Data Complexity in Machine Learning

Ling Li and Yaser S. Abu-Mostafa

Learning Systems Group, California Institute of Technology

Abstract. We investigate the role of data complexity in the context of binary classification problems. The universal data complexity is defined for a data set as the Kolmogorov complexity of the mapping enforced by the data set. It is closely related to several existing principles used in machine learning such as Occam's razor, the minimum description length, and the Bayesian approach. The data complexity can also be defined based on a learning model, which is more realistic for applications. We demonstrate the application of the data complexity in two learning problems, data decomposition and data pruning. In data decomposition, we illustrate that a data set is best approximated by its principal subsets which are Pareto optimal with respect to the complexity and the set size. In data pruning, we show that outliers usually have high complexity contributions, and propose methods for estimating the complexity contribution. Since in practice we have to approximate the ideal data complexity measures, we also discuss the impact of such approximations.

1 Introduction

Machine learning is about pattern[1] extraction. A typical example is an image classifier that automatically tells the existence of some specific object category, say cars, in an image. The classifier would be constructed based on a training set of labeled image examples. It is relatively easy for computers to "memorize" all the examples, but in order for the classifier to also be able to correctly label images that have not been seen so far, meaningful patterns about images in general and the object category in particular should be learned. The problem is "what kind of patterns should be extracted?"

Occam's razor states that entities should not be multiplied beyond necessity. In other words, if presented with multiple hypotheses that have indifferent predictions on the training set, one should select the simplest hypothesis. This preference for simpler hypotheses is actually incorporated, explicitly or implicitly, in many machine learning systems (see for example Quinlan, 1986; Rissanen, 1978; Vapnik, 1999). Blumer et al. (1987) showed theoretically that, under very general assumptions, Occam's razor produces hypotheses that can correctly predict unseen examples with high probability. Although experimental evidence was found against the utility of Occam's razor (Webb, 1996), it is still generally believed that the bias towards simpler hypotheses is justified for real-world problems (Schmidhuber, 1997). Following this line, one should look for patterns that are consistent with the examples, and simple.

But what exactly does "simple" mean? The Kolmogorov complexity (Li and Vitányi, 1997) provides a universal measure for the "simplicity" or complexity of patterns. It says that a pattern is simple if it can be generated by a short program or if it can be compressed, which essentially means that the pattern has some "regularity" in it. The Kolmogorov complexity is also closely related to the so-called universal probability distribution (Li and Vitányi, 1997; Solomonoff, 2003), which is able to approximate any computable distributions. The universal distribution assigns high probabilities to simple patterns, and thus implicitly prefers simple hypotheses.

While most research efforts integrating Occam's razor in machine learning systems have been focused on the simplicity of hypotheses, the other equivalently important part in learning systems, training sets, has received much less attention in the complexity aspect, probably because training

[1] In a very general sense, the word "pattern" here means hypothesis, rule, or structure.

Caltech Computer Science Technical Report CaltechCSTR:2006.004, May 2006
http://resolver.caltech.edu/CaltechCSTR:2006.004

sets are given instead of learned. However, except for some side information such as hints (Abu-Mostafa, 1995), the training set is the sole information source about the underlying learning problem. Analyzing the complexity of the training set, as we will do in this paper, can actually reveal much useful information about the underlying problem.

This paper is a summary of our initial work on data complexity in machine learning. We focus on binary classification problems, which are briefly introduced in Section 2. We define the data complexity of a training set essentially as the Kolmogorov complexity of the mapping relationship enforced by the set. Any hypothesis that is consistent with the training set would have a program length larger than or equal to that complexity value. The properties of the data complexity and its relationship to some related work are discussed in Section 3.

By studying in Section 4 the data complexity of every subset of the training set, one would find that some subsets are Pareto optimal with respect to the complexity and the size. We call these subsets the *principal subsets*. The full training set is best approximated by the principal subsets at different complexity levels, analogous to the way that a signal is best approximated by the partial sums of its Fourier series. Examples not included in a principal subset are regarded as outliers at the corresponding complexity level. Thus if the decomposition of the training set is known, a learning algorithm with a complexity budget can just train on a proper principal subset to avoid outliers.

However, locating principal subsets is usually computationally infeasible. Thus in Section 5 we discuss efficient ways to identify some principal subsets.

Similar to the Kolmogorov complexity, the ideal data complexity measures are either incomputable or infeasible for practical applications. Some practical complexity measure that approximates the ideal ones has to be used. Thus we also discuss the impact of such approximation to our proposed concepts and methods. For instance, a data pruning strategy based on linear regression is proposed in Section 5 for better robustness.

Some related work is also briefly reviewed at the end of every section. Conclusion and future work can be found in Section 6.

2 Learning Systems

In this section, we briefly introduce some concepts and notations in machine learning, especially for binary classification problems.

We assume that there exists an unknown function f, called the *target function* or simply the *target*, which is a deterministic mapping from the input space $\mathcal{X}$ to the output space $\mathcal{Y}$. We focus on *binary classification problems* in which $\mathcal{Y} = \{0, 1\}$. An *example* or *observation* (denoted by $\mathbf{z}$) is in the form of an input-output pair $(\mathbf{x}, y)$, where the input $\mathbf{x}$ is generated independently from an unknown probability distribution $P_{\mathcal{X}}$, and the output y is computed via $y = f(\mathbf{x})$. A *data set* or *training set* is a set of examples, and is usually denoted by $\mathcal{D} = \{\mathbf{z}_n = (\mathbf{x}_n, y_n)\}_{n=1}^{N}$ with $N = |\mathcal{D}|$, the size of $\mathcal{D}$.

A *hypothesis* is also a mapping from $\mathcal{X}$ to $\mathcal{Y}$. For classification problems, we usually define the *out-of-sample error* of a hypothesis h as the expected error rate,

$$\pi(h) = \mathop{\mathrm{E}}_{\mathbf{x} \sim P_{\mathcal{X}}} [\![h(\mathbf{x}) \neq f(\mathbf{x})]\!],$$

where the Boolean test $[\![\cdot]\!]$ is 1 if the condition is true and 0 otherwise. The goal of learning is to choose a hypothesis that has a low out-of-sample error. The set of all candidate hypotheses (denoted by $\mathcal{H}$) is called the *learning model* or *hypothesis class*, and usually consists of some parameterized functions.

Since both the distribution $P_{\mathcal{X}}$ and the target function f are unknown, the out-of-sample error is inaccessible, and the only information we can access is often limited in the training set $\mathcal{D}$. Thus, instead of looking for a hypothesis h with a low out-of-sample error, a learning algorithm may try to find an h that minimizes the number of errors on the training set,

$$e_{\mathcal{D}}(h) = \sum_{n=1}^{N} [\![h(\mathbf{x}_n) \neq y_n]\!].$$

A hypothesis is said to *replicate* or *be consistent with* the training set if it has zero errors on the training set.

However, having less errors on the training set by itself cannot guarantee a low out-of-sample error. For example, a lookup table that simply memorizes all the training examples has no ability to generalize on unseen inputs. Such an *overfitting* situation is usually caused by endowing the learning model with too much complexity or flexibility. Many techniques such as early stopping and regularization were proposed to avoid overfitting by carefully controlling the hypothesis complexity.

The Bayes rule states that the most probable hypothesis h given the training set $\mathcal{D}$ is the one that has high likelihood $\Pr\{\mathcal{D} \mid h\}$ and prior probability $\Pr\{h\}$,

$$\Pr\{h \mid \mathcal{D}\} = \frac{\Pr\{\mathcal{D} \mid h\} \Pr\{h\}}{\Pr\{\mathcal{D}\}}. \tag{1}$$

Having less errors on the training set makes a high likelihood, but it does not promise a high prior probability. Regularizing the hypothesis complexity is actually an application of Occam's razor, since we believe simple hypotheses should have high prior probabilities.

The problem of finding a generalizing hypothesis becomes harder when the examples contain noise. Due to various reasons, an example may be contaminated in the input and/or the output. When considering only binary classification problems, we take a simple view about the noise—we say an example $(\mathbf{x}, y)$ is an *outlier* or *noisy* if $y = 1 - f(\mathbf{x})$, no matter whether the actual noise is in the input or the output.

3 Data Complexity

In this section, we investigate the complexity of a data set in the context of machine learning. The Kolmogorov complexity and related theories (Li and Vitányi, 1997) are briefly reviewed at the beginning, with a focus on things most relevant to machine learning. Our complexity measures for a data set are then defined, and their properties are discussed. Since the ideal complexity measures are either incomputable or infeasible for practical applications, we also examine practical complexity measures that approximate the ideal ones. At the end of this section, other efforts in quantifying the complexity of a data set are briefly reviewed and compared.

3.1 Kolmogorov Complexity and Universal Distribution

Consider a universal Turing machine $\mathcal{U}$ with input alphabet $\{0, 1\}$ and tape alphabet $\{0, 1, \text{␣}\}$, where ␣ is the blank symbol. A binary string p is a (prefix-free) *program* for the Turing machine $\mathcal{U}$ if and only if $\mathcal{U}$ reads the entire string and halts. For a program p, we use $|p|$ to denote its length in bits, and $\mathcal{U}(p)$ the output of p executed on the Turing machine $\mathcal{U}$. It is possible to have an input string x on an auxiliary tape. In that case, the output of a program p is denoted as $\mathcal{U}(p, x)$.

Given a universal Turing machine $\mathcal{U}$, the Kolmogorov complexity measures the algorithmic complexity of an arbitrary binary string s by the length of the shortest program that outputs s on $\mathcal{U}$. That is, the (prefix) *Kolmogorov complexity* $K_{\mathcal{U}}(s)$ is defined as

$$K_{\mathcal{U}}(s) = \min \{|p| : \mathcal{U}(p) = s\}. \tag{2}$$

$K_{\mathcal{U}}(s)$ can be regarded as the length of the shortest description or encoding for the string s on the Turing machine $\mathcal{U}$. Since universal Turing machines can simulate each other, the choice of $\mathcal{U}$ in (2) would only affect the Kolmogorov complexity by at most a constant that only depends on $\mathcal{U}$. Thus we can drop the $\mathcal{U}$ and denote the Kolmogorov complexity by $K(s)$.

The *conditional Kolmogorov complexity* $K(s \mid x)$ is defined as the length of the shortest program that outputs s given the input string x on the auxiliary tape. That is,

$$K(s \mid x) = \min \{|p| : \mathcal{U}(p, x) = s\}. \tag{3}$$

In other words, the conditional Kolmogorov complexity measures how many additional bits of information are required to generate s given that x is already known. The Kolmogorov complexity is a special case of the conditional one where x is empty.

For an arbitrary binary string s, there are many programs for a Turing machine $\mathcal{U}$ that output s. If we assume a program p is randomly picked with probability $2^{-|p|}$, the probability that a random program would output s is

$$P_{\mathcal{U}}(s) = \sum_{p \colon \mathcal{U}(p)=s} 2^{-|p|}. \tag{4}$$

The sum of $P_{\mathcal{U}}$ of all binary strings is clearly bounded by 1 since no program can be the prefix of another. The $\mathcal{U}$ can also be dropped since the choice of $\mathcal{U}$ in (4) only affects the probability by no more than a constant factor independent of the string. This partly justifies why P is named the *universal distribution*. The other reason is that the universal distribution P dominates any computable distributions by up to a multiplicative constant, which makes P the *universal prior*.

The Kolmogorov complexity and the universal distribution are closely related, since we have $K(s) \approx -\log P(s)$ and $P(s) \approx 2^{-K(s)}$. The approximation is within a constant additive or multiplicative factor independent of s. This is intuitive since the shortest program for s gives the most weight in (4).

The Bayes rule for learning (1) can be rewritten as

$$-\log \Pr \{h \mid \mathcal{D}\} = -\log \Pr \{\mathcal{D} \mid h\} - \log \Pr \{h\} + \log \Pr \{\mathcal{D}\}. \tag{5}$$

The most probable hypothesis h given the training set $\mathcal{D}$ would minimize $-\log \Pr \{h \mid \mathcal{D}\}$. Let's assume for now a hypothesis is also an encoded binary string. With the universal prior in place, $-\log \Pr \{h\}$ is roughly the code length for the hypothesis h, and $-\log \Pr \{\mathcal{D} \mid h\}$ is in general the minimal description length of $\mathcal{D}$ given h. This leads to the *minimum description length* (MDL) principle (Rissanen, 1978) which is a formalization of Occam's razor: the best hypothesis for a given data set is the one that minimizes the sum of the code length of the hypothesis and the code length of the data set when encoded by the hypothesis.

Both the Kolmogorov complexity and the universal distribution are incomputable.

3.2 Universal Data Complexity

As we have seen, for an arbitrary string, the Kolmogorov complexity $K(s)$ is a universal measure for the amount of information needed to replicate s, and $2^{-K(s)}$ is a universal prior probability of s. In machine learning, we care about similar perspectives: the amount of information needed to approximate a target function, and the prior distribution of target functions. Since a training set is usually the only information source about the target, we are thus interested in the amount of information needed to replicate a training set, and the prior distribution of training sets. In short, we want a complexity measure for a training set.

Unlike the Kolmogorov complexity of a string for which the exact replication of the string is mandatory, one special essence about "replicating" a training set is that the exact values of the inputs and outputs of examples do not matter. What we really want to replicate is the input-output relationship enforced by the training set, since this is where the target function is involved. The unknown input distribution $P_{\mathcal{X}}$ might be important for some machine learning problems. However, given the input part of the training examples, it is also irrelevant to our task.

At first glance the conditional Kolmogorov complexity may seem suitable for measuring the complexity of replicating a training set. Say for $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, we collect the inputs and the outputs of all the examples, and apply the conditional Kolmogorov complexity to the outputs given the inputs, i.e.,

$$K(y_1, y_2, \ldots, y_N \mid \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N).$$

This conditional complexity, as defined in (3), finds the shortest program that takes as a whole all the inputs and generates as a whole all the outputs. In other words, the shortest program that maps all the inputs to all the outputs. The target function, if encoded properly as a program, can serve the mapping with an extra loop to take care of the N inputs, as will any hypotheses that can replicate the training set.

