SANDIA REPORT
SAND2023-10771R
Printed October 2023

Sandia National Laboratories

Chaconne: A Statistical Approach to Nonlocal Compression for Supervised Learning, Semi-Supervised Learning, and Anomaly Detection

Alexander Foss, Kurtis Shuler, Christina Ting, Rich Field, Travis Bauer, Sihai Dave Zhao, Eduardo Cardenas-Torres

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

Sandia National Laboratories
U.S. DEPARTMENT OF ENERGY NNSA

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods

ABSTRACT

This project developed a novel statistical understanding of compression analytics (CA), which has challenged and clarified some core assumptions about CA, and enabled the development of novel techniques that address vital challenges of national security. Specifically, this project has yielded the development of novel capabilities including

1. Principled metrics for model selection in CA,
2. Techniques for deriving/applying optimal classification rules and decision theory to supervised CA, including how to properly handle class imbalance and differing costs of misclassification,
3. Two techniques for handling nonlocal information in CA,
4. A novel technique for unsupervised CA that is agnostic with regard to the underlying compression algorithm,
5. A framework for semisupervised CA when a small number of labels are known in an otherwise large unlabeled dataset.
6. The academic alliance component of this project has focused on the development of a novel exemplar-based Bayesian technique for estimating variable length Markov models (closely related to PPM [prediction by partial matching] compression techniques).

We have developed examples illustrating the application of our work to text, video, genetic sequences, and unstructured cybersecurity log files.

ACKNOWLEDGMENT

We would like to thank the following individuals for their help with this project. David Stracuzzi, John Feddema, and the rest of the CIS committee for the helpful feedback in the early stages of project ideation and throughout the life of the project; Mayuri Shakamuri, for help with obtaining cyber log-based data and identifying opportunities for Chaconne method application; Michael Bernard, for helpful discussions of relevant national security applications and customer meetings; Amelia Henriksen, for helpful work on streaming data analysis considerations; Jim Brandt, for helpful discussions of relevant HPC log-based data analysis.

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

CONTENTS

LIST OF FIGURES

NOMENCLATURE

CA Compression analytics; the use of file compression algorithms for machine learning (ML).

DFT Discrete Fourier Transform.

ML Machine learning.

NCD Normalized Compression Distance.

NID Normalized Information Distance.

PPM Prediction by partial matching, an adaptive statistical compression technique.

1. INTRODUCTION

Compression Analytics (CA) refers to the use of file compression algorithms for machine learning. One of the foundational papers in this field ([22]) establishes the use of file compression to construct a similarity/distance metric between bytestreams as a way to approximate the Kolmogorov complexity [23]. This allows any distance-based machine learning technique (nearest-neighbors techniques, hierarchical clustering [9], etc) to be used. The goals of this project are two-fold: (1) seek to explain and extend these and other results in the field of CA, specifically, by developing a statistical interpretation of CA, and (2) extending CA techniques to handle nonlocal relationships.

Section 2 of this report reconsiders the assumptions underpinning this construction in light of optimal classification theory [3], and offers a novel formulation of CA for provably optimal classification and the use of CA with decision theoretic techniques. Section 3 offers a novel statistical interpretation of the common compression algorithm Prediction by Partial Matching (PPM; [10]), allowing for uncertainty quantification of PPM predictions. Section 4 describes multiple approaches for extending CA techniques to handle scenarios where the probabilities underlying the structure of a categorical sequence change slowly over time. Section 5 extends the results of Section 2 and describes the derivation of general algorithms for unsupervised and semi-supervised cluster analysis using any arbitrary compression algorithms. Section 6 describes extensions to PPM that allow for modeling relationships between windows of tokens much larger than traditional PPM analysis allows. Section 7 describes the use of dimension reduction in conjunction with CA approaches for analysis of video. Section 8 describes the use of Bayesian context trees for modeling categorical sequences, and a novel model for flexibly estimating parameters governing tree behavior. Finally, Section 9 offers concluding remarks.

