx 4$:
  - Grace Lace-Wing Adult (harmless).

Totally, we have six different values of k; for three of these values ($k = 2$, $k = 5$, and $k = 8$), we get a conclusion that the corresponding insects are harmless. For three other values, to distinguish between harmless and harmful insects, we must consider other geometric characteristics.

3.4.3 Additional geometric characteristic

A natural idea is to use characteristics similar to the ones used to classify trash. The simplest geometric characteristic is the area A of the actual image; however, the area itself is not a good characteristic for insect classification because it increases when the insect grows: when an insect grows as $\vec{x} \to \lambda \cdot \vec{x}$, the area increases by λ^2. Thus, the area itself characterizes not only the type of image but also the actual size of the insect of this particular type. Since we cannot use the area A directly, we must use area to define a new characteristic which is growth-invariant (i.e., scale-invariant) and thus, changes only from one specie to another, but not within the same specie.

Our whole idea is to approximate the shape with an ellipse, find the lengths $D_{\max}$ and $D_{\min}$ of the major and minor axes of the approximating ellipse, and then compute the ratio. Thus, for each image, we know the lengths $D_{\max}$ and $D_{\min}$. As an insect grows $\vec{x} \to \lambda \cdot \vec{x}$, these lengths increase as $D_{\max} \to \lambda \cdot D_{\max}$ and $D_{\min} \to \lambda \cdot D_{\min}$. Thus, the ratio $A/D^2_{\max}$ is scale-invariant.

This ratio has a direct geometric sense: indeed, for a given $D_{\max}$, this ratio attains the largest possible value when A is the largest possible, i.e., when:

- the area of the approximating ellipse is the largest possible for a given $D_{\max}$ and
- the image occupies the largest possible part of this approximating ellipse.

The second condition simply means that the image actually coincides with the approximating ellipse. For a fixed length $D_{\max}$ of the major axis, the area of the ellipse increases with $D_{\min}$, so this area is the largest when $D_{\min}$ attains its largest possible value $D_{\max}$, i.e., when this elliptical image is actually a round circle. Thus, the above ratio attains the largest possible value when the image is round; in view of this property, this ratio is called *roundness factor*.

For a circle, the area is equal to $A = \pi \cdot D^2/4$, and so the above roundness factor is equal to $\pi/4$. For manual analysis, it is convenient to "re-scale" the roundness factor in such a way that this maximum be equal to 1, in other words, it is convenient to consider the ratio $RF = (4A)/(\pi \cdot D^2_{\max})$.

3.4.4 Experimental use of roundness factor

This characteristic enables us to distinguish between several harmful and harmless insects:

- For $k = 5$,
  - harmful Assassin Bug has $RF \approx 0.25$, while
  - destructive Lygus Adult and harmless Lacewing Larva both have $RF \approx 0.36$.
- For $k = 6$, we get a full distinction:
  - for harmless Habid Adult, $RF \approx 0.32$; while
  - for possibly destructive Leaf Hopper, $RF \approx 0.28$.

In both cases, we have a similar geometric phenomenon:

- harmless insects are more round (have larger values of the roundness factor), while
- harmful insects are less round (have smaller values of the roundness factor).

For $k = 3$, all three species have approximately the same value of roundness factor $RF \approx 0.5$; so, to distinguish between them, we need an additional characteristic. It turns out that for these insects, the size (characterized, e.g., by the area A itself) can distinguish between harmless and harmful insects:

- harmless insects (Collops Beetle) are typically smaller ($A \approx 1,000$ pixels), while
- harmful insects are usually larger: for Three-Corned Alfalfa Hopper, $A \approx 2,000$, and for Cucumber Beetle, $A \approx 3,000$.

3.5 Practical results

By using aspect ratio, roundness factor, and area, we get an almost perfect insect classification; the only exception is that it is difficult to distinguish between destructive Lygus Adult and harmless Lacewing Larva.

