Compression and Machine Learning: A New Perspective on Feature Space Vectors

D. Sculley and Carla E. Brodley {dsculley, brodley}@cs.tufts.edu
Department of Computer Science, Tufts University, Medford, MA 02155, USA

Abstract

The use of compression algorithms in machine learning tasks such as clustering and classification has appeared in a variety of fields, sometimes with the promise of reducing problems of explicit feature selection. The theoretical justification for such methods has been founded on an upper bound on Kolmogorov complexity and an idealized information space. An alternate view shows compression algorithms implicitly map strings into implicit feature space vectors, and compression-based similarity measures compute similarity within these feature spaces. Thus, compression-based methods are not a "parameter free" magic bullet for feature selection and data representation, but are instead concrete similarity measures within defined feature spaces, and are therefore akin to explicit feature vector models used in standard machine learning algorithms. To underscore this point, we find theoretical and empirical connections between traditional machine learning vector models and compression, encouraging cross-fertilization in future work.

1. Introduction: Re-Inventing the Razor

The fundamental idea that data compression can be used to perform machine learning tasks has surfaced in a several areas of research, including data compression (Witten et al., 1999a; Frank et al., 2000), machine learning and data mining (Cilibrasi and Vitanyi, 2005; Keogh et al., 2004; Chen et al., 2004), information theory, (Li et al., 2004), bioinformatics (Chen et al., 1999; Hagenauer et al., 2004), spam filtering (Bratko and Filipic, 2005), and even physics (Benedetto et al., 2002). The principle at work is that if strings x and y compress more effectively together than they do apart, then they must share similar information.

This powerful insight has potential application in any field that applies machine learning techniques to strings, but is not especially new. Primacy may lie with Ian Witten in the mid-1980's (Frank et al., 2000), but can perhaps be traced back more deeply, all the way to William of Ockham and his now famous razor of the 14th century. Yet although the basic idea is established, specific approaches are still evolving. This paper seeks to contribute a deeper understanding of the implicit feature spaces used by compression-based machine learning methods. The heady dream of "parameter-free data mining" (Keogh et al., 2004) encouraged by an idealized information space under Kolmogorov complexity is not realized by compression-based methods. Instead, each compression algorithm operates within a concrete feature space, and compression-based measures calculate similarity between vectors in that space. We find direct connections between compression and established feature space models, such as TF*IDF and n-gram vector methods, illustrated by experiment. Our hope is to invite increased dialog between the compression and learning communities.

We focus this paper on the various *compression-based similarity measures* that have been proposed and applied (Chen et al., 2004; Li et al., 2004; Keogh et al., 2004), while noting that other work in this area has included an entropy-based measure (Benedetto et al., 2002), and discriminant functions analogous to the Fisher linear discriminant (Frank et al., 2000; Bratko and Filipic, 2005). Analyzing similarity measures allows us to isolate the effects of specific feature models from those of similarity functions and learning algorithms, allowing a true apples-to-apples comparison.

The remainder of this paper is organized as follows. In Section 2, we review the various compression-based methods that have appeared in the literature, summarize the Kolmogorov complexity arguments, and introduce the need for a feature space model of compression. Section 3 details the mapping from string to feature space vectors, examines similarity measures in vector space, and then shows how explicit feature vector models may be linked back to compression. We report a supporting experiment on Unix user classification in Section 4. Concluding in Section 5, we discuss of the limitations and insights gained from compression-based similarity measures.

2. Compression-Based Similarity Measures

Much previous work has focused on the formulation and application of compression-based similarity metrics and measures.[1] In this section, we review four compression-based similarity measures which will serve as the foundation for our later discussion.