2. OPTIMAL CLASSIFICATION WITH CA

2.1. The Main Idea

Normalized Information Distance (NID) The foundational development of compression distance is based on Kolmogorov complexity (see [23]). For a binary string x, we define $K(x)$ as the length of the shortest binary program x^* to reconstruct x on a universal Turing machine. We similarly define the conditional Kolmogorov complexity of x given y as $K(x \mid y)$, which denotes the length of the shortest binary program to compute x if y is given [22].

The normalized information distance (NID) is defined by Li et al. [22] as

$$d(x,y) = \frac{max\{K(x \mid y^*), K(y \mid x^*)\}}{max\{K(x), K(y)\}}. \tag{1}$$

However, in that paper, various alternatives are described and rejected on arguments based on properties of the associated distance metrics. One alternative formulation given is

$$d(x,y) = \frac{K(x \mid y^*) + K(y \mid x^*)}{K(x,y)}. \tag{2}$$

At least five other alternatives are mentioned, based on taking Equations (1) and (2) and replacing the denominator with either $K(x,y)$, $max\{|x|,|y|\}$, or $|x|+|y|$. Ultimately, Equation (1) is adapted for compression analytics and reasonable justification is given, but ideally we would be able to select a distance metric based on statistical first principles (more on this upcoming).

Normalized Compression Distance (NCD) Equation (1) is adapted for real-world computations by using the compressed length of x, denoted $C(x)$, as an approximation for $K(x)$:

$$d(x,y) \approx NCD(x,y) = \frac{C(xy) - min\{C(x), C(y)\}}{max\{C(x), C(y)\}}, \tag{3}$$

where $C(xy)$ denotes the compressed length of the concatenation of x and y. Note that the form of NCD compared to Equation (1) is somewhat different due to the substitution of terms using well-known identities, with the aim of avoiding practical limitations of computing $C(x \mid y)$[1].

If we seek to classify a string x into either class A or B, we can build a classifier by comparing $d_A = NCD(x, s(A))$ and $d_B = NCD(x, s(B))$, where $s(A)$ denotes a representative string[2] from class A. The decision rule involves classifying x into class A if $d_A < d_B$. [3]

The NCD has proven to be a useful tool for compression analytics (CA), but, as we show below, an improved alternative formulation for NCD is available. For example, the NCD-based decision

[1] $C(x \mid y)$ is certainly computable in theory, but most implementations of compression algorithms do not offer convenient ways to compute these quantities.

[2] Depending on the data available and goals of the project, the representative string may be a collection of documents from class A, or a single document.

[3] Ties can be handled by allocating into class A with probability 0.5.

rule does not yield a probability of class membership, a method for handling prior probabilities of class membership, nor access to decision-theoretic tools. We first introduce some necessary concepts, and then offer this improved alternative decision rule based on established optimal classification theory [3].

Link Between Compression Distance and Probability We know from Shannon's Source Coding Theorem [28] that the expected length of an optimal code for a random variable X is contained in the interval

$$\left(\frac{H(X)}{\log_2(a)}, \frac{H(X)}{\log_2(a)} + 1\right)$$

where $H(X)$ denotes the entropy of X and a denotes the alphabet size used for encoding. This is based on the fact that $H(X) = E[I_X(X)]$, where $I_X(x) = -\log_2 P_X(x)$ denotes the information content in the observation $X = x$, that is, the number of bits needed to optimally encode x when the distribution of X is known. $P_X(x)$ denotes the probability distribution of the random variable X as a function of manifestation x.

Compression as an approximation to information Compression algorithms work because they are generally able to approximate the information content of input files. Following this fact, and in a similar spirit to ([22]), we can approximate $I_X(x)$ by considering the number of bytes required to compress x by a compression algorithm C that has been trained[4] on a sufficiently large sample from X (which we denote $\mathbb{X}$):