While describing the three characteristics, we showed natural rules which use the values of these characteristics to determine the insect type. These rules, however, are difficult to implement directly because they use words from natural language like "small", "large", etc. We therefore need fuzzy logic to formalize these rules. In this formalization, we used the Adaptive Network-Based Fuzzy Inference System ANFIS (similar to trash classification).

The resulting systems achieves an almost 100% correct classification of insects – a much higher percentage than the previously known methods; for further details see, e.g., [3, 4].

4 Third Case Study: Automated System for Tracking High-Speed Targets

4.1 A practical problem: brief description

In air traffic control and in military applications, it is necessary to track high-speed targets such as airplanes, missiles, etc.

Typically, radar images are used for this tracking, but radar provides a very crude picture of the targets: typically, a distant target is seen as a blurred point; from this image, it is difficult to tell whether it is a plane at all and what type of plane it is. This low resolution is caused by the physical fact that the angular size of the smallest visible detail is $\approx \lambda/D$, where λ is the wavelength and D is the antenna's diameter; a radar operates on radiowaves, for which λ is reasonably large. To improve the quality of the target pictures, it is therefore necessary to switch to smaller wavelengths, e.g., to visible light.

Thus, to get a good quality picture of the target, we must use a rotating telescope. Since we are interested in fast-moving targets, the tracking telescope must rotate fast, so fast that a human operator is unable to control it. We therefore need an automatic control of the tracking system.

4.2 The need for fuzzy control

The corresponding control problem is highly non-linear, so traditional control methods – which work well for linear or almost linear systems – cannot be efficiently applied. Therefore, we need non-linear control techniques.

Typically, in non-linear control, we select a general expression for a non-linear controller, and then tune this expression in such a way as to get an efficient controller. There are many different general non-linear expression which can be used (and which have been actually used) in this process.

Most of these expressions come from the mathematical analysis of this problem: e.g., from the known fact that an arbitrary function can be approximated within any given accuracy by a polynomial (e.g., for an analytical function, we can take the sum of the first several terms in the Taylor series expansion as the desired polynomial approximation). The main disadvantage of these expressions is that they are very formal, they lack an intuitive understanding which is extremely important in real-life control problems.

There is a class of non-linear expressions, however, which does not have this drawback: namely, the class of non-linear expressions which come from *fuzzy control* (see, e.g., [9]). The corresponding expressions are universal approximators in the sense that an arbitrary non-linear control function can be approximated, with any given accuracy, an input-output function of an appropriate fuzzy controller. However, no matter how complex is the resulting input-output function, the system can still be described by a system of understandable if-then rules formulated in terms of words from natural language. This understandability and transparency of a non-linear controller is, as we have mentioned, an extremely important advantage of fuzzy control. With this advantage in mind, in our research, we used fuzzy control for tracking high-speed targets.

4.3 The need for data mining

Most applications of fuzzy control deal with near-linear systems, in which we can achieve a reasonably good control by using a near-linear controller. The most practically used linear controller is a PID controller, in which, in order to maintain a certain trajectory $x_0(t)$ of the controlled system (called *plant*), the system uses, at each given moment of time t, a control value $u(t)$ which is equal to the linear combination of three quantities:

- the state error $P(t) = x(t) - x_0(t)$, i.e., the difference between the actual state $x(t)$ and the desired state $x_0(t)$;
- the *integral* $I(t) = \int^t P(s)\,\mathrm{d}s$ of the state error; and
- the time derivative $D(t) = \dot{P}(t)$ of the state error.

In other words, in PID control, to determine the control value $u(t)$, we use three inputs $P(t)$, $I(t)$, and $D(t)$, and form a linear combination to produce the desired control value: $u(t) = K_P \cdot P(t) + K_I \cdot I(t) + K_D \cdot D(t)$.

Naturally, for near-linear systems, the typically used fuzzy control uses rules with the same three inputs. If we use three membership functions (e.g., "small", "medium", and "large") to describe each of the three inputs, then we have to have $3 \times 3 \times 3 = 27$ rules to describe the control values for all 27 possible combinations of the input values.