It will be helpful to define some notation and conventions. We will use $|x|$ to denote the number of symbols in string x. $C(x)$ gives the length of string x after it has been compressed by compression algorithm $C(\cdot)$ measured as a number of bits (most often rounded up to whole bytes). The concatenation of strings x and y is written as xy; thus, $C(xy)$ gives the number of bits needed to compress x and y together. The term $C(x|y)$ shows the length of x when *conditionally compressed* with a model built on string y. While some researchers have implemented specialized conditional compression algorithms (Chen et al., 1999), the approximation $C(x|y) = C(yx) - C(y)$ accommodates the use of off-the-shelf compressors (Li et al., 2004).[2]

NCD: the Normalized Compression Distance. Li et al. (2004) define the Normalized Compression Distance (NCD) as follows:

$$NCD(x,y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}$$

This metric has the advantage of "minorizing in an appropriate sense every effective metric" (Li et al., 2004), which means that when NCD says that two strings are strongly related, they very likely are. Cilibrasi and Vitanyi (2005) show that NCD is a formal distance metric within certain tolerances by defining bounds on what they called a *normal compressor*, which include the requirement that $C(x) = C(xx)$ within logarithmic bounds. When these bounds are met, NCD operates in the range $[0, 1+\epsilon]$, where 0 shows x and y are identical, and 1 shows they are completely dissimilar. Here, ϵ is a small error term on the order of .1 (Li et al., 2004). Although some standard compression algorithms such as $LZ77$, $LZ78$, and even PPM, are not guaranteed to satisfy these bounds, NCD has been successfully applied to a host of clustering applications (Cilibrasi and Vitanyi, 2005; Cilibrasi et al., 2004; Li et al., 2004).

CLM: the Chen-Li Metric. Another metric has been defined and used by Chen, Li, and their collaborators (Chen et al., 1999; Li et al., 2001; Chen et al., 2004).[3] Their formulation reads:

$$CLM(x,y) = 1 - \frac{C(x) - C(x|y)}{C(xy)}$$

1. Recall that there is a technical distinction between the terms *metric* and a *measure*: metrics satisfy formal requirements on identity, symmetry, and triangle inequality. We refer to the class as compression-based similarity measures, which includes some measures that satisfy the metric definition within certain bounds.
2. Li et al. (2004) proposed this approximation in light of Kolmogorov complexity; we provide additional support with the following probabilistic argument. An effective compressor will approach the bound from information theory, $C(x) = -\log P(x)$, and within small bounds $C(x|y) = -\log P(x|y)$. Basic probability defines $P(x|y) = \frac{P(x \cap y)}{P(y)}$. Thus, $C(x|y) = -\log \frac{P(y \cap x)}{P(y)} = -\log P(y \cap x) - \log P(y) = C(yx) - C(y)$.
3. These authors did not give their metric an explicit name in the literature. We thus take the liberty of referring to it here as CLM, the Chen-Li Metric.

The term $C(x) - C(x|y)$ gives an upper bound on *mutual algorithmic information* between strings (Li et al., 2001), a measure of shared information defined in terms of Kolmogorov complexity. Like NCD, this metric is normalized to the range $[0, 1]$, with 0 showing complete similarity and 1 showing complete dissimilarity. CLM has achieved empirical success in several important applications, including genomic sequence comparison and plagiarism detection (Chen et al., 1999; Li et al., 2001; Chen et al., 2004).

CDM: the Compression-based Dissimilarity Measure. Keogh et al. (2004) set forth their Compression-based Dissimilarity Measure (CDM) in response to NCD, calling it a "simpler measure," but avoiding theoretical analysis.

$$CDM(x, y) = \frac{C(xy)}{C(x) + C(y)}$$

These authors are aware that CDM is non-metric, failing the identity property. CDM gives values in the range of $[\frac{1}{2}, 1]$, where $\frac{1}{2}$ shows pure identity and 1 shows pure disparity. But although CDM was proposed without theoretical analysis, it was used to produce successful results in clustering and anomaly detection (Keogh et al., 2004).

CosS: Compression-based Cosine. Meanwhile, we have designed a compression-based measure, $CosS$, based on the familiar cosine-vector dissimilarity measure:

$$CosS(x, y) = 1 - \frac{C(x) + C(y) - C(xy)}{\sqrt{C(x)C(y)}}$$

This measure is normalized to the range $[0, 1]$, with 0 showing total identity between the strings, and 1 total dissimilarity.