However, with this measure one has to assume some permutation of the examples. This is not only undesired but also detrimental in that some "clever" permutations would ruin the purpose of reflecting the amount of information in approximating the target. Say there are N_0 examples that have 0 as the output. With permutations that put examples having output 0 before those having output 1, the shortest program would probably just encode the numbers N_0 and $(N - N_0)$, and print a string of N_0 zeros and $(N - N_0)$ ones. The conditional Kolmogorov complexity would be approximately $\log N_0 + \log(N - N_0) + O(1)$, no matter how complicated the target might be.

Taking into consideration that the order of the examples should not play a role in the complexity measure, we define the *data complexity* as

Definition 1. *Given a fixed universal Turing machine $\mathcal{U}$, the data complexity of a data set $\mathcal{D}$ is*

$$C_{\mathcal{U}}(\mathcal{D}) = \min \{|p| : \forall\, (\mathbf{x}, y) \in \mathcal{D}, \mathcal{U}(p, \mathbf{x}) = y\}\,.$$

That is, the data complexity $C_{\mathcal{U}}(\mathcal{D})$ is the length of the shortest program that can correctly map every input in the data set $\mathcal{D}$ to its corresponding output.

Similar to the Kolmogorov complexity, the choice of the Turing machine can only affect the data complexity up to a constant. Formally, we have this invariance theorem.

Theorem 1. *For two universal Turing machines $\mathcal{U}_1$ and $\mathcal{U}_2$, there exists a constant c that only depends on $\mathcal{U}_1$ and $\mathcal{U}_2$, such that for any data set $\mathcal{D}$,*

$$|C_{\mathcal{U}_1}(\mathcal{D}) - C_{\mathcal{U}_2}(\mathcal{D})| \le c. \tag{6}$$

Proof. Let $\langle \mathcal{U}_1 \rangle$ be the encoding of $\mathcal{U}_1$ on $\mathcal{U}_2$. Any program p for $\mathcal{U}_1$ can be transformed to a program $\langle \mathcal{U}_1 \rangle\, p$ for $\mathcal{U}_2$. Thus $C_{\mathcal{U}_2}(\mathcal{D}) \le C_{\mathcal{U}_1}(\mathcal{D}) + |\langle \mathcal{U}_1 \rangle|$. Let $\langle \mathcal{U}_2 \rangle$ be the encoding of $\mathcal{U}_2$ on $\mathcal{U}_1$. By symmetry, we have $C_{\mathcal{U}_1}(\mathcal{D}) \le C_{\mathcal{U}_2}(\mathcal{D}) + |\langle \mathcal{U}_2 \rangle|$. So (6) holds for $c = \max \{|\langle \mathcal{U}_1 \rangle|\,, |\langle \mathcal{U}_2 \rangle|\}$. $\square$

Thus the data complexity is also universal, and we can drop the $\mathcal{U}$ and simply write $C(\mathcal{D})$.

Unfortunately, the data complexity is also not a computable function.

Lemma 1. *$C(\mathcal{D}) \leq K(\mathcal{D}) + c$ where c is a constant independent of $\mathcal{D}$.*

Proof. Let p be the shortest program that outputs $\mathcal{D}$. Consider another program p' that takes an input $\mathbf{x}$, calls p to generate $\mathcal{D}$ on an auxiliary tape, searches $\mathbf{x}$ within the inputs of examples on the auxiliary tape, and returns the corresponding output if $\mathbf{x}$ is found and 0 otherwise. The program p' adds a "shell" with constant length c to the program p, and c is independent of $\mathcal{D}$. Thus $C(\mathcal{D}) \leq |p'| = |p| + c = K(\mathcal{D}) + c$. □

Lemma 2. *The data complexity $C(\cdot)$ is not upper bounded.*

Proof. Consider any target function $f\colon \{0,1\}^m \to \{0,1\}$ that accepts m-bit binary strings as inputs. A data set including all possible m-bit binary inputs and their outputs from the target f would fully decide the mapping from $\{0,1\}^m$ to $\{0,1\}$. Since the Kolmogorov complexity of such mapping (for all integer m) is not upper bounded (Abu-Mostafa, 1988b,a), neither is $C(\cdot)$. □

Theorem 2. *The data complexity $C(\cdot)$ is incomputable.*

Proof. We show this by contradiction. Assume there is a program p to compute $C(\mathcal{D})$ for any data set $\mathcal{D}$. Consider another program p' that accepts an integer input l, enumerates over all data sets, uses p to compute the data complexity for each data set, and stops and returns the first data set that has complexity at least l. Due to Lemma 2, the program p' will always halt. Denote the returned data set as $\mathcal{D}_l$. Since the program p' together with the input l can generate $\mathcal{D}_l$, we have $K(\mathcal{D}_l) \leq |p'| + K(l)$. By Lemma 1 and the fact that $C(\mathcal{D}_l) \geq l$, we obtain $l \leq K(l) + |p'| + c$, where c is the constant in Lemma 1. This is contradictory for l large enough since we know $K(l)$ is upper bounded by $\log l$ plus some constant. □

With fixed inputs, a universal prior distribution can be defined on all the possible outputs, just similar to the universal prior distribution. However, the details will not be discussed in this paper.

3.3 Data Complexity with Learning Models

Using our notions in machine learning, the universal data complexity is the length of the shortest hypothesis that replicates the data set, given that the learning model is the set of all programs. However, it is not common that the learning model includes all possible programs. For a limited set of hypotheses, we can also define the data complexity.

Definition 2. *Given a learning model $\mathcal{H}$, the data complexity of a data set $\mathcal{D}$ is*

$$C_{\mathcal{H}}(\mathcal{D}) = \min \{|h| : h \in \mathcal{H} \text{ and } \forall\, (\mathbf{x}, y) \in \mathcal{D},\, h(\mathbf{x}) = y\}.$$

This definition is almost the same as Definition 1, except that program p has now been replaced with hypothesis $h \in \mathcal{H}$. An implicit assumption is that there is a way to measure the "program length" or complexity of any hypothesis in the learning model. Here we assume an encoding scheme for the learning model that maps a hypothesis to a prefix-free binary codeword. For example, the encoding scheme for feed-forward neural networks (Bishop, 1995) can be the concatenation of the number of network layers, the number of neurons in every layer, and the weights of every neuron, with each number represented by a self-delimited binary string. We also assume that a program $p_{\mathcal{H}}$, called the *interpreter* for the learning model $\mathcal{H}$, can take a codeword and emulate the encoded hypothesis.

Thus $|h|$, the complexity of the hypothesis h, is defined as the length of its codeword.[2] It is easy to see that $C_{\mathcal{U}}(\mathcal{D}) \le |p_{\mathcal{H}}| + C_{\mathcal{H}}(\mathcal{D})$.

The data complexity as Definition 2 is in general not universal, i.e., it depends on the learning model and the encoding scheme, since full simulation of one learning model by another is not always possible. Even with the same learning model, two encoding schemes could differ in a way that it is impossible to bound the difference in the codeword lengths of the same hypothesis.

Definition 2 requires that some hypothesis in the learning model can replicate the data set. This is probably reasonable if the target is in the learning model and the data set is also noiseless. What if the target is not in the learning model or there is noise in the data set? A data set might not be consistent with any of the hypotheses, and thus the data complexity is not defined for it. Actually in the case of noisy data sets, even if there are hypotheses that are consistent, it is not desirable to use their complexity as the data complexity. The reason will be clear later in this subsection. In summary, we need another definition that can take care of replication errors.

Consider a hypothesis h that is consistent with all the examples except $(\mathbf{x}_1, y_1)$. We can construct a program p by memorizing input $\mathbf{x}_1$ with a lookup table entry:

$$p = \text{if input is } \mathbf{x}_1\text{, then output } y_1\text{; else run the interpreter } p_{\mathcal{H}} \text{ on } h.$$

Excluding the length of the interpreter, which is common to all hypotheses, the program p is just a little longer than h, but can perfectly replicate the data set. Actually, the increase of the program length is the length of the "if ... then ... else" structure plus the Kolmogorov complexity of $\mathbf{x}_1$ and y_1. For a hypothesis that has more than one error, several lookup table entries can be used.[3] If we assume the increase in the program length is a constant for every entry, we have

Definition 3. *Given a learning model $\mathcal{H}$ and a proper positive constant λ, the data complexity (with a lookup table) of a data set $\mathcal{D}$ is*

$$C_{\mathcal{H},\lambda}(\mathcal{D}) = \min \{|h| + \lambda e_{\mathcal{D}}(h) \colon h \in \mathcal{H}\}.$$

The constant λ can be seen as the equivalent complexity of implementing one lookup table entry with the learning model. It can also be regarded as the complexity cost of one error. Definition 2 does not allow any errors, so the data complexity $C_{\mathcal{H}}(\cdot)$ is actually $C_{\mathcal{H},\infty}(\cdot)$.

For positive and finite λ, the data complexity $C_{\mathcal{H},\lambda}(\mathcal{D})$ is actually computable. This is because the complexity is bounded by $\min\{|h| : h \in \mathcal{H}\} + \lambda|\mathcal{D}|$, and we can enumerate all codewords that are not longer than that bound.[4]

Given a learning model $\mathcal{H}$ and an encoding scheme, which determines the hypothesis complexity, we consider a prior of hypotheses where $\Pr\{h\} = 2^{-|h|}$. Let's also assume a Bernoulli noise model where the probability of an example being noisy is ε. This gives the likelihood as

$$\Pr\{\mathcal{D} \mid h\} = \varepsilon^{e_{\mathcal{D}}(h)}(1-\varepsilon)^{N-e_{\mathcal{D}}(h)} = (1-\varepsilon)^N \left(\varepsilon^{-1} - 1\right)^{-e_{\mathcal{D}}(h)}.$$

---

[2] This also includes the possibility of using a universal Turing machine as the interpreter and directly mapping a hypothesis to a program. In this case, $|h|$ is the program length.

[3] There are other general ways to advise that h fails to replicate the example $(\mathbf{x}_1, y_1)$. Here is another one:

$$p = \text{let } y = h(\text{input})\text{; if input is in } \{\mathbf{x}_1\}\text{, then output } 1-y\text{; else output } y.$$

When there are several erroneous examples, $\{\mathbf{x}_1\}$ can be replaced with the set of the inputs of the erroneous examples. If only very basic operations are allowed for constructing p from h, all these ways lead to the same Definition 3 of the data complexity.

[4] Well, we also assume that every hypothesis, simulated by the interpreter, always halts. This is true for any reasonable learning models.

And according to (5), we have

$$-\log \Pr\{h \mid \mathcal{D}\} = |h| + e_{\mathcal{D}}(h) \cdot \log\left(\varepsilon^{-1} - 1\right) + c,$$

where $c = \log \Pr\{\mathcal{D}\} - N \log(1-\varepsilon)$ is a constant independent of the hypothesis h. To maximize the posterior probability $\Pr\{h \mid \mathcal{D}\}$ or to minimize $-\log \Pr\{h \mid \mathcal{D}\}$ is equivalent to minimize the sum of the hypothesis complexity and the error cost for $C_{\mathcal{H},\lambda}(\mathcal{D})$, with $\lambda = \log\left(\varepsilon^{-1} - 1\right)$. And the case of $C_{\mathcal{H}}(\cdot)$ or $C_{\mathcal{H},\infty}(\cdot)$ corresponds to $\varepsilon = 0$. The Bayesian point of view justifies the use of λ, and also emphasizes that the encoding scheme should be based on a proper prior of hypotheses.

We also have this straightforward property:

Theorem 3. $C_{\mathcal{H},\lambda}(\mathcal{D}) \le C_{\mathcal{H},\lambda}(\mathcal{D} \cup \mathcal{D}') \le C_{\mathcal{H},\lambda}(\mathcal{D}) + \lambda\,|\mathcal{D}'|$.

The first inequality says that the data complexity is increasing when more examples are added. The second inequality states that the increase of the complexity is at most $\lambda\,|\mathcal{D}'|$, the cost of treating all the added examples with lookup table entries. The increase would be less if some of the added examples can be replicated by the shortest hypothesis for $\mathcal{D}$, or can form patterns which are shorter than lookup table entries. More will be discussed on these two cases when the complexity-error path is introduced (Definition 6 on page 19).

For the rest of the paper, we will mostly work with Definition 2 and Definition 3, and we will use just $C(\cdot)$ for $C_{\mathcal{H},\lambda}(\cdot)$ or $C_{\mathcal{U}}(\cdot)$ when the meaning is clear from the context. Because lookup table entries are an integrated part of Definition 3, we will also simply use "hypothesis" to mean a hypothesis together with lookup table entries. Thus all the three data complexity definitions can be unified as the length of the shortest consistent hypothesis. We use $h_{\mathcal{D}}$ to denote one of the shortest hypotheses that can replicate $\mathcal{D}$.

We will also assume that any mechanisms for memorizing individual examples, no matter whether it is built-in or implemented as lookup table entries as in Definition 3, would cost the same complexity as a lookup table. In other words, if an example cannot help build patterns for other examples, adding it to a set would increase the data complexity by λ.

3.4 Practical Measures

Although we now have three data complexity measures, none of them is feasible in practice. The universal data complexity $C_{\mathcal{U}}(\cdot)$ is incomputable. The data complexity defined on a learning model $\mathcal{H}$, $C_{\mathcal{H}}(\cdot)$, may be computable for some $\mathcal{H}$, but finding a hypothesis that is consistent with the data set is usually NP-complete, not to mention finding a shortest one. The data complexity with a lookup table $C_{\mathcal{H},\lambda}(\cdot)$ seems the most promising to be used in practice. But it also suffers from the exponential time complexity in searching for a shortest hypothesis (with errors). We need to have some approximate complexity measure for practical applications.

A reasonable approximation to $C_{\mathcal{H}}(\cdot)$ or $C_{\mathcal{H},\lambda}(\cdot)$ can be obtained as a byproduct of the learning procedure. A learning algorithm usually minimizes the number of errors plus some regularization term over the learning model, and the regularization term is usually meant to approximate the complexity (encoding length) of a hypothesis. Thus some information about the learned hypothesis can be used as a practical data complexity measure. For example, the number of different literals used to construct a mixed DNF-CNF rule was used by Gamberger and Lavrač (1997). In the following text, we will deduce another practical data complexity measure based on the hard-margin support vector machine.