For a strongly non-linear system, PID control, in which the gains K_i are the same for all pieces of the plant's trajectory, is no longer efficient: if we choose the gains K_i which lead to reasonable control on one part of the trajectory, these gains often lead to a low quality control in different parts of it. It is therefore necessary to make the control $u(t)$ depending not only on the difference $P(t)$ and on its integral $I(t)$ and derivative $D(t)$, but also on the actual values of the trajectory, i.e., on the current position $x(t)$, the current velocity $\dot{x}(t)$, etc. Therefore, in formulating fuzzy rules for such strongly non-linear systems, it is reasonable, in addition to the original three inputs $P(t)$, $I(t)$, and $D(t)$, to provide extra inputs such as $x(t)$, $\dot{x}(t)$, etc.

Even if use only 3 fuzzy values for each of the inputs, and even if we use only 5 inputs, we already need a huge number of $3^5 = 243$ rules. For each rules, we need to tune at least one parameter, so we need at least 243 parameters. It is extremely difficult to tune in a rule base with that many parameters. To decrease the number of rules, we use the *data mining* technique: namely, instead of simply using all input variables, we find a smaller number of relevant *combinations* $y_1, \ldots, y_m$ of these input variables, and then use these combinations y_j as the new inputs. So, to be able to use soft computing techniques (namely, fuzzy control techniques) for automated tracking of high-speed targets, we must first use *data mining* to find relevant combinations of input variables.

4.4 How this data mining was done

We have already mentioned one relationship between soft computing and data mining: that in order to use the expert-formalizing fuzzy control methodology, we must use data mining. Let us show that this relationship works the other way around too: the expert's knowledge (naturally formulated in fuzzy terms) can help in solving the corresponding data mining problem.

4.4.1 Handling the first non-linearity

The first reason why the control problem is non-linear is that the control is different for different elevations of the telescope; e.g.:

- when the system is pointing straight up (i.e., its elevation is 90°), it is very easy to rotate it;
- when the system is horizontal, its moment of inertia is much larger, and it is more difficult to rotate.

For a fixed elevation, we can reasonably expect that a linear controller, with

$$u(t) = K_P \cdot P(t) + K_I \cdot I(t) + K_D \cdot D(t) + K_x \cdot x(t) + K_v \cdot \dot{x}(t),$$

will work well. For different elevation angles, we can tune this linear controller, and find the appropriate values of gains. The resulting control can now be described by the following three fuzzy rules:

- for low elevations, use the gains K_i corresponding to elevation angle 0;
- for medium elevations, use the gains K_i corresponding to elevation angle 45°;
- for high elevations, use the gains K_i corresponding to elevation angle 90°.

As a result of applying fuzzy control methodology to these rules, we get, for different values of the evelation angle, different values of the gains K_i.

4.4.2 Handling the second non-linearity

The above idea takes care of the first non-linearity. Dependence on the elevation, however, is not the only reason why this control problem is non-linear. Another reason is related to the following fundamental problem of linearly controlled plants:

- Our goal is tracking. Therefore, when we slightly deviate from the correct trajectory, we want the system to get back to it as fast as possible. So, for small values of the deviation $P(t)$, the gain coefficient K_P corresponding to $P(t)$ should be large.
- On the other hand, when the deviation becomes large, the use of the large coefficient K_P will lead to an extremely strong force applied to the telescope, and, as a result, the telescope will overcompensate and get into wild oscillations before getting back. So, for large value of the deviation $P(t)$, we must use the control corresponding to the smaller coefficient K_P.

In other words, for an ideal control, instead of the term $K_P \cdot P(t)$ with a constant gain, we should have the term $K_P(P(t)) \cdot P(t)$, in which the gain K_P changes with the absolute value of $P(t)$. Alternatively, we can describe this desired control by assuming that the gain is fixed (e.g., as the gain $K_P(0)$ which is optimal for small deviations), but this gain is multiplied not by the deviation $P(t)$ itself, but by a "re-scaled" deviation $P_{\text{new}}(t) = c(P(t)) \cdot P(t)$, where $c(P) = K_P(P)/K_P(0)$.