Although the formulae of each of these four measures appears to be quite distinct from the others, we will show in Section 3.2 that the only actual differences among them are in the normalizing terms.

Beyond Kolmogorov Complexity Two of the above metrics, NCD and CLM, were first developed in terms of Kolmogorov complexity[4] measuring distance in idealized information space. However, because Kolmogorov complexity is uncomputable (Li and Vitanyi, 1997), compression algorithms are employed to approximate an upper bound on $K(x)$. This bound may be very loose. Trivial examples include compressing numbers generated by a pseudo-random number generator, or compressing the non-terminating non-repeating digits of π. In both cases, $C(x) \gg K(x)$ (Li and Vitanyi, 1997; Grünwald, 2005). And in general, we cannot know how close the compression bound $C(x)$ is to $K(x)$. Thus, it makes sense to analyze compression-based similarity metrics not within the context of an idealized information space, which is unattainable, but within the concrete feature spaces employed by actual compression algorithms.

3. Compression and Feature Space

In this section, we will examine the compression-based mapping from strings to vectors in feature space, explore the similarity functions computed in that space, and finish by turning the tables to show the connection from existing explicit feature vector models back to compression.

4. Kolmogorov complexity, $K(x)$, is a theoretical measure of the amount of information in string x, and is defined as the length of the shortest computer program generating x. See the comprehensive text by Li and Vitanyi (1997).

3.1 Compression and Feature Vectors

Here, we show that compression-based similarity measures operate within specific feature spaces of high dimension. We define two requirements for this to be true. First, for each compressor $C(\cdot)$ we define an associated vector space $\aleph$, such that $C(\cdot)$ maps an input string x into a vector $\vec{x} \in \aleph$. Second, we must show that the value $C(x)$, the length of the compressed string x, corresponds to a vector norm $||\vec{x}||$. An exhaustive examination of the feature spaces underlying all compression algorithms is precluded by space; instead, we choose to examine three representative lossless compression methods, LZW, $LZ77$, and PPM.[5] At the end of this section, we discuss the connection between string concatenation and vector addition, as the addition operator will be useful when we take apart the formulae of the similarity measures to examine their effect in the implied feature spaces.

$LZ77$ Feature Space. The $LZ77$ algorithm, prototypical of one branch of the Lempel-Ziv family of compressors, encodes substrings as a series of output codes that reference previously occurring substrings in a sliding dictionary window. Although parameter values vary by implementation, we may assume that the repeated substrings found by $LZ77$ are of maximum length m, and that the dictionary window is of length $p \geq m$. Furthermore, we will for simplicity assume an implementation of $LZ77$ in which the output codes are of constant length c.

The feature space $\aleph$, then, is a high dimensional space in which each coordinate corresponds to one of the possible substrings of up to length m. The compressor implicitly maps string x to $\vec{x} \in \aleph$ as follows. Initially, each element $\vec{x}_i = 0$. As $LZ77$ passes over the string and produces output codes, each new output code referring to substring i causes the update $\vec{x}_i := \vec{x}_i + c$. At the end of the process, the implicit $\vec{x}$ is a vector modeling x. Furthermore, as all $\vec{x}_i$ are non-negative, $C(x)$ yields the 1-norm of $\vec{x}$, written $||\vec{x}||_1 = \sum_i |\vec{x}_i|$, which is sometimes referred to as the *city block distance*.[6] Note that although the $LZ77$ feature space $\aleph$ is very large, with $O(2^m)$ dimensions, mapping from x to $\vec{x} \in \aleph$ is fast, taking only $O(|x|)$ operations.