The hard-margin support vector machine (SVM) (Vapnik, 1999) is a learning algorithm that finds an optimal hyperplane to separate the training examples with maximal minimum margin. A hyperplane is defined as $\langle \mathbf{w}, \mathbf{x} \rangle - b = 0$, where $\mathbf{w}$ is the weight vector and b is the bias. Assuming the training set is linearly separable, SVM solves the optimization problem below:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \|\mathbf{w}\|^2 , \\ \text{subject to} \quad & y_n \left(\langle \mathbf{w}, \mathbf{x}_n \rangle - b \right) \geq 1, \quad n = 1, \ldots, N. \end{aligned} \tag{7}$$

The dual problem is

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle - \sum_{n=1}^{N} \alpha_n , \\ \text{subject to} \quad & \alpha_n \geq 0, \quad n = 1, \ldots, N, \\ & \sum_{n=1}^{N} y_n \alpha_n = 0. \end{aligned}$$

The optimal weight vector, given the optimal α^* for the dual problem, is a linear combination of the training input vectors,

$$\mathbf{w}^* = \sum_{n=1}^{N} y_n \alpha_n^* \mathbf{x}_n . \tag{8}$$

Note that only the so-called *support vectors*, for which the equality in (7) is achieved, can have nonzero coefficients α_n^* in (8).

For a linearly nonseparable training set, the kernel trick (Aizerman et al., 1964) is used to map input vectors to a high-dimensional space and an optimal separating hyperplane can be found there. Denote the inner product in the mapped space of two inputs $\mathbf{x}$ and $\mathbf{x}'$ as $\mathcal{K}(\mathbf{x}, \mathbf{x}')$, the so-called kernel operation. The dual problem with the kernel trick uses the kernel operation instead of the normal inner product, and the optimal hypothesis (a hyperplane in the mapped space but no longer a hyperplane in the input space) becomes

$$\sum_{n=1}^{N} y_n \alpha_n^* \mathcal{K}(\mathbf{x}_n, \mathbf{x}) - b^* = 0.$$

Since the mapped space is usually high-dimensional or even infinite-dimensional, it is reasonable to describe the SVM hypothesis by listing the support vectors and their coefficients. Thus the descriptive length is approximately $(Mc_1 + c_2 + c_3)$, where M is the number of support vectors, c_1 is the average Kolmogorov complexity of describing an input vector and a coefficient, c_2 is the Kolmogorov complexity of the bias, and c_3 is the descriptive length of the summation and the kernel operation. Since c_3 is a common part for all SVM hypotheses using the same kernel, and c_2 is relatively small compared to c_1, we can use just the number of support vectors, M, as the complexity measure for SVM hypotheses.

With some minor conditions, SVM with powerful kernels such as the stump kernel and the perceptron kernel (Lin and Li, 2005a,b) can always perfectly replicate a training set. Thus the measure based on SVM with such kernels fit well with Definition 2. In the experiments for this paper, we used the perceptron kernel, which usually has comparable learning performance to the popular Gaussian kernel, but do not require a parameter selection (Lin and Li, 2005b).

Note that SVM can also be trained incrementally (Cauwenberghs and Poggio, 2001). That is, if new examples are added after an SVM has already been learned on the training set, the hyperplane can be updated to accommodate the new examples in an efficient way. Such capability of incrementally computing the complexity can be quite useful in some applications, such as data pruning (Subsection 5.2) and deviation detection (Arning et al., 1996).

3.5 Related Work

Our data complexity definitions share some similarity to the randomness of decision problems. Abu-Mostafa (1988b,a) discussed decision problems where the input space of the target function was finite, and defined the randomness of a problem based on the Kolmogorov complexity of the target's truth table. The randomness can also be based on the length of the shortest program that implements the target function, which is essentially equivalent to the previous definition (Abu-Mostafa, 1988a). However, in our settings, the input space is infinite and the training set includes only finite examples; hence an entire truth table is infeasible. Thus the second way, which we have adopted, seems to be the only reasonable definition.

Definition 3, the data complexity with a lookup table, is also similar to the two-part code length of the minimum description length (MDL) principle (Rissanen, 1978; Grünwald, 2005). The two-part scheme explains the data via encoding a hypothesis for the data, and then encoding the data with the help of the hypothesis. The latter part usually takes care of the discrepancy information between the hypothesis and the data, just like in Definition 3. However, in our definition, the inputs of the data are not encoded, and we explicitly ignore the order of examples when considering the discrepancy.

The data complexity is also conceptually aligned with the CLCH value (complexity of the least complex correct hypothesis) proposed by Gamberger and Lavrač (1997). They required the complexity measure for hypotheses, which is the program length in this paper, to be "reasonable." That is, for two hypotheses, h_1 and h_2, where h_2 is obtained by "conjunctively or disjunctively adding conditions" to h_1, h_1 should have no larger complexity than h_2. However, this intuitively correct requirement is actually troublesome. For instance, h_1 recognizes all points in a fixed hexagon, and h_2 recognizes all points in a fixed triangle enclosed in that hexagon. Although h_2 can be obtained by adding more constraints on h_1, it is actually simpler than h_1. Besides, their definition of a set being "saturated" and the corresponding "saturation test" depend heavily on the training set being large enough to represent the target function, which might not be practical.

Except the usual complexity measures based on logic clauses, not many practical complexity measures have been studied. Schmidhuber (1997) implemented a variant of the general universal search (Levin, 1973) to find a neural network with a close-to-minimal Levin complexity. Although the implementation is only feasible on very simple toy problems, his experiments still showed that such search, favoring short hypotheses, led to excellent generalization performance, which reinforced the validity of Occam's razor in learning problems.

Wolpert and Macready (1999) proposed a very interesting complexity measure called self-dissimilarity. They observed that many complex systems tend to exhibit different structural patterns over different space and time scale. Thus the degrees of self-dissimilarity between the various scales with which a system is examined constitute a complexity signature of that system. It is mainly a complexity measure for a system, or a target function in our context, which can provide information at different scales, and is not straightforward to be applied to a data set.

4 Data Decomposition

In this section, we discuss the issue of approximating a data set with its subsets. Compared with the full data set, a subset is in general simpler (lower data complexity) but less informative (fewer examples). In addition, different subsets can form different patterns, and thus lead to different combinations of the data complexity and the subset size. We show that the principal subsets, defined later in this section, have the Pareto optimal combinations and best approximate the full set at different complexity levels. The concept of the principal subsets is also useful for data pruning (Section 5).

4.1 Complexity-Error Plots

When there is only one class of examples, the data complexity is a small constant. Only with examples from both classes can more interesting patterns be formed. Given a training set $\mathcal{D}$, different subsets of $\mathcal{D}$ may form different patterns, and thus lead to different complexity values.

Figure 1(a) shows a toy learning problem with a target consisting of three concentric disks. The

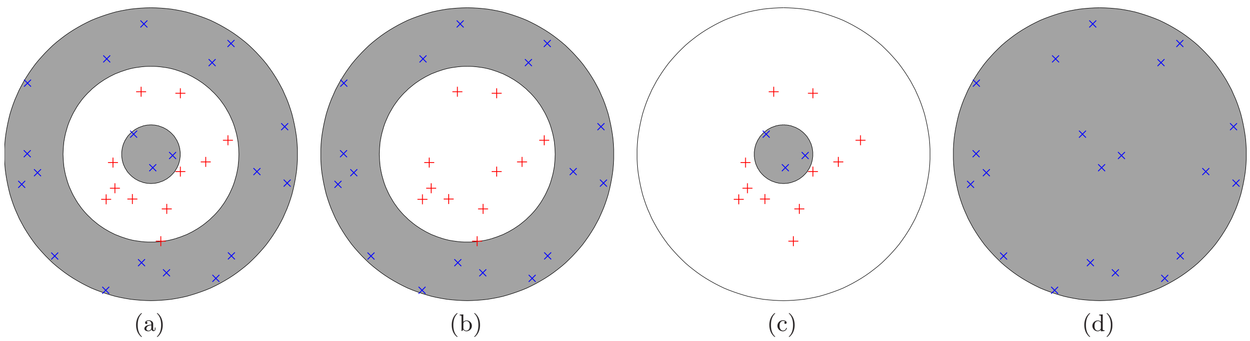

Figure 1. Subsets of examples from the concentric disks problem: (a) all examples; (b) examples from the two outer rings; (c) examples within the middle disk; (d) all "×" examples

target is depicted with the "white" and "gray" backgrounds in the plot—examples on the white background are classified as class 1, and examples on the gray background are classified as 0. The examples in the plot were randomly generated and are marked with "+" and "×" according to their class labels. The other three plots in Figure 1 illustrate how different subsets of the examples can be explained by hypotheses of different complexity levels, and thus may have different complexity values. We also see that different subsets approximate the full set to different degrees.

For a given data set $\mathcal{D}$, we are interested in all possible combinations of the data complexity and the approximation accuracy of its subsets. Consider the following set of pairs:

$$\Omega_1 = \{(C(\mathcal{S}), |\mathcal{D}| - |\mathcal{S}|) : \mathcal{S} \subseteq \mathcal{D}\}\,. \tag{9}$$

Here we use $|\mathcal{D}| - |\mathcal{S}|$ as the approximation error of $\mathcal{S}$.[5] The set Ω_1 can be regarded as a plot of points on the 2-D plane. For each subset $\mathcal{S}$, there is a point in Ω_1 with the horizontal axis giving the data complexity and the vertical axis showing the approximation error. Such a plot is called the *subset-based complexity-error plot* (see Figure 2).

We can also consider another set built upon programs or hypotheses:

$$\Omega_2 = \{(|h|\,, e_{\mathcal{D}}(h)) : h \in \mathcal{H}\}\,.$$

This set, Ω_2, has a point for each hypothesis h in the learning model, depicting the complexity of the hypothesis and the number of errors on the training set. It is called the *hypothesis-based complexity-error plot*. Note that the hypothesis h and the learning model $\mathcal{H}$ shall agree with the data complexity measure $C(\cdot)$ used in (9). For example, if the data complexity measure allows lookup tables, the learning model $\mathcal{H}$ would then includes hypotheses appended with lookup tables of all sizes.

The two plots in Figure 2 demonstrate for a fictional training set how the two sets of pairs look. Here are some observations for the two complexity-error plots:

[5] If we regard $\mathcal{S}$ as a lookup table, the error of the lookup table on the full set is $|\mathcal{D}| - |\mathcal{S}|$.

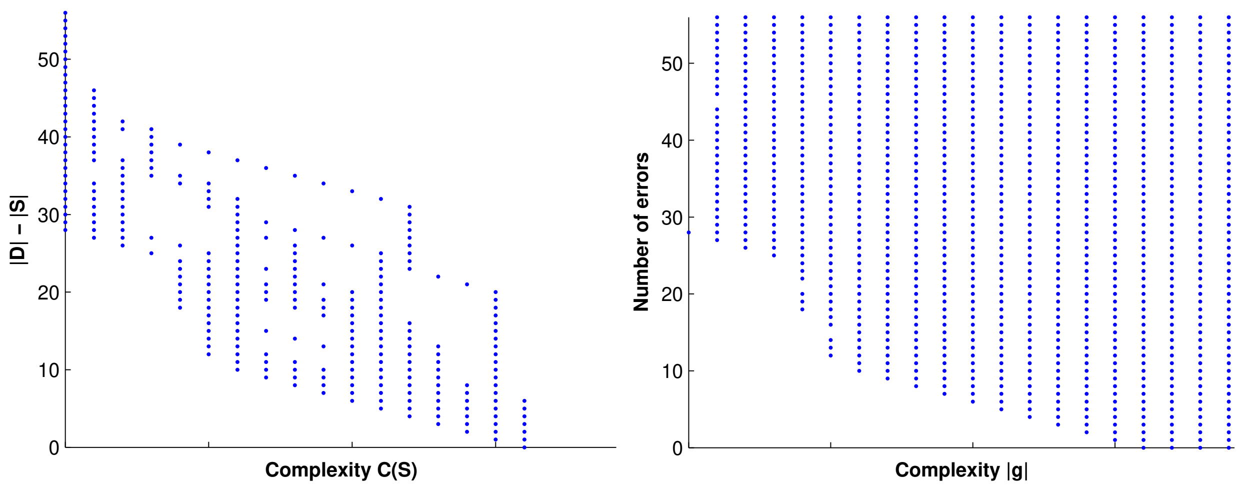

Figure 2. Fictional complexity-error plots for (left) Ω_1 and (right) Ω_2

1. For each point in Ω_1, there is at least one subset $\mathcal{S}$ associated with it. The point associated with $\mathcal{S}$ also shows the complexity of $h_{\mathcal{S}}$.
2. There is a subset $\mathcal{D}_h$ associated with each hypothesis h,

$$\mathcal{D}_h = \{(\mathbf{x}, y) : (\mathbf{x}, y) \in \mathcal{D} \text{ and } h(\mathbf{x}) = y\} .$$

   And $e_{\mathcal{D}}(h) = |\mathcal{D}| - |\mathcal{D}_h|$. Thus the point associated with h also shows the approximation error of the subset $\mathcal{D}_h$.
3. The leftmost points in both plots are with subsets of only one class, since that gives the lowest complexity.
4. The points on the horizontal axis are associated with the full set.
5. For any point in Ω_1, there is a point in Ω_2 that has the same complexity value but a smaller or equal error value. This is because $|\mathcal{D}_{h_{\mathcal{S}}}| \geq |\mathcal{S}|$ when $\mathcal{S} \subseteq \mathcal{D}$.
6. For any point in Ω_2, there is a point in Ω_1 that has the same error value but a smaller or equal complexity value. This is because $C(\mathcal{D}_h) \leq |h|$.

4.2 Principal Points and Principal Subsets

The two complexity-error plots depict all the possible combinations of the data complexity and the approximation error for subsets of the training set. In general, if one subset or hypothesis gets more examples correct, it would be more complex. However, with the same data complexity, some subsets may contain more examples than others; and with the same size, some subsets may be simpler than others. Ideally, to approximate the full set, we want a subset to have the most examples but the least complexity.

With respect to the data complexity and the approximation error, some points in a complexity-error plot are optimal in the sense that no other points are better than them. They are called the *principal points*:

Definition 4. *A point (c, e) in a complexity-error plot is a principal point if and only if we have for any other point (c', e') in the plot, $c' > c$ or $e' > e$.*

In other words, a principal point is a Pareto optimal point, since there are no other points that have one coordinate smaller without making the other coordinate larger.