For each fixed value of P, we can fine-tune the resulting control and find the optimal value of $K_P(P)$ and thus, of $c(P)$. We can thus get the values $c(P)$ for several different values of P, and then use fuzzy control methodology to interpolation from these values to arbitrary values of P. Namely, it turns out that we can use the following rules:

- if the angle deviation P is negligible, then $c(P) = 1$;
- if the angle deviation P is small, then $c(P) = 0.7$;
- if the angle deviation P is medium, then $c(P) = 0.4$;
- if the angle deviation P is large, then $c(P) = 0.2$.

As a result of applying fuzzy control technique to these rules, we get a continuous function $c(P)$.

4.4.3 Handling the third non-linearity

The third reason for non-linearity is the saturation of control. This non-linearity is taken care of by simply thresholding the corresponding inputs, so that each value above the threshold is replaced by this threshold.

4.5 Practical results

The resulting hierarchical fuzzy control system is indeed very effective in tracking; see, e.g., [13, 14].

Conclusion

In many applications of soft computing, the number of input variables is so large that traditional soft computring techniques are no longer practically applicable. This happens, e.g., in image processing, when the number of numerical inputs (i.e., the number of pixels in an image) can be astronomic; this also happens in processing highly dynamical data, when the number of numerical inputs (i.e., the number of moments of time) can also be large.

In such situations, instead of simply using all input variables, we must find a smaller number of relevant *combinations* $y_1, \ldots, y_m$ of these input variables, and then use these combinations y_j as the new inputs. In other words, in such situations, to be able to use soft computing techniques, we must first use *data mining* to find relevant combinations of input variables.

In this paper, we showed how the resulting combination of data mining and soft computing techniques can help to solve practical problems in cotton industry and in tracking high-speed targets.

Acknowledgments

This work was supported in part by NASA under cooperative agreement NCC5-209, by NSF grant No. DUE-9750858, by United Space Alliance, grant No. NAS 9-20000 (PWO C0C67713A6), by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-95-1-0518, and by the National Security Agency under Grant No. MDA904-98-1-0561.

References

[1] A. D. Aleksandrov, A. N. Kolmogorov, and M. A. Lavrentiev, *Mathematics, its content, methods, and meaning*, Vol. 2, American Mathematical Society, Providence, R.I., 1963.

[2] L. Fausett, *Fundamentals of Neural Networks*, Englewood Cliffs, New Jersey: Prentice-Hall, 1994.

[3] H. Gassoumi, J. J. Ellington, H. T. Nguyen, and N. R. Prasad, "A soft computing approach to insects classification in the cotton field", *Proceedings of the International Symposium on Medical Informatics and Fuzzy Technology MIF'99*, Hanoi, Vietnam, August 27–29, 1999, pp. 454–485.

[4] H. Gassoumi, J. J. Ellington, H. T. Nguyen, and N. R. Prasad, "Integrated pest management system", In: H. Mohanty and C. Baral (eds.), *Trends in Information Technology, Proceedings of the International Conference on Information Technology ICIT'99, Bhubaneswar, India, December 20–22, 1999*, Tata McGraw-Hill, New Delhi, 2000, pp. 126–131.

[5] V. Kreinovich and J. Wolff von Gudenberg, "An optimality criterion for arithmetic of complex sets", *Geombinatorics*, 2000 (to appear).

[6] M. A. Lieberman and R. B. Patil, "Clustering and neural networks to categorize cotton trash", *Optical Engineering*, 1994, Vol. 33, No. 5, pp. 1642–1653.

[7] M. A. Lieberman and R. B. Patil, "Evaluation of learning vector quantization to classify cotton trash", *Optical Engineering*, 1997, Vol. 36, No. 3, pp. 914–921.

[8] H. T. Nguyen and V. Kreinovich, *Applications of continuous mathematics to computer science*, Kluwer, Dordrecht, 1997.

[9] H. T. Nguyen and M. Sugeno (eds.), *Fuzzy Systems: Modeling and Control*, Kluwer, Boston, MA, 1998.