LZW Feature Space. The other branch of the Lempel-Ziv family is represented by the LZW algorithm, which is closely related to $LZ78$ and a host of variants. Unlike $LZ77$, which refers to strings in a sliding dictionary window, LZW builds and stores an explicit substring dictionary on the fly, selecting new substrings to add to the dictionary using a greedy selection heuristic. Output codes refer to substrings already in the dictionary; we will again assume fixed-length output codes of size c for simplicity, and assume that the dictionary can hold $O(2^c)$ entries. With the standard LZW substring selection heuristic, the maximum length of a substring in the LZW dictionary is also $O(2^c)$. The vector space $\aleph$ implied by LZW is thus actually larger than that of $LZ77$ (assuming reasonable parameter values for each), and has one dimension for each possible substring with maximum length $O(2^c)$, that is, $O(2^{(2^c)})$ dimensions. LZW implicitly maps x to $\vec{x} \in \aleph$ as follows; each $\vec{x}_i$ is initialized at zero, and is incremented by c when an output code corresponding to dimension i is produced. Despite the high dimensionality of $\aleph$, the mapping is completed in time $O(|x|)$, and at the end of the process, $C(x) = ||\vec{x}||_1$.

PPM Feature Space. Extremely effective lossless compression is achieved with arithmetic encoding under a model of prediction by partial matching (Witten et al., 1999b). Arithmetic coding is a compression technique that attempts to make $C(x)$ as close as possible to the ideal $-\log P(x)$ by allowing symbols within a message to be encoded with fractional quantities of bits. An order-n predictive model is generated by building statistics for symbol frequencies, based on a Markovian assumption that the probability distribution of symbols at a given point relies only

5. Readers who wish to review the details of particular compression algorithms are referred to texts by Sayood (2000), Witten et al. (1999b), or Hankerson et al. (2003).
6. For a complete review of vector norms, see the text by Trefethen and Bau (1997).

on the previous n symbols in the stream. At each step, PPM encodes a symbol s following the n-symbol context t, using $c \approx -\log P(s|t)$ bits. Because of arithmetic coding, c is positive, but is not necessarily a whole number. Note that the probability estimate $P(s|t)$ is the algorithm's estimate at that step, and these estimates change during compression as the algorithm adapts to the data. However, the details of the probability estimation scheme, which differ by implementation and algorithm variant, do not alter the implicit feature space.

The implicit PPM feature space $\aleph$ has one coordinate for each possible combination of symbol s and context t, which may be thought of together as a single string ts. Thus, an order-n PPM feature space has one dimension for each possible string of length $n+1$. PPM maps x to $\vec{x} \in \aleph$ by beginning with each $\vec{x}_i = 0$. For each new symbol-context pair ts encountered during compression, where s is encoded with c bits and ts corresponds to dimension i in $\aleph$, $\vec{x}_i := \vec{x}_i + c$. At the end of compression, then, $C(x) = ||\vec{x}||_1$.

String Concatenation and Vector Addition. We have just shown that each of these prototypical compression algorithms has an associated feature space, $\aleph$, that each compressor maps a string x into a vector $\vec{x} \in \aleph$, and that for each compressor, $C(x) = ||\vec{x}||_1$. This is the foundation of the analysis of compression-based similarity measures in vector space. But before we can take apart the similarity measures, themselves, we need to examine the effect of string concatenation.

We define concatenation as string addition: strings $x + y = xy$. Compressing xy maps each string into a vector and adds the vectors together. Thus, $C(xy) = ||\vec{x} + \vec{y}||_1$, which satisfies the triangle inequality requirement of vector norms: $||\vec{x}+\vec{y}|| \leq ||\vec{x}||+||\vec{y}||$, because $C(xy) \leq C(x)+C(y)$ (in the absence of pathological cases; see below.) Note that the salient quality of the compressors here is that they are *adaptive*. In the absence of adaptive compression, $C(x) + C(y) = C(xy)$ for all strings, and machine learning is impossible. With adaptive compression, $\max\{C(x), C(y)\} \leq C(xy) \leq C(x) + C(y)$, just as $\max\{||\vec{x}||_1, ||\vec{y}_1||\} \leq ||\vec{x} + \vec{y}||_1 \leq ||\vec{x}||_1 + ||\vec{y}||_1$ if all elements of $\vec{x}$ and $\vec{y}$ are non-negative. Adaptive compression of concatenated strings performs vector addition within the implicitly defined feature space.