Although the subset-based complexity-error plot looks quite different from the hypothesis-based complexity-error plot, they actually have the same set of principal points.

Theorem 4. *The subset-based and hypothesis-based complexity-error plots have the same set of principal points.*

Proof. The proof utilizes the observations in the previous subsection that relates points in these two plots. For any principal point $(c, e) \in \Omega_1$, there is a point $(c_2, e_2) \in \Omega_2$ with $c_2 = c$ and $e_2 \leq e$. If (c_2, e_2) is not a principal point in Ω_2, we can find a point $(c'_2, e'_2) \in \Omega_2$ such that $c'_2 \leq c_2$, $e'_2 \leq e_2$, and at least one inequality would be strict; otherwise let $c'_2 = c_2$ and $e'_2 = e_2$. For (c'_2, e'_2), there is a point $(c', e') \in \Omega_1$ with $c' \leq c'_2$ and $e' = e'_2$. Overall we have $c' \leq c$ and $e' \leq e$, and either $e_2 < e$ or (c_2, e_2) not being a principal point will make at least one inequality be strict, which contradicts the assumption that (c, e) is a principal point in Ω_1. Thus $e = e_2$ and (c, e) is also a principal point in Ω_2. Likewise we can also prove that any principal point of Ω_2 is also a principal point in Ω_1. □

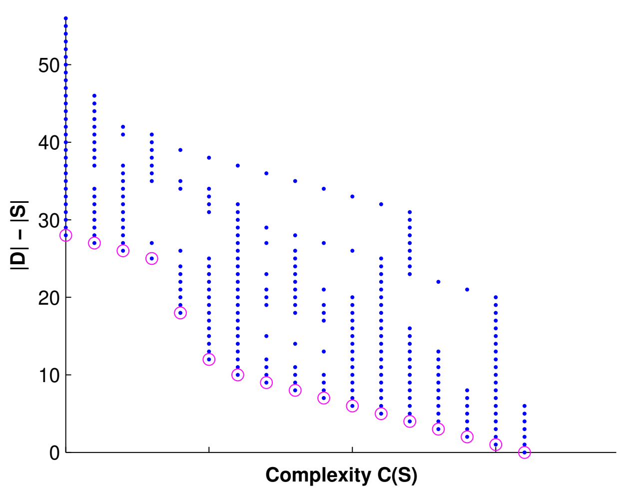

Figure 3. A fictional complexity-error plot with principal points circled

Figure 3 shows the principal points in the subset-based complexity-error plot from the last fictional problem. Note that each principal point is associated with at least one subset and one hypothesis. Each principal point represents some optimal trade-off between the data complexity and the size. To increase the size of the associated subset, we have to go to a higher complexity level; to reduce the data complexity, we have to remove examples from the subset. Each principal point also represents some optimal trade-off between the hypothesis complexity and the hypothesis error. To decrease the error on the training set, a more complex hypothesis should be sought; to use a simpler hypothesis, more errors would have to be tolerated. Thus the principal point at a given complexity level gives the optimal error level, and implicitly the optimal subset to learn, and the optimal hypothesis to pick.

A subset associated with a principal point is called a *principal subset*. The above arguments actually say that a principal subset is a best approximation to the full set at some complexity level.

4.3 Toy Problems

We verify the use of the data complexity in data decomposition with two toy problems. The practical data complexity measure is the number of support vectors in a hard-margin SVM with the perceptron kernel (see Subsection 3.4).

The first toy problem is the concentric disks problem with 31 random examples (see page 11). To have the subset-based complexity-error plot, we need to go over all the $(2^{31} - 1)$ subsets and compute the data complexity for each subset. To make the job computationally more feasible, we cluster the examples as depicted in Figure 4 by examples connected with dotted lines, and only examine subsets that consist of whole clusters.[6] The complexity-error plot using these 15 clusters, with principal points circled, is shown in Figure 5.

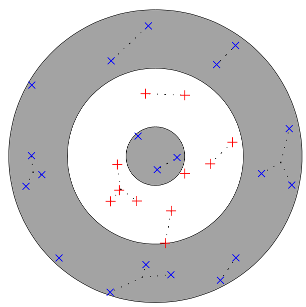

Figure 4. 15 clusters of the concentric disks problem, shown as groups of examples connected with dotted lines

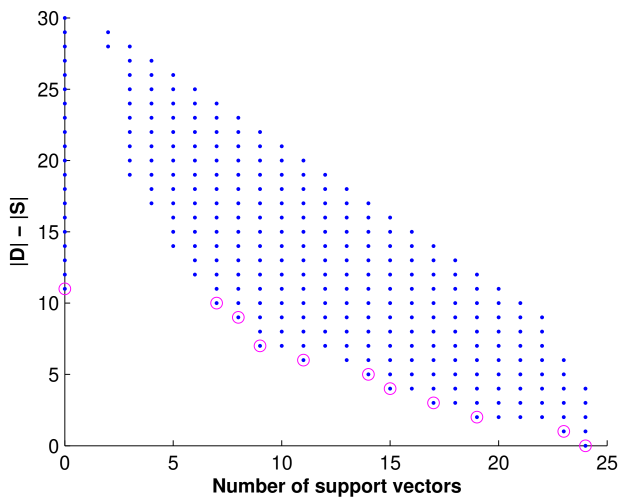

Figure 5. The complexity-error plot based on the clusters of the concentric disks problem (Figure 4), with the principal points circled

There are 11 principal points associated with 17 principal subsets. In Figure 14 on page 30, we list all the 17 principal subsets, of which three selected ones are shown in Figure 6. The data

[6] The 15 clusters in Figure 4 were manually chosen based on example class and distance, such that each cluster contains only examples of the same class, and is not too close to examples of the other class. Although this could be done by some carefully crafted algorithm, we did it manually since the training set is small.

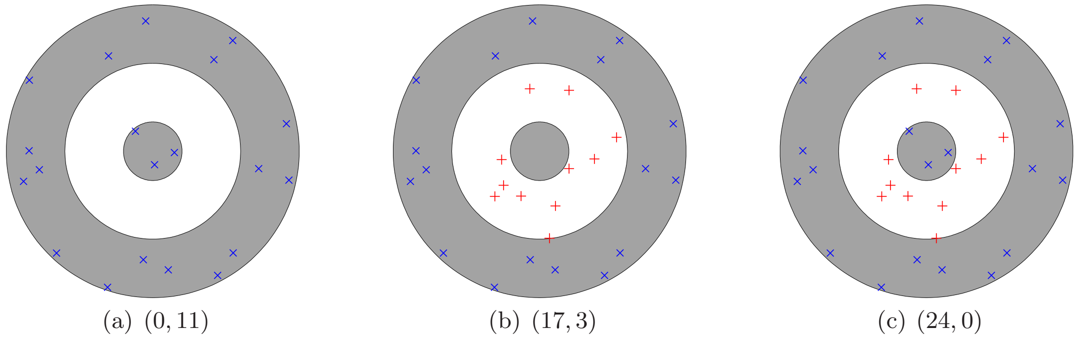

Figure 6. Three selected principal subsets of the concentric disks problem (complexity-error pairs are also listed)

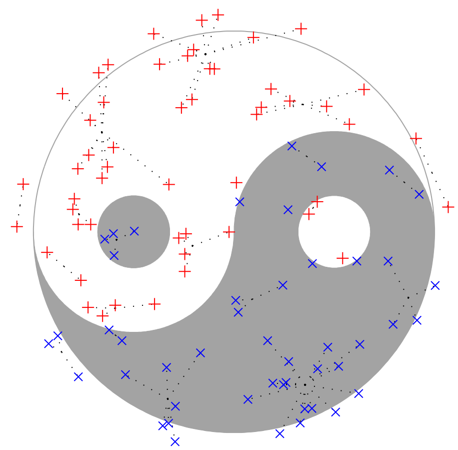

Figure 7. 32 clusters of the Yin-Yang problem, shown as groups of examples connected with dotted lines

complexity based on SVM-perceptron and the number of errors are also listed below the subset plots, and the white and gray background depicts the target function. Plot (a) shows the situation where a simplest hypothesis predicts the negative class regardless of the actual inputs. The subset is same as the one in Figure 1(d) on page 11. Plot (c) is the full training set with the highest data complexity, same as the one in Figure 1(a). The middle one, plot (b) or Figure 1(b), gives an intermediate situation such that the two classes of examples in the two outer rings are replicated, but the examples in the inner disk are deserted. This implies that, at that level of complexity, examples in the inner disk should rather be regarded as outliers than exceptions to the middle disk.

The second toy problem is about the Yin-Yang target function used by Li et al. (2005), which is also a 2-D binary classification problem. The background colors in Figure 7 depict how the Yin-Yang target classifies examples within a round plate centered at the origin; examples out of the plate belong to the Yang (white) class if it is in the upper half-plane. The training set consists of 100 examples randomly picked within a circle slightly larger than the plate. Clustering is also required for generating the complexity-error plot. Figure 7 also shows the 32 manually chosen clusters. The resulted complexity-error plot, based on the practical data complexity measure with SVM-perceptron, is show in Figure 8.

This time we get 25 principal points and 48 associated principal sets (see Figure 15 on page 31 and Figure 16 on page 32 for most of them). Here in Figure 9, we list and organize with arrows 11 principal sets that we think are representative. With the flow of arrows, the data complexity

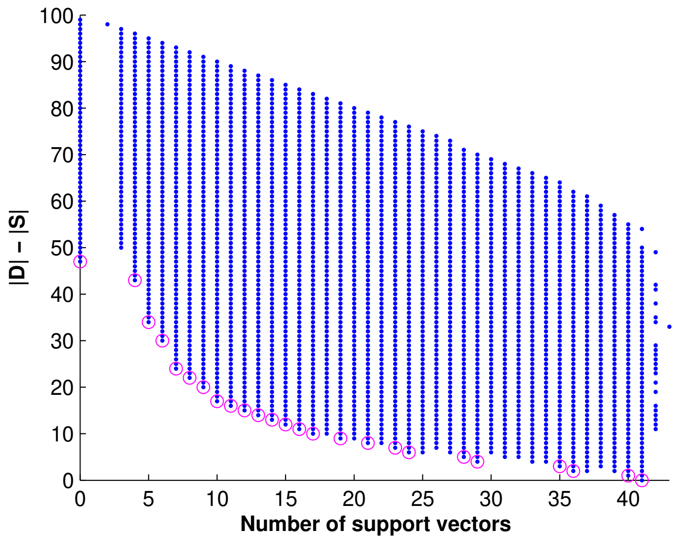

Figure 8. The complexity-error plot based on the clusters of the Yin-Yang problem (Figure 7), with the principal points circled

increases and the number of errors goes down. The first principal subset shows a simplest hypothesis classifying any input as the Yang (white) class. The one to the right shows a slightly more complex hypothesis that seems to separate the two classes by a line. From now on, the classification patterns fork in slightly different routes. The two subsets in the left route with points $(12, 15)$ and $(16, 11)$ continue the trend to separate two classes by relatively straight boundaries, only that more examples are included. Probably due to the specific random sample that we have as the training set, it is relatively "easier" or "cheaper" to replicate the examples in the small white disk than those in the small gray disk. This is reflected in both subsets in the left route. On the other hand, the hypotheses associated with the subsets in the right route, also with points $(12, 15)$ and $(16, 11)$, try to mimic the main S-shape boundary and ignore all examples in the two small disks. This gets to an extreme situation with the fourth subset in the right route (point $(23, 7)$), where all examples except the seven ones in the two small disks can be correctly replicated. With higher complexity values, the two routes merge at subsets (points $(23, 7)$ and $(29, 4)$ on the left) that include both examples in the small white disk but also examples around the S-shape boundary. Finally the examples in the small gray disk are also included.

4.4 Discussion

The concept of data decomposition is quite similar to approximating a signal function with partial sums of its Fourier series expansion. Each component in the Fourier series carries information about the original signal function at some frequency level. And the partial sum gives the best approximation to the signal function up to some frequency level. Higher precision can be obtained by adding more series components of higher frequency levels to the partial sum, and in the limit, the sum becomes the signal function itself.