[10] J. C. Russ, *The image processing handbook*, Boca Raton, FL, CRC Press, 1994.

[11] M. Siddaiah, M. A. Lieberman, S. E. Hughs, and N. R. Prasad, "A soft computing approach to classification of trash in ginned cotton", *Proceedings of the 8th International Fuzzy Systems Association World Congress IFSA'99*, Taipei, Taiwan, August 17–20, 1999, pp. 151–155.

[12] M. Siddaiah, M. A. Lieberman, S. E. Hughs, and N. R. Prasad, "Identification of trash types in ginned cotton using neuro fuzzy techniques", *Proceedings of the 8th IEEE International Conference on Fuzzy Systems FUZZ-IEEE'99*, Seoul, Korea, August 22–25, 1999, Vol. 2, pp. 738–743.

[13] J. Stufflebaum, *An integrated systems approach to tagret tracking using hierarchical fuzzy control*, Ph.D. Diseertation, New Mexico State University, 1999.

[14] J. Stufflebaum and N. R. Prasad, "Hierarchical fuzzy control", *Proceedings of the 8th IEEE International Conference on Fuzzy Systems FUZZ-IEEE'99*, Seoul, Korea, August 22–25, 1999, Vol. 1, pp. 498–503.

[15] J. Wolff von Gudenberg and V. Kreinovich, "Candidate Sets for Complex Interval Arithmetic", In: H. Mohanty and C. Baral (eds.), *Trends in Information Technology, Proceedings of the International Conference on Information Technology ICIT'99, Bhubaneswar, India, December 20–22, 1999*, Tata McGraw-Hill, New Delhi, 2000, pp. 230–233.

[16] P. B. Yale, *Geometry and symmetry*, Dover, New York, 1988.

Appendix: Proofs

Proof of Proposition 1

This proof is similar to the proofs used in [8] to justify different heuristic methods in soft computing (fuzzy, neural, etc.).

1. Let us first show that the optimal family $A_{\rm opt}$ is itself shift-, rotation-, and scale-invariant.

Indeed, let T be an arbitrary shift, rotation, or scaling. Since $A_{\rm opt}$ is optimal, for every other family B, we have $A_{\rm opt} \succeq T^{-1}B$ (where T^{-1} means the inverse transformation). Since the optimality criterion $\succeq$ is invariant, we conclude that $TA_{\rm opt} \succeq T(T^{-1}B) = B$. Since this is true for every family B, the family $TA_{\rm opt}$ is also optimal. But since our criterion is final, there is only one optimal family and therefore, $TA_{\rm opt} = A_{\rm opt}$. In other words, the optimal family is indeed invariant.

2. Let us now show that all functions from $A_{\rm opt}$ are polynomials.

Indeed, every function $F \in A_{\rm opt}$ is analytical, i.e., can be represented as a Taylor series (sum of monomials). Let us combine together monomials $c \cdot x^a \cdot y^b$ of the same degree $a + b$; then we get

$$F(z) = F_0(z) + F_1(z) + \ldots + F_k(z) + \ldots,$$

where $F_k(z)$ is the sum of all monomials of degree k. Let us show, by induction over k, that for every k, the function $F_k(z)$ also belongs to $A_{\rm opt}$.

Let us first prove that $F_0(z) \in A_{\rm opt}$. Since the family $A_{\rm opt}$ is scale-invariant, we conclude that for every $\lambda > 0$, the function $F(\lambda \cdot z)$ also belongs to $A_{\rm opt}$. For each term $F_k(z)$, we have $F_k(\lambda \cdot z) = \lambda^k \cdot F_k(z)$, so

$$F(\lambda \cdot z) = F_0(z) + \lambda \cdot F_1(z) + \ldots \in A_{\rm opt}.$$

When $\lambda \to 0$, we get $F(\lambda \cdot z) \to F_0(z)$. The family $A_{\rm opt}$ is finite-dimensional hence closed; so, the limit $F_0(z)$ also belongs to $A_{\rm opt}$. The induction base is proven.