Yet a few caveats are in order. First, the commutative property of addition is not strictly met with all compressors. Because of string alignment issues and other details such as model flushing,[7] many adaptive compression algorithms are not purely symmetric – that is, $C(xy)$ is not exactly equal to $C(yx)$. Second, in some cases, even the triangle inequality requirement may fail to hold. If the initial string x uses up the entire dictionary space in Lempel-Ziv methods, $C(xy) > C(x)+C(y)$. And under PPM, if x is very different from y, it is possible that $C(xy) > C(x) + C(y)$, depending on the nature of the adaptive modeling scheme. In this case, the presence of model flushing is a benefit, keeping $C(xy)$ close to $C(x) + C(y)$ when x and y are highly dissimilar.

3.2 Measures in Vector Space

Measuring distance between our implicit vectors would be simple given a subtraction operator defined for the implicit vectors formed by compression mapping. If subtraction were available, we could use a quantity like $||\vec{x} - \vec{y}||$ as a distance metric. However, the only available vector operators in this implicit space are addition and magnitude. It turns out that all four of the compression-based similarity measures address this issue in the same way, and use the quantity $||\vec{x}||_1+||\vec{y}||_1-||\vec{x}+\vec{y}||_1$ as a vector similarity measure. Simple transformations show that this term occurs in each compression-based measure. When x and y are very similar, the term approaches $\max\{||\vec{x}||_1, ||\vec{y}||_1\}$. When the two share little similarity, the term approaches 0. However, when left un-normalized, the term may show more absolute similarity between longer strings than shorter strings. To allow meaningful comparisons in similarity between strings of different lengths, the measures are normalized. As

7. Some compression algorithms perform *model flushing* when the current model does not provide sufficient compression rates. The model is flushed, or thrown away, and the algorithm begins adapting from a null assumption.

Table 1: Reducing similarity measures to canonical form.

$CosS$	$= 1 - \frac{C(x)+C(y)-C(xy)}{\sqrt{C(x)C(y)}} = 1 - \frac{\|\|\vec{x}\|\|_1+\|\|\vec{y}\|\|_1-\|\|\vec{x}+\vec{y}\|\|_1}{\sqrt{\|\|\vec{x}\|\|_1\|\|\vec{y}\|\|_1}}$
CLM	$= 1 - \frac{C(x)-C(x\|y)}{C(xy)} = 1 - \frac{C(x)+C(y)-C(xy)}{C(xy)} = 1 - \frac{\|\|\vec{x}\|\|_1+\|\|\vec{y}\|\|_1-\|\|\vec{x}+\vec{y}\|\|_1}{\|\|\vec{x}+\vec{y}\|\|_1}$
CDM	$= \frac{C(xy)}{C(x)+C(y)} = 1 - \left(1 - \frac{C(xy)}{C(x)+C(y)}\right) = 1 - \left(\frac{C(x)+C(y)}{C(x)+C(y)} - \frac{C(xy)}{C(x)+C(y)}\right)$ $= 1 - \frac{C(x)+C(y)-C(xy)}{C(x)+C(y))} = 1 - \frac{\|\|\vec{x}\|\|_1+\|\|\vec{y}\|\|_1-\|\|\vec{x}+\vec{y}\|\|_1}{\|\|\vec{x}\|\|_1+\|\|\vec{y}\|\|_1}$
NCD	$= \frac{C(xy)-\min\{C(x),C(y)\}}{\max\{C(x),C(y)\}} = 1 - \left(1 - \frac{C(xy)-\min\{C(x),C(y)\}}{\max\{C(x),C(y)\}}\right)$ $= 1 - \left(\frac{\max\{C(x),C(y)\}}{\max\{C(x),C(y)\}} - \frac{C(xy)-\min\{C(x),C(y)\}}{\max\{C(x),C(y)\}}\right)$ $= 1 - \frac{C(x)+C(y)-C(xy)}{\max\{C(x),C(y)\}} = 1 - \frac{\|\|\vec{x}\|\|_1+\|\|\vec{y}\|\|_1-\|\|\vec{x}+\vec{y}\|\|_1}{\max\{\|\|\vec{x}\|\|_1,\|\|\vec{y}\|\|_1\}}$