Following this analogy, we would want to "decompose" the full data set $\mathcal{D}$ into a series of "components," denoted as δ_ℓ, which has information about the full set at different complexity levels quantified with ℓ. That is,

$$\mathcal{D} = \biguplus_{\ell=1}^{\infty} \delta_\ell,$$

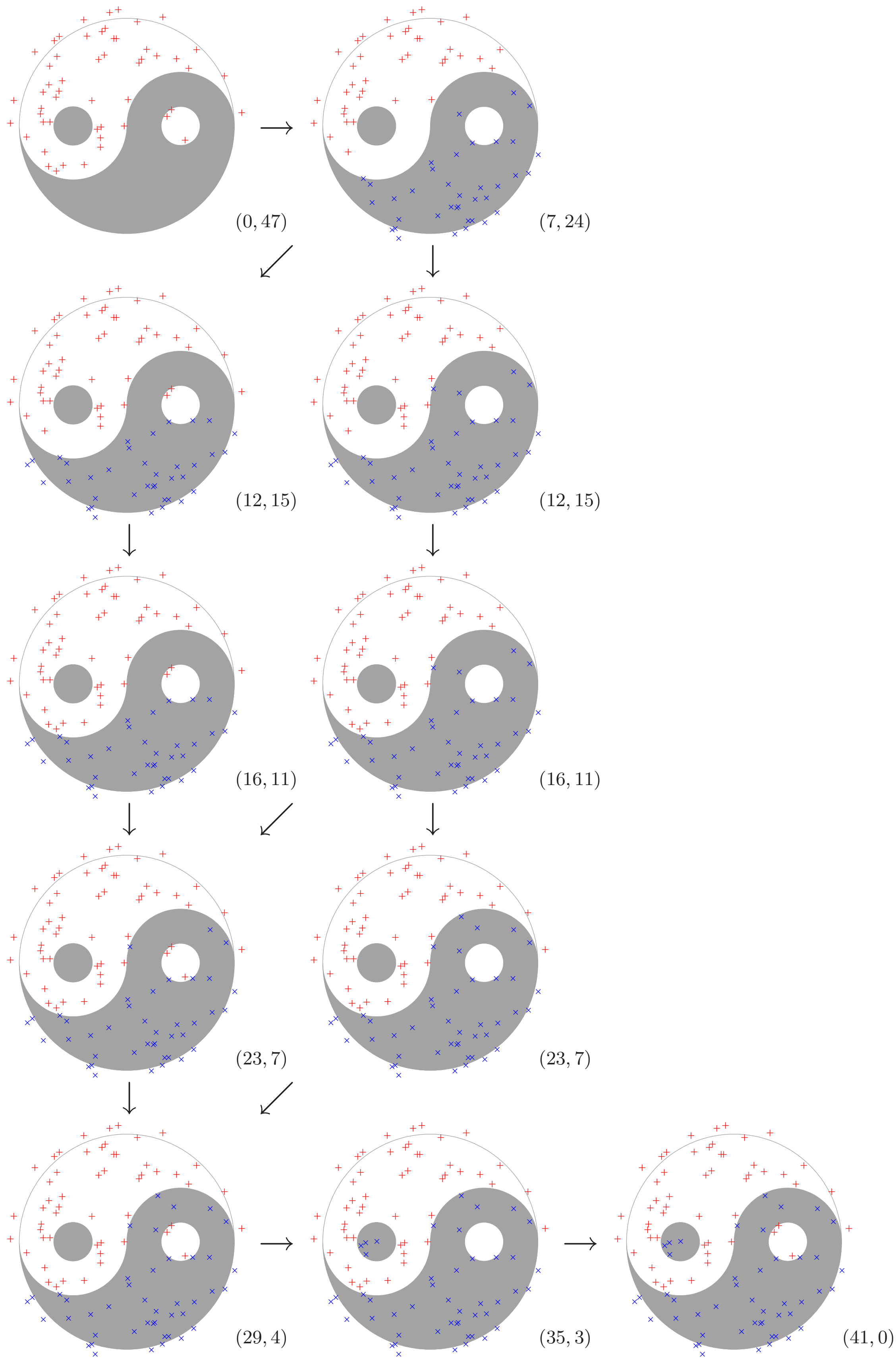

Figure 9. Eleven selected principal subsets of the Yin-Yang problem (complexity-error pairs are also listed)

where $\biguplus$ is some operation to add up the components. Although at this point we are still unclear exactly what the components δ_ℓ are and what the operation $\biguplus$ does, we would expect the partial sum,

$$\mathcal{D}_L = \biguplus_{\ell=1}^{L} \delta_\ell,$$

which should be a subset of $\mathcal{D}$, to be the best approximation to $\mathcal{D}$ within the complexity level L.

Our analysis about the complexity-error pairs of all the subsets concludes that $\mathcal{D}_L$ should be a principal subset. Since a smaller principal subset is not necessarily a subset of a larger principal subset, the decomposition component δ_ℓ is in general not a set of examples. We may consider δ_ℓ as a set of examples that should be added *and* a set of examples that should be removed at the complexity level ℓ.[7] That is, moving from one principal subset to another generally involves adding and removing examples.

This fact also leads to the inevitable high computational complexity of locating principal subsets. It is not straightforward to generate a new principal subset from a known one, and to determine whether a given subset is principal, exponentially many other subsets needed to be checked and compared. This issue will be reexamined in Subsection 5.1. There we will see that the computational complexity is related to the number of the complexity-error paths that contain principal points.

4.5 Related Work

A general consensus is that not all examples in a training set are equivalently important to learning. For instance, examples contaminated by noise are harmful to learning. Even in cases where all the examples are noiseless, there are situations in which we want to deal with examples differently. For instance, in cases where none of the hypotheses can perfectly model the target, it is better to discard examples that cannot be classified correctly by any hypothesis as they may "confuse" the learning (Nicholson, 2002; Li et al., 2005).

There have been many different criteria to discriminate examples as different categories: consistent vs. inconsistent (Brodley and Friedl, 1999; Hodge and Austin, 2004), easy vs. hard (Merler et al., 2004), typical vs. informative (Guyon et al., 1996), etc. Li et al. (2005) unified some of the criteria with the concept of intrinsic margin and group examples as typical, critical, and noisy.

The approach mentioned in this paper takes a quite different view for categorizing examples. There are no natively good or bad examples. Examples are different only because they demand different amount of data complexity for describing them together with other examples. Although we usually think an example is noisy if it demands too much complexity, the amount can be different depending on what other examples are also included in the set. Thus an example that seems noisy at some level of complexity, or with some subset of examples, could be very innocent at another level of complexity or with another subset of examples.

Hammer et al. (2004) studied Pareto optimal patterns in logical analysis of data. The preferences on patterns were deliberately picked, which is quite different from our data complexity measures, so that a Pareto optimal pattern could be found efficiently. Nevertheless, they also favored simpler patterns or patterns that were consistent with more examples. Their experimental results showed that Pareto optimal patterns led to superior learning performance.

[7] To even further generalize the data decomposition concept, we can formulate both the component δ_ℓ and the partial sum $\mathcal{D}_L$ as a set of input-belief pairs, where the belief replaces the original binary output and tells how much we believe what the output should be. However, this more generalized setting is not compatible with our data complexity measures for binary classification data sets, and will not be discussed in this paper.

The so-called function decomposition (Zupan et al., 1997, 2001) is a method that learns the target function in terms of a hierarchy of intermediate hypotheses, which effectively decomposes a learning problem into smaller and simpler subproblems. Although the idea and approaches are quite different from data decomposition, it shares a similar motivation for simple hypotheses. The current methods of function decomposition are usually restricted to problems that have discrete/nominal input features.

5 Data Pruning

Every example in a training set carries its own piece of information about the target function. However, if some examples are corrupted with noise, the information they provide may be misleading. Even in cases where all the examples are noiseless, some examples may form patterns that are too complex for hypotheses in the learning model, and thus may still be detrimental to learning. The process of identifying and removing such outliers and too complex examples from the training set is called *data pruning*. In this section, we apply the data complexity for data pruning.

We have known that given the decomposition of a training set, it is straightforward to select a principal subset according to a complexity budget. However, we also know that, even with a computable practical complexity measure, it is usually prohibitively expensive to find the decomposition. Fortunately, there are ways to approximately identify some principal subsets with affordable computational requirements.

We first show that, with the ideal data complexity measures, outliers or too complex examples can be identified efficiently. Then we show that some more robust methods are needed for practical data complexity measures. Our methods involve a new concept of complexity contribution and a linear regression model for estimating the complexity contributions. The examples with high complexity contributions are deemed as noisy for learning.

5.1 Rightmost Segment

If we start from an empty set, and gradually grow the set by adding examples from a full set $\mathcal{D}$, we observe that the set becomes more and more complex, and reveals more and more details of $\mathcal{D}$, until finally its data complexity reaches $C(\mathcal{D})$. This leads to the definitions of subset paths and complexity-error paths:

Definition 5. *A subset path of set $\mathcal{D}$ is an ordered list of sets $(\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_N)$ where $|\mathcal{D}_n| = n$ and $\mathcal{D}_n \subset \mathcal{D}_{n+1}$ for $0 \le n < N$, and $\mathcal{D}_N = \mathcal{D}$.*

Definition 6. *A complexity-error path of $\mathcal{D}$ is a set of pairs $\{(C(\mathcal{D}_n), |\mathcal{D}| - |\mathcal{D}_n|)\}$ where $0 \le n \le N$ and $(\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_N)$ is a subset path of $\mathcal{D}$.*

Along a subset path, the data complexity increases and the approximation error decreases. So the complexity-error path, if plotted on a 2-D plane, would go down and right, like the one in Figure 10.

Visually, a complexity-error path consists of *segments* of different slopes. Say from $\mathcal{D}_n$ to $\mathcal{D}_{n+1}$, the newly added example can already be replicated by $h_{\mathcal{D}_n}$, one of the shortest hypotheses associated with $\mathcal{D}_n$. This means that no new patterns are necessary to accommodate the newly added example, and $C(\mathcal{D}_{n+1})$ is the same as $C(\mathcal{D}_n)$. Such a case is depicted as those vertical segments in Figure 10. If unfortunately that is not the case, a lookup table entry may be appended and the data complexity goes up by λ. This is shown as the segments of slope $-\lambda^{-1}$. It is also possible that some lookup table entries together can be covered by a pattern with reduced program length, or that the new

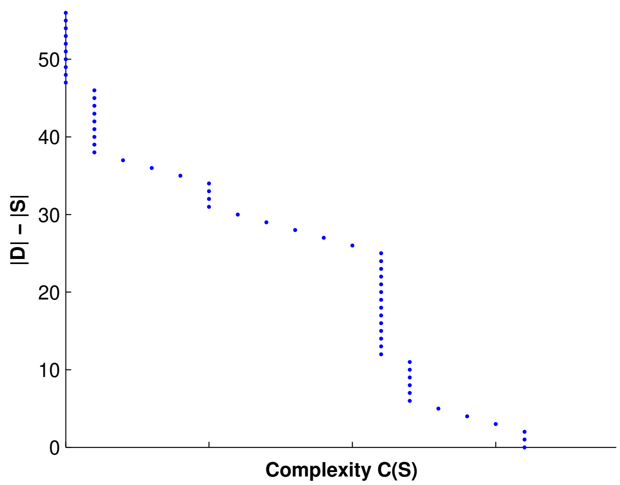

Figure 10. A fictional complexity-error path

example can be accounted for with patterns totally different from those associated with $\mathcal{D}_n$. For both cases, the data complexity increases by an amount less than λ.

A vertical segment is usually "good"—newly added examples agree with the current hypothesis. A segment of slope $-\lambda^{-1}$ is usually "bad"—newly added examples may be outliers according to the current subset since they can only be memorized.

The subset-based complexity-error plot can be regarded as the collection of all complexity-error paths of $\mathcal{D}$. The principal points comprise segments from probably more than one complexity-error path. Due to the mixing of complexity-error paths, a segment of slope $-\lambda^{-1}$ of the principal points may or may not imply that outliers are added. Fortunately, the *rightmost segment*, as defined below, can still help identify outliers.

Definition 7. *Given a data set $\mathcal{D}$, the rightmost segment of the principal points consists of all (c, e) such that (c, e) is a principal point and $c + \lambda e = C(\mathcal{D})$.*

In the following text, we will analyze the properties of the rightmost segment, and show how to use them to identify outliers.

Let's look at any shortest hypothesis for the training set $\mathcal{D}$, $h_{\mathcal{D}}$. Suppose $h_{\mathcal{D}}$ has lookup table entries for a subset $\mathcal{D}_{\rm b}$ of $N_{\rm b}$ examples, and the rest part of $h_{\mathcal{D}}$ can replicate $\mathcal{D}_{\rm g} = \mathcal{D} \backslash \mathcal{D}_{\rm b}$, the subset of $\mathcal{D}$ with just those $N_{\rm b}$ examples removed. Intuitively, examples in $\mathcal{D}_{\rm b}$ are outliers since they are merely memorized, and examples in the pruned set $\mathcal{D}_{\rm g}$ are mostly innocent. The question is, without knowing the shortest hypothesis $h_{\mathcal{D}}$, "can we find out the examples in $\mathcal{D}_{\rm b}$?"

Note that according to the structure of $h_{\mathcal{D}}$, we have $C(\mathcal{D}_{\rm g}) \leq |h_{\mathcal{D}}| - \lambda N_{\rm b}$ and $C(\mathcal{D}) = |h_{\mathcal{D}}|$. Thus $\mathcal{D}_{\rm g}$ is on the rightmost segment, i.e., $C(\mathcal{D}) = C(\mathcal{D}_{\rm g} \cup \mathcal{D}_{\rm b}) = C(\mathcal{D}_{\rm g}) + \lambda N_{\rm b}$, and removing $\mathcal{D}_{\rm b}$ from $\mathcal{D}$ would reduce the data complexity by $\lambda \left|\mathcal{D}_{\rm b}\right|$. This is due to the Lemma 3 below. Furthermore, removing any subset $\mathcal{D}'_{\rm b} \subseteq \mathcal{D}_{\rm b}$ from $\mathcal{D}$ would reduce the data complexity by $\lambda \left|\mathcal{D}'_{\rm b}\right|$.

Lemma 3. *Assume $C(\mathcal{D}_{\rm g} \cup \mathcal{D}_{\rm b}) \geq C(\mathcal{D}_{\rm g}) + \lambda \left|\mathcal{D}_{\rm b}\right|$. We have for any $\mathcal{D}'_{\rm b} \subseteq \mathcal{D}_{\rm b}$, $C(\mathcal{D}_{\rm g} \cup \mathcal{D}'_{\rm b}) = C(\mathcal{D}_{\rm g}) + \lambda \left|\mathcal{D}'_{\rm b}\right|$.*

Proof. From Theorem 3 on page 8, $C(\mathcal{D}_{\rm g} \cup \mathcal{D}'_{\rm b}) \leq C(\mathcal{D}_{\rm g}) + \lambda \left|\mathcal{D}'_{\rm b}\right|$, and

$$C(\mathcal{D}_{\rm g} \cup \mathcal{D}_{\rm b}) = C(\mathcal{D}_{\rm g} \cup \mathcal{D}'_{\rm b} \cup (\mathcal{D}_{\rm b} \backslash \mathcal{D}'_{\rm b})) \leq C(\mathcal{D}_{\rm g} \cup \mathcal{D}'_{\rm b}) + \lambda \left(\left|\mathcal{D}_{\rm b}\right| - \left|\mathcal{D}'_{\rm b}\right|\right).$$

With the assumption $C(\mathcal{D}_{\rm g} \cup \mathcal{D}_{\rm b}) \geq C(\mathcal{D}_{\rm g}) + \lambda \left|\mathcal{D}_{\rm b}\right|$, we have $C(\mathcal{D}_{\rm g} \cup \mathcal{D}'_{\rm b}) \geq C(\mathcal{D}_{\rm g}) + \lambda \left|\mathcal{D}'_{\rm b}\right|$. □

Geometrically, the lemma says that $\mathcal{D} \backslash \mathcal{D}_\mathrm{b}'$ would always be on the rightmost segment if $\mathcal{D}_\mathrm{b}' \subseteq \mathcal{D}_\mathrm{b}$.