Let us now suppose that we have already proven that for all $k < s$, $F_k(z) \in A_{\rm opt}$. Let us prove that $F_s(z) \in A_{\rm opt}$. For that, let us take

$$G(z) = F(z) - F_1(z) - \ldots - F_{s-1}(z).$$

We already know that $F_1, \ldots, F_{s-1} \in A_{\rm opt}$; so, since $A_{\rm opt}$ is a linear space, we conclude that

$$G(z) = F_s(z) + F_{s+1}(z) + \ldots \in A_{\rm opt}.$$

The family $A_{\rm opt}$ is scale-invariant, so, for every $\lambda > 0$, the function

$$G(\lambda \cdot z) = \lambda^s \cdot F_s(z) + \lambda^{s+1} \cdot F_{s+1}(z) + \ldots$$

also belongs to $A_{\rm opt}$. Since $A_{\rm opt}$ is a linear space, the function

$$H_\lambda(z) = \lambda^{-s} \cdot G(\lambda \cdot z) = F_s(z) + \lambda \cdot F_{s+1}(z) + \lambda^2 \cdot F_{s+2}(z) + \ldots$$

also belongs to $A_{\rm opt}$.

When $\lambda \to 0$, we get $H_\lambda(z) \to F_s(z)$. The family $A_{\rm opt}$ is finite-dimensional hence closed; so, the limit $F_s(z)$ also belongs to $A_{\rm opt}$. The induction is proven.

Now, monomials of different degree are linearly independent; therefore, if we have infinitely many non-zero terms $F_k(z)$, we would have infinitely many linearly independent functions in a finite-dimensional family $A_{\rm opt}$ – a contradiction. Thus, only finitely many monomials $F_k(z)$ are different from 0, and so, $F(z)$ is a sum of finitely many monomials, i.e., a polynomial.

3. Let us prove that if a function $F(x,y)$ belongs to $A_{\rm opt}$, then its partial derivatives $F_{,x}(x,y)$ and $F_{,y}(x,y)$ also belong to $A_{\rm opt}$.

Indeed, since the family $A_{\rm opt}$ is shift-invariant, for every $h > 0$, we get $F(x+h,y) \in A_{\rm opt}$. Since this family is a linear space, we conclude that a linear combination $h^{-1}(F(x+h,y) - F(x,y))$ of two functions from $A_{\rm opt}$ also belongs to $A_{\rm opt}$. Since the family $A_{\rm opt}$ is finite-dimensional, it is closed and therefore, the limit $F_{,x}(x,y)$ of such linear combinations also belongs to $A_{\rm opt}$. (For $F_{,y}$, the proof is similar).

4. Due to Parts 2 and 3 of this proof, if any polynomial from $A_{\rm opt}$ has a non-zero part F_k of degree $k > 0$, then it also has a non-zero part ($(F_k)_{,x}$ or $(F_k)_{,y}$) of degree $k-1$. Similarly, it has non-zero parts of degrees $k-2, \ldots, 1, 0$.

So, in all cases, $A_{\rm opt}$ contains a non-zero constant and a non-zero linear function $F_1(x,y) = b \cdot x + c \cdot y$. We can now use the fact that the family $A_{\rm opt}$ is rotation-invariant; let T be a rotation which transforms (b,c) into the x-axis, then we conclude that

$$F_1(Tz) = b' \cdot x \in A_{\rm opt},$$

and hence $x \in A_{\rm opt}$. Similarly, $y \in A_{\rm opt}$. So, the family $A_{\rm opt}$ contains at least 3 linearly independent functions: a non-zero constant, x, and y.

If $d = 3$, then the 3-D family $A_{\rm opt}$ cannot contain anything else, and all the pieces of borders $F(x,y) = 0$ of all the sets defined by this family are straight lines.