shown in Table 1, all of the measures can be reduced to a canonical form $1 - \frac{\|\|\vec{x}\|\|_1+\|\|\vec{y}\|\|_1-\|\|\vec{x}+\vec{y}\|\|_1}{f(\vec{x},\vec{y})}$, where $f(\vec{x},\vec{y})$ is a particular normalizing term. (Subtracting the normalized similarity term from 1 makes these measures dissimilarity measures.)

Thus, the only differences among the measures is in the choice of normalizing term: $CosS$ normalizes by the geometric mean of the two vector magnitudes, CDM by twice the arithmetic mean, NCD by the max of the two, CLM by the magnitude of the vector sum. Indeed, in the experimental section, we see that the four measures give strikingly similar results.

3.3 From Feature Vectors to Compression

We have shown that compression maps strings into feature vectors, and have looked briefly at the similarity measures applied in the feature spaces. Now we show that standard explicit feature vectors have strong links back to compression. These connections show the potential for improved explicit models based on insights from adaptive compression methods.

TF*IDF. One conventional approach to text representation is the TF*IDF vector model (Salton and Buckley, 1988), in which coordinates of the vector space correspond to individual words, and are given a score based on the term frequency times its inverse document frequency. The value of each $\vec{x}_i$ corresponding to a word w_i occurring n_i times in string x containing $|x|$ total words, and which occurs with probability $P(w_i)$ in some reference corpus, is given by $\vec{x}_i = n_i \log \frac{1}{P(w_i)}$. The connection to compression is clear: a word based compression algorithm, given a fixed probability distribution $P(w)$ for all possible words, will compress x to a length $C(x)$ such that:

$$\sum_i n_i \log \frac{1}{P(w_i)} = ||\vec{x}||_1 = C(x)$$

Thus, the only difference between a word-based compression method and TF*IDF is that the latter represents its feature space explicitly – which is an advantage for certain learning algorithms such as SVM or decision trees. Yet the insight begs the question: if compression algorithms are able to better model the data by *adapting* their estimation of the probability distribution $P(w)$ during compression, may TF*IDF achieve better results using an adaptive scoring method inspired by compression techniques? We plan to pursue this interesting line of inquiry in future work.

Binary Bag of Words. The *binary bag of words* method gives the elements of a word vector a binary $\{0, 1\}$ score indicating presence or non-presence of a word, but ignores the number of

repetitions. This may be viewed as equivalent to a form of *lossy compression* of a given text, in which all words are given equal probability, but the frequencies of words in the text are discarded in compression.

n-Grams and k-Mers. In the n-gram feature model, the vector space has one coordinate for each possible substring of n symbols, called an n-gram (or, alternately, a k-mer or p-spectrum) (Shawe-Taylor and Cristianini, 2004). The score for an element x_i of an n-gram vector representing string x is the count of n_i times that the (possibly overlapping) n-gram g_i appears in x. The link to compression is made with a uniform probability distribution $P(g)$:

$$\sum_i n_i \log \frac{1}{P(g_i)} = ||\vec{x}||_1 = C(x)$$

With compression techniques at hand, we can see the potential for an n-gram method with adaptive probability distributions inspired by the PPM algorithm as another area for future work.