This also assures that removing any example $\mathbf{z} \in \mathcal{D}_\mathrm{b}$ from $\mathcal{D}$ would reduce the data complexity by λ. But the inverse is not always true. That is, given an example $\mathbf{z}$ such that $C(\mathcal{D} \backslash \{\mathbf{z}\}) = C(\mathcal{D}) - \lambda$, $\mathbf{z}$ might not be in $\mathcal{D}_\mathrm{b}$. This is because there might be several shortest hypotheses $h_\mathcal{D}$ with different lookup table entries and subsets $\mathcal{D}_\mathrm{b}$. An example in one such $\mathcal{D}_\mathrm{b}$ causes the data complexity to decrease by λ, but may not be in another $\mathcal{D}_\mathrm{b}$. With that said, we can still prove that an example that reduces the data complexity by λ can only be from $\mathcal{D}_\mathrm{b}$ associated with some $h_\mathcal{D}$.

Theorem 5. *Assume $\mathcal{D} = \mathcal{D}_\mathrm{g} \cup \mathcal{D}_\mathrm{b}$, $\mathcal{D}_\mathrm{g} \cap \mathcal{D}_\mathrm{b} = \emptyset$, the following propositions are equivalent:*

1. *There is a shortest hypothesis $h_\mathcal{D}$ for set $\mathcal{D}$ that has lookup table entries for examples in $\mathcal{D}_\mathrm{b}$;*
2. *$C(\mathcal{D}) = C(\mathcal{D}_\mathrm{g}) + \lambda \, |\mathcal{D}_\mathrm{b}|$;*
3. *For any subset $\mathcal{D}_\mathrm{b}' \subseteq \mathcal{D}_\mathrm{b}$, $C(\mathcal{D}_\mathrm{g} \cup \mathcal{D}_\mathrm{b}') = C(\mathcal{D}_\mathrm{g}) + \lambda \, |\mathcal{D}_\mathrm{b}'|$.*

Proof. We have seen $1 \Rightarrow 2$ and $2 \Rightarrow 3$. Here we prove $3 \Rightarrow 1$. Pick any shortest hypothesis $h_{\mathcal{D}_\mathrm{g}}$ for $\mathcal{D}_\mathrm{g}$. For any subset $\mathcal{D}_\mathrm{b}'$, add lookup table entries for examples in $\mathcal{D}_\mathrm{b}'$ to $h_{\mathcal{D}_\mathrm{g}}$ and we get $h_{\mathcal{D}_\mathrm{g} \cup \mathcal{D}_\mathrm{b}'}$, with $\left|h_{\mathcal{D}_\mathrm{g} \cup \mathcal{D}_\mathrm{b}'}\right| = \left|h_{\mathcal{D}_\mathrm{g}}\right| + \lambda \, |\mathcal{D}_\mathrm{b}'| = C(\mathcal{D}_\mathrm{g} \cup \mathcal{D}_\mathrm{b}')$. Thus $h_{\mathcal{D}_\mathrm{g} \cup \mathcal{D}_\mathrm{b}'}$ is a shortest hypothesis for $\mathcal{D}_\mathrm{g} \cup \mathcal{D}_\mathrm{b}'$. We let $\mathcal{D}_\mathrm{b}' = \mathcal{D}_\mathrm{b}$ to get the proposition 1. □

Thus to identify $\mathcal{D}_\mathrm{b}$, we may try all subsets to see which satisfies proposition 2. Alternatively, we can also use a greedy method to remove examples from $\mathcal{D}$ as long as the reduction of the complexity is λ.

5.2 Complexity Contribution

From our analysis of the rightmost segment, removing an example from a training set would reduce the data complexity by some amount, and a large amount of complexity reduction (λ) implies that the removed example is an outlier. For convenience, define the *complexity contribution* of an examples as

Definition 8. *Given a data set $\mathcal{D}$ and an example $\mathbf{z} \in \mathcal{D}$, the complexity contribution of $\mathbf{z}$ to $\mathcal{D}$ is*

$$\gamma_\mathcal{D}(\mathbf{z}) = C(\mathcal{D}) - C(\mathcal{D} \backslash \{\mathbf{z}\}).$$

The greedy method introduced in the last subsection just repeatedly removes examples with complexity contribution equal λ.

However, this strategy does not work with practical data complexity measures. Usually, an approximation of the data complexity is based on a learning model and uses the descriptive length of a hypothesis learned from the training set. It is usually not minimal even within the learning model. In addition, the approximation is also noisy in the sense that data sets of similar data complexity may have quite different approximation values.

For the purpose of a more robust strategy for data pruning, we may look at the complexity contribution of an example to more than one data set. If most of the contributions are high, the example is likely to be an outlier; if most of the contributions are close to zero, the example is probably noiseless. Thus, for instance, we can use the average complexity contribution over different data sets as an indication of the outliers. In general, we can use a linear regression model for robustly estimating the complexity contributions.

Assume that for every training example $\mathbf{z}_n$ there is a real number $\tilde{\gamma}_n$ that is the expected complexity contribution of $\mathbf{z}_n$. To be more formal, we assume that, if a subset $\mathcal{S}$ of $\mathcal{D}$ has $\mathbf{z}_n \in \mathcal{S}$ and $s_1 < |\mathcal{S}| \leq s_2$, we have

$$C(\mathcal{S}) - C(\mathcal{S} \backslash \{\mathbf{z}_n\}) = \tilde{\gamma}_n + \varepsilon,$$

where $0 \le s_1 < s_2 \le N$ are two size constants, and ε is a random variable with mean 0 representing the measure noise. In general, if $\mathcal{S}' \subset \mathcal{S} \subseteq \mathcal{D}$, $s_1 \le |\mathcal{S}'|$, and $|\mathcal{S}| \le s_2$, we assume

$$C(\mathcal{S}) - C(\mathcal{S}') = \sum_{\mathbf{z}_n \in \mathcal{S} \setminus \mathcal{S}'} \tilde{\gamma}_n + \varepsilon.$$

With this assumption, we can set up linear equations of $\tilde{\gamma}_n$'s with different pairs of subsets $\mathcal{S}$ and $\mathcal{S}'$. If we just pick subset pairs in random, we would roughly need two data complexity measurings for each linear equation. To save the number of data complexity measurings, we try to reuse the subsets for setting up the equations. One way is to construct equations with subset paths, as detailed below.

Denote $s = s_2 - s_1$. For an N-permutation $(i_1, i_2, \dots, i_N)$, define $\mathcal{S}_n = \{\mathbf{z}_{i_1}, \mathbf{z}_{i_2}, \dots, \mathbf{z}_{i_n}\}$ for $0 \le n \le N$, which form a subset path $(\mathcal{S}_0, \mathcal{S}_1, \dots, \mathcal{S}_N)$. After getting the data complexity values for the subsets $\mathcal{S}_{s_1}$ until $\mathcal{S}_{s_2}$, we construct s linear equations for $1 \le m \le s$,

$$C(\mathcal{S}_{s_1+m}) - C(\mathcal{S}_{s_1}) = \sum_{n=s_1+1}^{s_1+m} \tilde{\gamma}_{i_n} + \varepsilon.$$

Thus we only need $(s+1)$ data complexity measurings for s linear equations. And if the practical measure supports incremental measuring, such as the number of support vectors in an SVM (Subsection 3.4), we have extra computational savings. With many different N-permutations, we would have many such equations. Let's write all the equations in vector form

$$\mathbf{\Delta} = \mathbf{P}\tilde{\boldsymbol{\gamma}} + \boldsymbol{\varepsilon}, \tag{10}$$

where $\mathbf{\Delta}$ is a column vector of complexity difference between subsets, $\mathbf{P}$ is a matrix and each row of $\mathbf{P}$ is an indication vector of which examples causes the complexity difference, $\tilde{\boldsymbol{\gamma}}$ is a column vector $[\tilde{\gamma}_1, \tilde{\gamma}_2, \dots, \tilde{\gamma}_N]^T$, and $\boldsymbol{\varepsilon}$ is also a column vector of corresponding noise. For instance, the vector form of the s linear equations constructed from the permutation $(1, 2, \dots, N)$ is

$$\begin{bmatrix} C(\mathcal{S}_{s_1+1}) - C(\mathcal{S}_{s_1}) \\ C(\mathcal{S}_{s_1+2}) - C(\mathcal{S}_{s_1}) \\ \vdots \\ C(\mathcal{S}_{s_2}) - C(\mathcal{S}_{s_1}) \end{bmatrix} = \begin{bmatrix} \overbrace{0 \dots 0}^{s_1 \text{ columns}} 1\,0\,0 \dots 0 \overbrace{0 \dots 0}^{(N-s_2) \text{ columns}} \\ 0 \dots 0\,1\,1\,0 \dots 0\,0 \dots 0 \\ \vdots \\ 0 \dots 0\,1\,1\,1 \dots 1\,0 \dots 0 \end{bmatrix} \begin{bmatrix} \tilde{\gamma}_1 \\ \tilde{\gamma}_2 \\ \vdots \\ \tilde{\gamma}_N \end{bmatrix} + \boldsymbol{\varepsilon}. \tag{11}$$

For a different permutation, the columns of the indication matrix shuffle according to the permutation, and the rest is pretty much the same. The actual vector $\mathbf{\Delta}$ and matrix $\mathbf{P}$ contain many block matrices from different permutations.

If we further assume some joint distribution of the measure noise ε, we may locate the optimal $\tilde{\gamma}_n$ for these equations. For example, if we assume the noise is normally distributed with $\mathbf{\Sigma} = \mathrm{E}\left[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T\right]$ as the covariance matrix, the best linear unbiased estimator for (10) is

$$\tilde{\boldsymbol{\gamma}} = \left(\mathbf{P}^T \mathbf{\Sigma}^{-1} \mathbf{P}\right)^{-1} \mathbf{P}^T \mathbf{\Sigma}^{-1} \mathbf{\Delta}. \tag{12}$$

Notice that for a specific covariance matrix, the estimate is as simple as the average complexity contribution:

Theorem 6. *Assume the measure noise is independent across different permutations, and has covariance matrix*

$$\mathbf{\Sigma}_1 = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & 2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2 & \cdots & s \end{bmatrix} \sigma^2$$

within equations of a same permutation. Thus the covariance matrix $\mathbf{\Sigma}$ for all the equations would be a block diagonal matrix with diagonal blocks being $\mathbf{\Sigma}_1$. The best linear unbiased estimator (12) is actually

$$\tilde{\gamma}_n = \mathrm{E}\left[C(\mathcal{S}_{i+1}) - C(\mathcal{S}_i) \mid s_1 \leq |\mathcal{S}_i| < s_2\right], \tag{13}$$

where, for a particular permutation, $\mathcal{S}_i$ is the largest in the subset path generated by the permutation that does not include $\mathbf{z}_n$, and the expectation is over all the permutations used in constructing the equations such that $\mathcal{S}_i$ has the proper size.

Proof. Let's first focus on equations constructed from a same permutation. Without loss of generality, we take those in (11) for our proof. Define an s-by-s square matrix

$$\mathbf{A}_1 = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix} \sigma^{-1}.$$

Left-multiplying both sides of (11) with $\mathbf{A}_1$, we get

$$\begin{bmatrix} C(\mathcal{S}_{s_1+1}) - C(\mathcal{S}_{s_1}) \\ C(\mathcal{S}_{s_1+2}) - C(\mathcal{S}_{s_1+1}) \\ \vdots \\ C(\mathcal{S}_{s_2}) - C(\mathcal{S}_{s_2-1}) \end{bmatrix} = \begin{bmatrix} \overbrace{0 \ldots 0}^{s_1 \text{ columns}}\, 1\, 0\, 0 \ldots 0\, \overbrace{0 \ldots 0}^{(N-s_2) \text{ columns}} \\ 0 \ldots 0\, 0\, 1\, 0 \ldots 0\, 0 \ldots 0 \\ \vdots \\ 0 \ldots 0\, 0\, 0 \ldots 0\, 1\, 0 \ldots 0 \end{bmatrix} \begin{bmatrix} \tilde{\gamma}_1 \\ \tilde{\gamma}_2 \\ \vdots \\ \tilde{\gamma}_N \end{bmatrix} + \mathbf{A}_1 \boldsymbol{\varepsilon}.$$

It is easily verified that the noise covariance, $\mathbf{A}_1\mathbf{\Sigma}_1\mathbf{A}_1^T$, is now the identity matrix. The optimal solution for this permutation only would be $\tilde{\gamma}_{s_1+m} = C(\mathcal{S}_{s_1+m}) - C(\mathcal{S}_{s_1+m-1})$ for $1 \leq m \leq s$. Consider a block diagonal matrix $\mathbf{A}$ that has as many diagonal blocks as the permutations, and each diagonal block is $\mathbf{A}_1$. Left-multiplying both sides of (10) with $\mathbf{A}$ transforms all the equations into some form of

$$C(\mathcal{S}_{i+1}) - C(\mathcal{S}_i) = \tilde{\gamma}_n + \varepsilon',$$

with $\mathcal{S}_{i+1} = \mathcal{S}_i \cup \{\mathbf{z}_n\}$ and the noise covariance matrix being the identity matrix. Thus the optimal linear solution would be $\tilde{\gamma}_n$ equal the average of such complexity contributions. Since our construction only allows $\mathcal{S}_i$ to have a size between s_1 and s_2, we have the estimator (13). □

Such assumption about the noise covariance matrix is somewhere between a uniform assumption and a fully-correlated assumption. The uniform assumption regards the noise in each equation as independent and of the same magnitude. The fully-correlated assumption fine-tunes the model to assume that, associated with each $\tilde{\gamma}_n$, there is a random noise variable with mean 0 and variance σ^2, and the noise of the equation is the sum of the random noise variables associated with $\tilde{\gamma}_n$'s in the equation. The noise covariance of two equations would then be proportional to the number of common $\tilde{\gamma}_n$'s in these two equations. That is, $\mathbf{\Sigma} = \mathbf{P}\mathbf{P}^T\sigma^2$. However, since usually there are more equations than unknown variables, $\mathbf{\Sigma}$ is singular, which gives trouble in solving the equations via (12). Although the actual noise covariance would depend many factors including the practical complexity measure, it happens that the assumption that leads to average complexity contribution (13) works well in practice, as we will see in the next subsection.