If $d = 4$, then we cannot have any cubic or higher order terms in $A_{\rm opt}$, because then, due to Part 3, we would have both this cubic part *and* a (linearly independent) quadratic part, and the total dimension of $A_{\rm opt}$ would be at least $3 + 2 = 5$. So, all functions from $A_{\rm opt}$ are quadratic. Since $\dim(A_{\rm opt}) = 4$, and the dimension of 0- and 1-D parts is 3, the dimension of possible parts of second degree is 1. Since $A_{\rm opt}$ is rotation-invariant, the quadratic part $d \cdot x^2 + e \cdot x \cdot y + f \cdot y^2$ must be also rotation-invariant (else, we would have two linearly independent quadratic terms in $A_{\rm opt}$: the original expression and its rotated version). Thus, this quadratic part must be proportional to $x^2 + y^2$.

Hence, every function $F \in A_{\rm opt}$ has the form

$$F(x,y) = a + b \cdot x + c \cdot y + d \cdot (x^2 + y^2),$$

and therefore, all the pieces of borders $F(x,y) = 0$ of all the sets defined by this family are either straight lines or circular arcs. Proposition 1 is proven.

Proof of Proposition 2

Similarly to Part 1 of the proof of Proposition 1, we can show that the optimal angle is swap-invariant, i.e., $\alpha_{\rm opt} = T(\alpha_{\rm opt})$. Therefore, $\alpha_{\rm opt} = 90 - \alpha_{\rm opt}$, hence $2\alpha_{\rm opt} = 90$, and $\alpha_{\rm opt} = 45$. The proposition is proven.

Proof of Proposition 3

While proving Proposition 1, we have already shown the following:

- all the functions F from the optimal family $A_{\rm opt}$ are polynomials;
- if a function $F(x,y)$ belongs to $A_{\rm opt}$, then its partial derivatives $F_{,x}(x,y)$ and $F_{,y}(x,y)$ also belong to $A_{\rm opt}$; and
- the optimal family $A_{\rm opt}$ contains at least 3 linearly independent functions: a non-zero constant, x, and y.

Let us now show that since $d \le 6$, we cannot have any quartic (or higher order) terms in $A_{\rm opt}$.

Indeed, in this case, due to Part 3 of Proposition 1, in addition to 3-dimensional linear part, $A_{\rm opt}$ would contain this quartic part, a (linearly independent) cubic part, *and* a (linearly independent) quadratic part, and the total dimension d of $A_{\rm opt}$ would be at least $d = 3 + d_2 + d_3 + d_4 + \ldots$, where d_2 is the dimension of the quadratic part, d_3 is the dimension of the cubic part, etc. We have $d_2 \ge 1$, $d_3 \ge 1$, and $d_4 \ge 1$, so, if we had $d_3 > 1$, we would have

$$d = 3 + d_2 + d_3 + d_4 + \ldots > 3 + 1 + 1 + 1 = 6.$$

Since we assumed that $d \le 6$, this is impossible and thus, $d_3 \le 1$, i.e., $d_3 = 1$. Since $A_{\rm opt}$ is rotation-invariant, the cubic part $a \cdot x^3 + b \cdot x^2 \cdot y + c \cdot x \cdot y^2 + d \cdot y^3$ must be also rotation-invariant (else, we would have two linearly independent cubic terms in $A_{\rm opt}$: the original expression and its rotated version). However, it is known that there are no rotation-invariant cubic terms (actually, every rotation-invariant polynomial is a polynomial in $x^2 + y^2$, and is, therefore, of even order). Thus, quartic terms are indeed impossible.

Since quartic and higher order terms are impossible, every polynomial $F \in A_{\rm opt}$ is either cubic or quadratic. Let us prove that for a cubic polynomial

$$F(x,y) = F_0(x,y) + F_1(x,y) + F_2(x,y) + F_3(x,y)$$

with a non-degenerate cubic part $F_3(x,y)$, the equation $F(x,y) = 0$ does not form a boundary of any bounded set at all.