4. An Empirical Test

As an initial confirmation of the tight connection between compression-based similarity methods and explicit feature vector models, we conducted a classification experiment on the Unix User Data Set archived at the UCI machine learning repository (Blake and Merz, 1998). We selected this publicly available data set, as it is free from some of the ambiguities and noise that occur in other benchmark data sets, such as 20-Newsgroups or Reuters-21578 (Khmelev and Teahan, 2003). This makes classification by the Nearest Neighbor technique a fair test, allowing direct comparison of similarity measures drawn from various data models. To the best of our knowledge, this is the first apples-to-apples test of compression-based measures against explicit feature vector models appearing in the literature.[8]

The UNIX user data set contains labeled transcripts of nine Unix system users, developed for testing intrusion detection systems (Lane and Brodley, 2003). We apply this data to a user classification problem: given a test string x of Unix commands and a training set of labeled user sessions, identify the user who generated x. We employed the Nearest Neighbor classification method, first using the four compression-based similarity measures NCD, CDM, CLM, and $CosS$ in combination with each of the compression algorithms discussed in Section 4. We then repeated the tests with three standard explicit feature vector models, the binary bag of words, TF*IDF vectors, and n-gram models, using the established cosine-vector similarity measure to compute similarity scores. Tests were run for 1000 randomly selected test sessions, with a minimum length of 10 Unix commands in the session string.[9] We report accuracy as the measure of success for each method, based on number of correct Nearest Neighbor classifications.

The results, detailed in Table 2, show that the compression-based methods compete with, and even exceed, the performance of common explicit vector space methods. It is interesting to note that the PPM methods out-perform the associated n-gram methods, despite the two sharing the same feature space. This suggests that the weights implicitly assigned to the coordinates in the vector space by PPM were indeed informative.

8. Frank et al. (2000) tested a compression-based discriminant against other learning methods such as SVM; thus, the learning method factored in results. Here the learning algorithm is kept constant across all models.
9. The split training and test data sets are publicly available at http://www.eecs.tufts.edu/~dsculley/unixSplits, posted with permission.

Table 2: **Unix User ID Results.** Accuracy is reported over 1000 trials.

Compressor	NCD	CosS	CDM	CLM
PPM 3	.801	.834	.836	.839
PPM 4	.808	.828	.830	.830
LZ77	.735	.710	.725	.720
LZW	.669	.691	.691	.714
Vector Model	Binary Bag	TF*IDF	4-gram	5-gram
	.838	.791	.777	.759

5. Discussion and Conclusions

Perhaps the most difficult problem in machine learning and data mining is choosing an appropriate representation for the data. At first blush, compression-based methods seem to side-step this problem, but more thorough examination shows that a choice of compression algorithm implies a specific, definable representation of the data within a feature space. This view removes compression-based similarity from idealized information space, and allows such measures to be justified on the same terms as explicit feature vector models such as TF*IDF and n-gram models.

Empirical results in a range of papers have shown that compression-based similarity can achieve success on many important problems, and we have shown here that they may be competitive with explicit feature models. However, there are a number of limitations on the use of standard compression algorithms for machine learning. First, the familiar counting argument from compression literature (Hankerson et al., 2003) coincides with the No Free Lunch Theorem (Wolpert and Macready, 1995). Simply put, there can be no algorithm that losslessly compresses all possible strings, just as there is no machine learning method that automatically learns from all data. The choice of compression algorithm implies a particular set of features, and these must align well with the chosen data. Second, computing similarity with off-the-shelf compression algorithms may require more computational overhead than using explicit feature vector methods. While both forms of similarity may be computed in time $O(|x| + |y|)$, in practice the constant for compression algorithms is much greater. Third, and perhaps most importantly as demonstrated by Frank et al. (2000), explicit feature models are more easily used by the full range of machine learning algorithms.

Yet while the lack of explicit features for machine learning algorithms with compression methods discouraged Frank et al. (2000) in this line of inquiry, we feel that there may well be productive future work combining the best elements of both compression and explicit feature models. It seems promising to start with the idea of adaptive term and n-gram weighting schemes inspired by adaptive compression. Another possibility is to store the substrings found by Lempel-Ziv schemes as explicit features. Ample opportunities for cross-fertilization between data compression and machine learning promise interesting, productive future work.