5.3 Experiments

We test with the Yin-Yang target the concept of complexity contribution and the methods for estimating the complexity contribution. The experimental settings are similar to those used by Li et al. (2005). That is, a data set of size 400 is randomly generated, and the outputs of 40 examples (the last 10% indices) are further flipped as injected outliers.

We first verify that outliers would have higher complexity contributions on average than noiseless examples. To do so, we pick a random subset path and compute the complexity increase along the path. This is repeated many times and the complexity increase is averaged. Figure 11 shows such average complexity contribution of noisy and noiseless examples versus the subset size. Here are

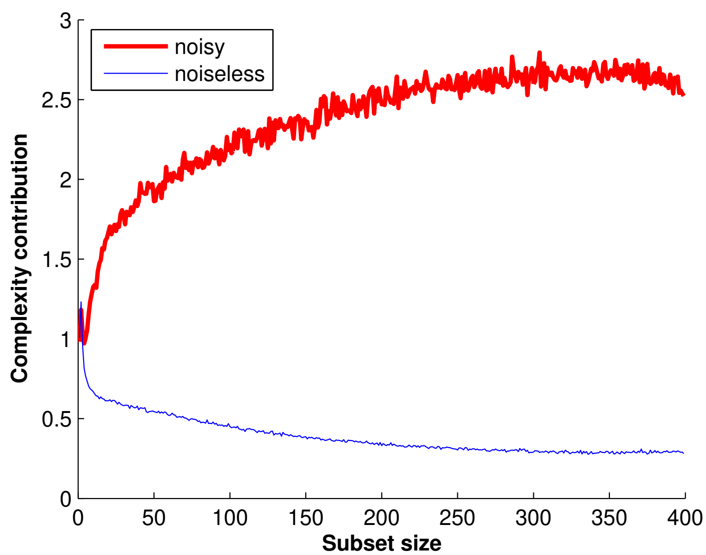

Figure 11. Average complexity contribution of the noisy and the noiseless examples in the Yin-Yang data

some observations:

- Overall, the 40 noisy examples have apparently much higher average complexity contribution than the 360 noiseless ones.
- When the subset size is really small (≤ 6), the noisy and the noiseless examples are indistinguishable with respect to complexity contribution. There has to be more information about the target in order to tell which examples are noisy and which are not.
- The average contribution of the noiseless examples becomes smaller when the subset gets larger. This is because when the subset has more details about the target function, newly added noiseless examples would have less chance to increase the data complexity.
- The average contribution of the noisy examples becomes larger when the subset gets larger, but it also seems converging to some value around 2.5. The reason of the contribution increase is related to that of the contribution decrease of the noiseless examples. When there is more

correct information about the target function, an outlier would cause more complexity increase than it would when there is less information.

- The two contribution curves have noticeable bends at the right ends. These are artifacts due to the lack of distinct subsets when the subset size is close to the full size.

One way to use the complexity contribution for data pruning is to set a threshold θ and claim any example $\mathbf{z}_n$ noisy if $\tilde{\gamma}_n \geq \theta$. For the uniform variance assumption and the assumption leading to the average complexity contribution, which were discussed in Subsection 5.2, we set $s_1 = 50$, $s_2 = 400$, and solve equations created from 200,000 random permutations. We plot their receiver operating characteristic (ROC) in Figure 12. The ROC summarizes how the false negative rate (portion of noisy examples being claimed as noiseless) changes with the false positive rate (portion of noiseless examples being claimed as noisy) as the threshold θ varies. Both methods achieve large area under the ROC curve (AUC), a criterion to compare different ROC curves.

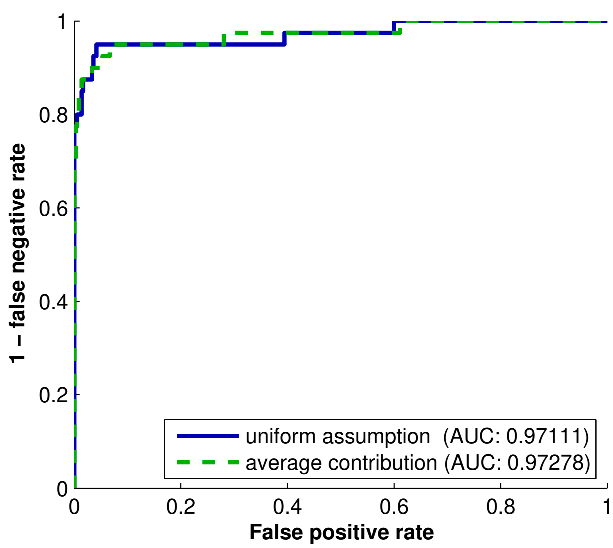

Figure 12. ROC curves of two estimators on the Yin-Yang data

Similar to the data categorization proposed by Li et al. (2005), we also group all the training examples into three categories: typical, critical, and noisy. With two ad hoc thresholds $\theta_1 = \frac{2}{3}$ and $\theta_2 = 1$, the typical examples are those with $\tilde{\gamma}_n < \theta_1$, and we hope they are actually noiseless and far from the class boundary; the noisy examples are those with $\tilde{\gamma}_n > \theta_2$, and we hope they are actually outliers; the critical examples are those have $\tilde{\gamma}_n$ between θ_1 and θ_2, and we hope they are close to the class boundary. Figure 13 is the fingerprint plot which visually shows the categorization. The examples are positioned according to their signed distance to the decision boundary on the vertical axis and their index n in the training set on the horizontal axis. Critical and noisy examples are shown as empty circles and filled squares, respectively. We can see that most of the outliers (last 10% of the examples) are categorized as noisy, and some examples around the zero distance value are categorized as critical. We also have some imperfections—some of the critical examples are categorized as outliers and vice versa.

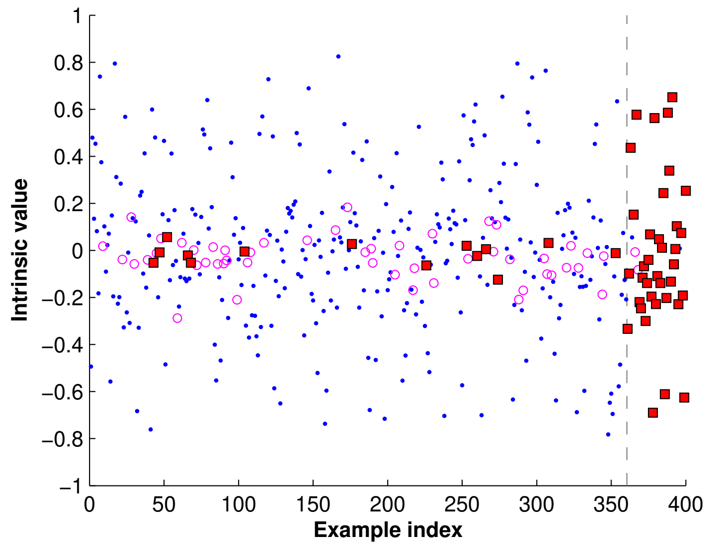

Figure 13. Fingerprint plot of the Yin-Yang data with the average contribution: · typical; ○ critical; ■ noisy

5.4 Discussion

With the estimated complexity contribution $\tilde{\gamma}_n$, we hope that a noiseless example would have a relatively low contribution and an outlier would have a relatively high contribution. However, whether an outlier can be distinguished using the complexity contribution heavily depends on the choice of the subsets used in the linear equations as well as many other factors.

For instance, the smaller Yin-Yang data set (Figure 7 on page 15) contains several examples in the small gray disk. It is reflected in the selected principal subsets (Figure 9 on page 17, also Figure 16 on page 32) that those examples are included in principal subsets of only relatively high data complexity (≥ 35). Since those examples constitute a small percentage of the training set, a random subset of size smaller than, say, half of the full training set size, has a small probability to have most of those examples, and would usually only have patterns shown in principal subsets with data complexity lower than 35. Thus those examples would have large complexity contributions with high probability. For similar reasons, if some outliers happen to be in the small gray disk of the Yin-Yang target, with small subsets they may be regarded as innocent.

It is still unclear what conditions can assure that an outlier would have a high average complexity contribution.

5.5 Related Work

The complexity contribution quantifies to what degree an example may affect learning with respect to the data complexity. It is similar to the information gain concept behind informative examples used by Guyon et al. (1996) for outlier detection.

Some other outlier detection methods also exploit practical data complexity measures. Gamberger and Lavrač (1997) used a saturation test which is quite similar to our greedy method. Arning et al. (1996) looked for the greatest reduction in complexity by removing a subset of examples, which can be approximately explained by the proposition 2 in Theorem 5.

Statistics community has studied outlier detection extensively (Barnett and Lewis, 1994). They usually assume an underlying statistical model and define outliers based on discordance tests, many of which can be described as some simple distance-based check (Knorr and Ng, 1997).

Learning algorithms can also be used for data pruning. For example, Angelova et al. (2005) combined many classifiers with naive Bayes learning for identifying troublesome examples. Some learning algorithms, such as boosting, can produce information about the hardness of an example as a byproduct, which can be used for outlier detection (Merler et al., 2004; Li et al., 2005).

Angiulli et al. (2004) encoded background knowledge in the form of a first-order logic theory and outliers were defined as examples for which no logical justification can be found in the theory. They also showed that such outlier detection was intrinsically intractable due to its high computational complexity.

6 Conclusion

We have defined three ideal complexity measures for a data set. The universal data complexity is the length of the shortest program that can replicate the data set. The data complexity for a learning model finds a consistent hypothesis with the shortest encoding. And the data complexity with a lookup table also takes hypothesis errors into consideration. All these complexity measures are closely related to learning principles such as Occam's razor.

We have demonstrated the usage of the data complexity in two machine learning problems, data decomposition and data pruning. In data decomposition, we have illustrated that the principal subsets best approximate the full data set; in data pruning, we have proved that outliers are examples with high complexity contributions. We have also proposed and tested methods for estimating the complexity contribution.

Underneath the concept and the applications of the data complexity is the desire for generalization, the central issue of machine learning. Theoretically, if the correct prior and the exact noise model are known, we can encode the hypotheses and the errors in a way such that the shortest hypothesis generalizes the best. If the prior is unknown, the universal prior is a good guess for any computable priors, and the shortest hypothesis would still have high chance to generalize well. In practice, we also make a reasonable guess on the prior since practical approximations are usually designed with Occam's razor in mind.

Many approaches in this paper require intensive computational efforts, which is inevitable when the shortest hypothesis is sought. However, for practical applications, more computationally feasible solutions should be studied.

Acknowledgments

We wish to thank Xin Yu, Amrit Pratap, and Hsuan-Tien Lin for great suggestions and helpful comments. This work was partially supported by the Caltech SISL Graduate Fellowship.

References

Abu-Mostafa, Y. S. (1988a). Complexity of random problems. In Abu-Mostafa, Y. S., editor, *Complexity in Information Theory*, pages 115–131. Springer-Verlag.

Abu-Mostafa, Y. S. (1988b). Random problems. *Journal of Complexity*, 4(4):277–284.

Abu-Mostafa, Y. S. (1995). Hints. *Neural Computation*, 7(4):639–671.

Aizerman, M., Braverman, E., and Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25(6):821–837.

Angelova, A., Abu-Mostafa, Y., and Perona, P. (2005). Pruning training sets for learning of object categories. In Schmid, C., Soatto, S., and Tomasi, C., editors, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, volume 1, pages 494–501.

Angiulli, F., Greco, G., and Palopoli, L. (2004). Detecting outliers via logical theories and its data complexity. In Suzuki, E. and Arikawa, S., editors, *Discovery Science: 7th International Conference, DS 2004*, volume 3245 of *Lecture Notes in Artificial Intelligence*, pages 101–113. Springer-Verlag.

Arning, A., Agrawal, R., and Raghavan, P. (1996). A linear method for deviation detection in large databases. In Simoudis, E., Han, J., and Fayyad, U., editors, *Proceedings of the Second International Conference on Knowledge Discovery & Data Mining*, pages 164–169. AAAI Press.

Barnett, V. and Lewis, T. (1994). *Outliers in Statistical Data*. John Wiley & Sons, 3rd edition.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1987). Occam's razor. *Information Processing Letters*, 24(6):377–380.

Brodley, C. E. and Friedl, M. A. (1999). Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167.

Cauwenberghs, G. and Poggio, T. (2001). Incremental and decremental support vector machine learning. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 409–415. MIT Press.

Gamberger, D. and Lavrač, N. (1997). Conditions for Occam's razor applicability and noise elimination. In van Someren, M. and Widmer, G., editors, *Machine Learning: ECML-97*, volume 1224 of *Lecture Notes in Artificial Intelligence*, pages 108–123. Springer-Verlag.

Grünwald, P. (2005). Minimum description length tutorial. In Grünwald, P. D., Myung, I. J., and Pitt, M. A., editors, *Advances in Minimum Description Length: Theory and Applications*, chapter 2, pages 23–80. MIT Press.

Guyon, I., Matić, N., and Vapnik, V. (1996). Discovering informative patterns and data cleaning. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, pages 181–203. AAAI Press/MIT Press.

Hammer, P. L., Kogan, A., Simeone, B., and Szedmák, S. (2004). Pareto-optimal patterns in logical analysis of data. *Discrete Applied Mathematics*, 144(1–2):79–102.

Hodge, V. J. and Austin, J. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126.

Knorr, E. M. and Ng, R. T. (1997). A unified notion of outliers: Properties and computation. In Heckerman, D., Mannila, H., Pregibon, D., and Uthurusamy, R., editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 219–222. AAAI Press.

Levin, L. A. (1973). Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266.

Li, L., Pratap, A., Lin, H.-T., and Abu-Mostafa, Y. S. (2005). Improving generalization by data categorization. In Jorge, A., Torgo, L., Brazdil, P., Camacho, R., and Gama, J., editors, *Knowledge Discovery in Databases: PKDD 2005*, volume 3721 of *Lecture Notes in Artificial Intelligence*, pages 157–168. Springer-Verlag.

Li, M. and Vitányi, P. (1997). *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, 2nd edition.

Lin, H.-T. and Li, L. (2005a). Infinite ensemble learning with support vector machines. In Gama, J., Camacho, R., Brazdil, P., Jorge, A., and Torgo, L., editors, *Machine Learning: ECML 2005*, volume 3720 of *Lecture Notes in Artificial Intelligence*, pages 242–254. Springer-Verlag.