Indeed, since $F_3 \ne 0$, there exists a point $z = (x,y)$ for which $F_3(x,y) \ne 0$. Without losing generality, we can assume that $F_3(z) > 0$. Let us take a new point $N \cdot z = (N \cdot x, N \cdot y)$, where N is a positive integer. For this new point, we have

$$F(N \cdot z) = F_0(z) + N \cdot F_1(z) + N^2 \cdot F_2(z) + N^3 \cdot F_3(z)$$

and hence,

$$\frac{F(N \cdot z)}{N^3} = N^{-3} \cdot F_0(z) + N^{-2} \cdot F_1(z) + N^{-1} \cdot F_2(z) + F_3(z).$$

When $N \to \infty$, we have $F(N \cdot z)/N^3 \to F_3(z) > 0$ and therefore, for all sufficiently large N, we have $F(N \cdot z)/N^3 > 0$ and thence, $F(N \cdot z) > 0$.

Similarly, we have

$$F(-N \cdot z) = F_0(z) - N \cdot F_1(z) + N^2 \cdot F_2(z) - N^3 \cdot F_3(z);$$

hence,

$$\frac{F(-N \cdot z)}{N^3} = N^{-3} \cdot F_0(z) - N^{-2} \cdot F_1(z) + N^{-1} \cdot F_2(z) - F_3(z).$$

When $N \to \infty$, we have $F(-N \cdot z)/N^3 \to -F_3(z) < 0$ and therefore, for all sufficiently large N, we have $F(-N \cdot z)/N^3 < 0$ and thence, $F(-N \cdot z) < 0$.

Both points $N \cdot z$ and $-N \cdot z$ belong to the same circle with a center in 0 and raduis $N \cdot \|z\|$ (where $\|z\| = \sqrt{x^2 + y^2}$). Thus, on this circle, there are two points for which the function $F(z)$ take values of different signs. Since this function $F(z)$ is continuous, it attains a 0 value somewhere on this circle. Thus, for arbitrarily large N, a circle of radius $N \cdot \|z\|$ contains a point z' for which $F(z') = 0$. Hence, the set of all the point for which $F(x, y) = 0$ is not bounded and therefore, cannot form a boundary of a bounded set.

Thus, if a bounded set defined as a whole by the optimal family $A_{\rm opt}$, then the corresponding function $F(x, y)$ cannot be cubic and, therefore, it has to be quadratic. The only bounded set bounded by a set $F(x, y) = 0$ for a quadratic function F is an ellipse. The proposition is proven.

Proof of Proposition 4

Let J be an invariant characteristic of an ellipse. It is well known that we can shift an arbitrary ellipse E so that its center coincides with the origin $(0, 0)$ of the coordinate system, and then rotate it in such a way that the major axis of the ellipse will lie on the coordinate axis $0x$, and its minor axis on the coordinate line $0y$. As a result, we get a new ellipse E_1 which is obtained from the original ellipse E by a combination T of shift and rotation: $E_1 = T(E)$. Since the characteristic J is invariant, shift and rotation do not change its value, so $J(E_1) = J(E)$. Shift and rotation preserve the axes of the ellipse, so for the new ellipse E_1, the lengths $D_{\max}$ and $D_{\min}$ of the ellipse's axes are the same as for the original ellipse E.

We can now scale E_1 by applying a scaling $\vec{x} \to \vec{x}/D_{\min}$. After this scaling, we get a new ellipse E_2 which is (similarly to E_1) aligned with the coordinate axes; the length of the axes of the new ellipse E_2 are equal to $D_{\max}/D_{\min}$ and 1. Since the characteristic J is scale-invariant, we have $J(E_2) = J(E_1)$; since we already know that $J(E_1) = J(E)$, we conclude that $J(E_2) = J(E)$.

For the ellipse E_2, we know its orientation, and we know the lengths of its minor axis (1) and of its major axis ($D_{\max}/D_{\min}$). This information uniquely determines the ellipse; therefore, if we know the aspect ratio $D_{\max}/D_{\min}$, we can uniquely determine the ellipse E_2 and hence, the value $J(E_2) = J(E)$. Thus, the value $J(E)$ indeed depends only on the aspect ratio. The proposition is proven.