Acknowledgments

Our deep appreciation is given to Roni Khardon for his insightful questions, and to Serdar Cabuk for his careful reading and comments. We would also like to thank the UC Irvine Machine Learning Archive for the use of the UNIX User data set. Finally, thanks are given to Mark Nelson for providing source code for LZW and PPM, and to Marcus Greelnard for his $LZ77$ code.

References

D. Benedetto, E. Caglioti, and V. Loreto. Language trees and zipping. *Phys. Review Lett.*, 88(4), 2002.

C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

A. Bratko and B. Filipic. Spam filtering using compression models. Technical Report IJS-DP-9227, Department of Intelligent Systems, Jozef Stefan Institute, Ljubljana, Slovenia, 2005.

X. Chen, B. Francia, M. Li, B. Mckinnon, and A. Seker. Shared information and program plagiarism detection. *IEEE Trans. Information Theory*, 7:1545–1550, 2004.

X. Chen, S. Kwong, and M. Li. A compression algorithm for DNA sequences and its applications in genome comparison. In *Genome Informatics: Proceedings of the 10th Workshop on Genome Informatics*, pages 51–61, 1999.

R. Cilibrasi and P. Vitanyi. Clustering by compression. *IEEE Trans. Information Theory*, 51:4, 2005.

R. Cilibrasi, P. Vitanyi, and R. de Wolf. Algorithmic clustering of music based on string compression. *Computer Music Journal*, 28:49–67, 2004.

J. Cleary and I. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communcations*, 32(4):396–402, 1984.

E. Frank, C. Chui, and I. H. Witten. Text categorization using compression models. In *Proceedings of DCC-00, IEEE Data Compression Conference*, pages 200–209. IEEE Computer Society Press, Los Alamitos, US, 2000.

Marcus Greelnard. Basic Compression Library, 2004. URL `bcl.sourceforge.net`.

P. Grünwald. A tutorial introduction to the minimal description length principle. In P. Grünwald, I. J. Myung, and M. Pitt editors*, Advances in Minimum Description Length: Theory and Applications*, MIT Press, 2005.

J. Hagenauer, Z. Dawy, B. Goebel, P. Hanus, and J. Mueller. Genomic analysis using methods from information theory. *IEEE Information Theory Workshop (ITW)*, pages 55–59, October 2004.

D. Hankerson, G. A. Harris, and P. D. Johnson. *Introduction to Information Theory and Data Compression, 2nd ed.* Champan and Hall, 2003.

E. Keogh, S Lonardi, and C. Ratanamahatana. Toward parameter-free data mining. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 206–215, 2004.

D. V. Khmelev and W. J. Teahan. A repetition based measure for verification of text collections and for text categorization. In *SIGIR '03: Proceedings of the 26th Annual International ACM SIGIR Conference*, pages 104–110. ACM Press, 2003.

T. Lane and C. E. Brodley. An empirical study of two approaches to sequence learning for anomaly detection. *Machine Learning*, 51(1):73–107, 2003.

M. Li, J. H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17(2): 149–154, 2001.

M. Li, X. Chen, X. Li, B. Ma, and P. Vitanyi. The similarity metric. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 863–872, 2004.

M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and its Application. 2nd ed.* Springer, 1997.

M. Nelson. LZW data compression. *Dr. Dobb's Journal*, pages 29–36, October 1989.

M. Nelson. Arithmetic coding and statistical modeling. *Dr. Dobbs Journal*, pages 16–29, February 1991.

G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.

K. Sayood. *Introduction to Data Compression, 2nd ed.* Morgan Kaufmann, 2000.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.

T. Welch. A technique for high-performance data compression. *Computer*, pages 8–19, June 1984.

I. H. Witten, Z. Bray, M. Mahoui, and W. J. Teahan. Text mining: A new frontier for lossless compression. In *Data Compression Conference*, pages 198–207, 1999a.

I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images, 2nd ed.* Morgan Kaufmann, 1999b.

D. H. Wolpert and W.G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, NM, USA, 1995.

J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.