Lin, H.-T. and Li, L. (2005b). Novel distance-based SVM kernels for infinite ensemble learning. In *Proceedings of the 12th International Conference on Neural Information Processing*, pages 761–766.

Merler, S., Caprile, B., and Furlanello, C. (2004). Bias-variance control via hard points shaving. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(5):891–903.

Nicholson, A. (2002). *Generalization Error Estimates and Training Data Valuation*. PhD thesis, California Institute of Technology.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5):465–471.

Schmidhuber, J. (1997). Discovering neural nets with low Kolmogorov complexity and high generalization capability. *Neural Networks*, 10(5):857–873.

Solomonoff, R. J. (2003). The universal distribution and machine learning. *The Computer Journal*, 46(6):598–601.

Vapnik, V. N. (1999). *The Nature of Statistical Learning Theory*. Springer-Verlag, 2nd edition.

Webb, G. I. (1996). Further experimental evidence against the utility of Occam's razor. *Journal of Artificial Intelligence Research*, 4:397–417.

Wolpert, D. H. and Macready, W. G. (1999). Self-dissimilarity: An empirically observable complexity measure. In Bar-Yam, Y., editor, *Unifying Themes in Complex Systems*, pages 626–643. Perseus Books.

Zupan, B., Bohanec, M., Bratko, I., and Demšar, J. (1997). Machine learning by function decomposition. In Douglas H. Fisher, J., editor, *Machine Learning: Proceedings of the Fourteenth International Conference (ICML '97)*, pages 421–429. Morgan Kaufmann.

Zupan, B., Bratko, I., Bohanec, M., and Demšar, J. (2001). Function decomposition in machine learning. In Paliouras, G., Karkaletsis, V., and Spyropoulos, C. D., editors, *Machine Learning and Its Applications: Advanced Lectures*, volume 2049 of *Lecture Notes in Artificial Intelligence*, pages 71–101. Springer-Verlag.

A Principal Sets

Here we collect all the principal subsets of the concentric disks problem (Figure 14) and almost all the principal subsets of the Yin-Yang problem (Figures 15 and 16), which are described in Subsection 4.3. The corresponding complexity-error plots can be found in Figure 5 on page 14 and Figure 8 on page 16. For the Yin-Yang problem, each of the principal points $(11, 16)$, $(15, 12)$, and $(16, 11)$ has 4 or more associated principal subsets, but only two are shown for space reason. The number of support vectors in SVM-perceptron is the practical data complexity measure used.

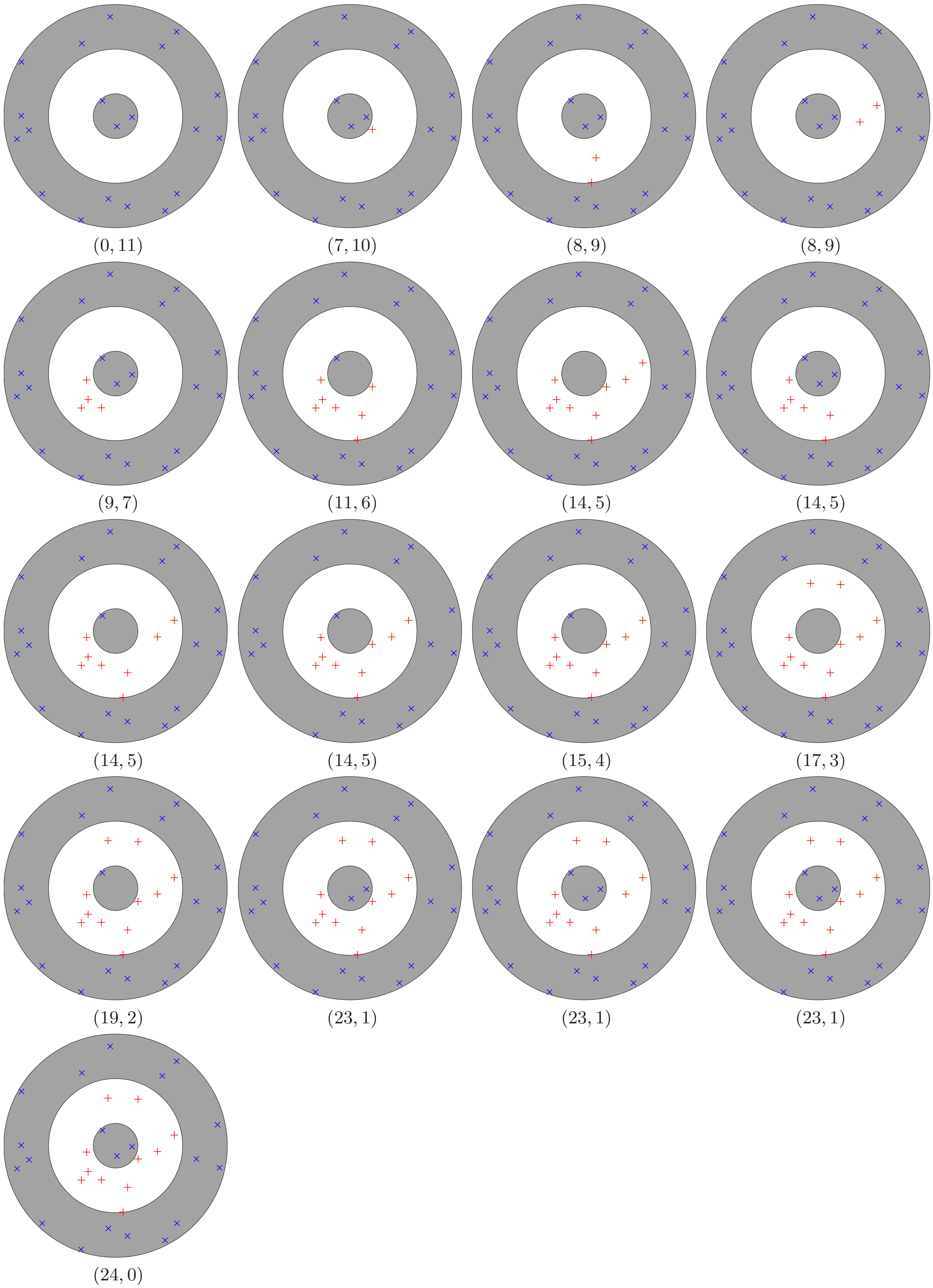

Figure 14. All principal subsets of the concentric disks problem (complexity-error pairs are also listed)

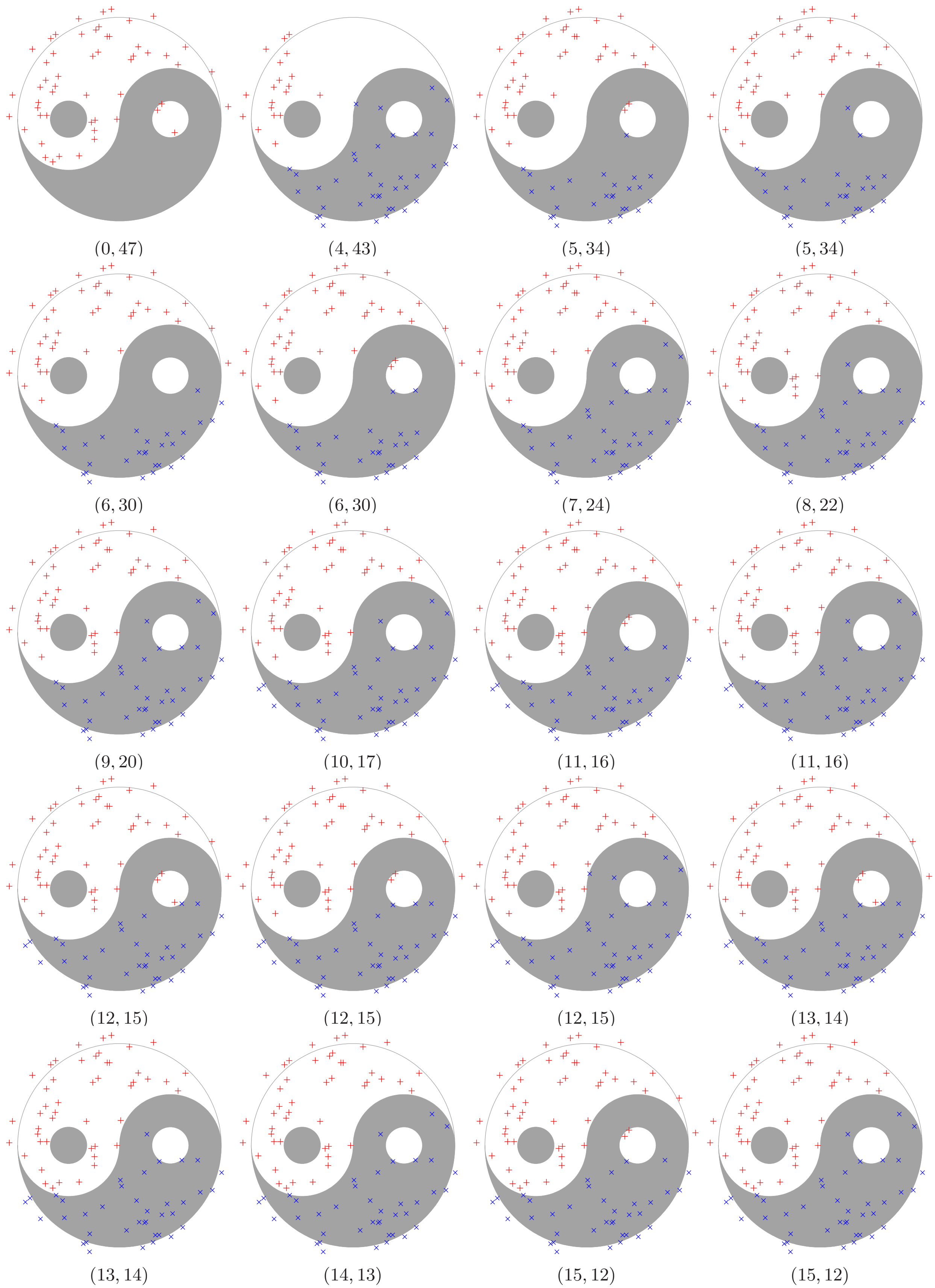

Figure 15. Principal subsets of the Yin-Yang data, part I

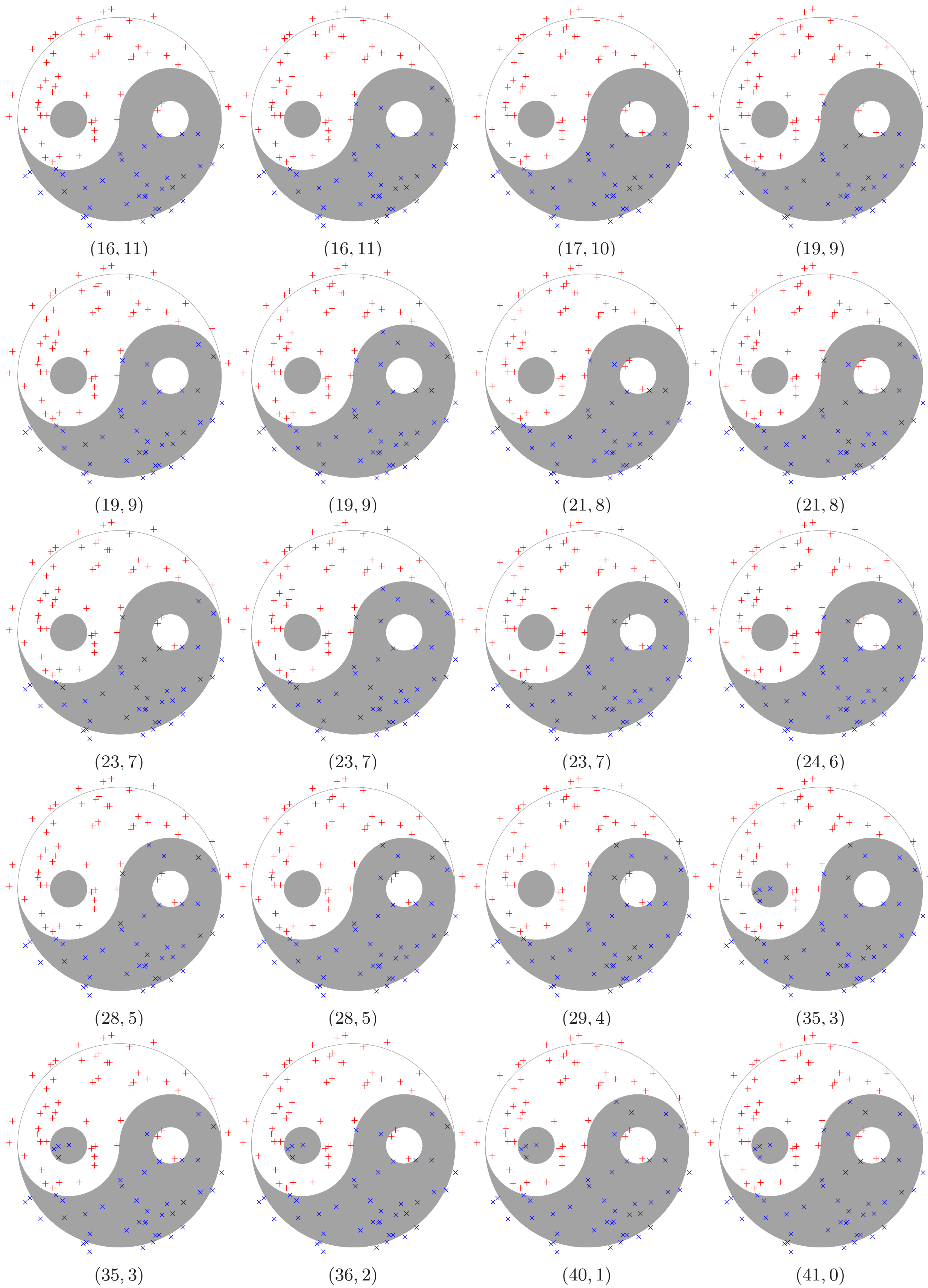

Figure 16. Principal subsets of the Yin-Yang data, part II