$$I_X(x) \approx C(x \mid \mathbb{X}), \tag{4}$$

where $C(x \mid \mathbb{X})$ denotes the compressed length of x given by compression algorithm C that has been trained on $\mathbb{X}$. The information content is a lower bound on the compressed length of x, so in fact $I_X(x) \leq C(x \mid \mathbb{X})$. Some of the subsequent derivations rely on the assumption that $C(x \mid \mathbb{X}) - I_X(x) \approx 0$. Poor choice of a compression algorithm C can violate this assumption. While tools for compression analytics often contain implementations allowing the calculation of $C(x \mid \mathbb{X})$, most standard implementations of compression algorithms do not contain an option to separate training and application of a compression algorithm. To accommodate these latter cases, we may further suppose that $C(x \mid \mathbb{X})$ can be approximated as the difference between the compressed length of $concat(\mathbb{X}, x)$ and $\mathbb{X}$:

$$I_X(x) \approx C(concat(\mathbb{X}, x)) - C(\mathbb{X}), \tag{5}$$

[4]Here we use the term "train" in the typical machine learning sense. However, this might not be immediately apparent when used in the context of CA. The training process in CA involves building a compression model from the training data as if we were compressing the training data set. The testing process involves using this compression model to compress the testing data without updating the internal state of the model. Since this latter step is not always feasible using off-the-shelf file compression tools, and approximations such as those in Eq. (5) are often used. Note that the information required from CA might not require the actual full file compression algorithm to be run; indeed, approaches have been developed (e.g., [4]) that eliminate some of these steps that are superfluous from a CA perspective.

where the simplified notation $C(z) := C(z \mid z)$ denotes a compression model both fit and applied to z. Note that these expressions are agnostic to the particular compression algorithm used. Selection of the appropriate compression model to use is analogous to the standard statistical problem of selecting the appropriate model for P_X.

2.2. Optimal Classification Rules with Compression Analytics

Following these relationships, we consider the optimal classification rule ([3]) for classifying an observation x into class k out of K possible classes. We allocate x to class k if

$$\pi_k P(x \mid \text{class}\, k) > \pi_j P(x \mid \text{class}\, j),\ \forall j \in \{1,2,...,K\},\ j \neq k.$$

where π_a denotes the prior probability of class a, and $P(x \mid \text{class}\, k)$ denotes the probability of x conditional on x being drawn from class k. More discussion on the definition and usage of prior probabilities is given in Sect. 2.4. Given the approximations above, and taking the log of both sides, we can rewrite this as

$$\log_2 \pi_k - C(x \mid \mathbb{X}_k) > \log_2 \pi_j - C(x \mid \mathbb{X}_j),\ \forall j \neq k, \tag{6}$$

or

$$\log_2 \pi_k - C(concat(\mathbb{X}_k, x)) + C(\mathbb{X}_k) > \log_2 \pi_j - C(concat(\mathbb{X}_j, x)) + C(\mathbb{X}_j),\ \forall j \neq k, \tag{7}$$

where $\mathbb{X}_k$ denotes a sufficiently large sample of data from class k.

We can further generalize this to accommodate decision rules that minimize the expected cost of misclassification (adapted from Theorem 6.7.1 in [3]). In this theorem, we partition our sample space into K exhaustive and mutually exclusive regions $R_1, R_2, ..., R_K$, and allocate an observation x into class k if $x \in R_k$. These regions are constructed as

$$R_k = \left\{ x : \sum_{i=1}^{K} \pi_i P(x \mid \text{class}\, i) cost(k \mid i) < \sum_{\ell=1}^{K} \pi_\ell P(x \mid \text{class}\, \ell) cost(j \mid \ell) \right\},\ \forall j = 1,2,...,K,\ j \neq k,$$

where $cost(k \mid i)$ denotes the cost incurred by classifying an observation into class k if it was truly drawn from class i. This accommodates scenarios where $cost(k \mid k) \neq 0$.

Using the approximation given in Eq. (4), this decision rule can be approximated as

$$R_k \approx \left\{ x : \sum_{i=1}^{K} \pi_i 2^{-C(x|\mathbb{X}_i)} cost(k \mid i) < \sum_{\ell=1}^{K} \pi_\ell 2^{-C(x|\mathbb{X}_\ell)} cost(j \mid \ell) \right\},\ \forall j = 1,2,...,K,\ j \neq k, \tag{8}$$

which follows because $P(x \mid \text{class}\, i)$ can be approximated by $2^{-C(x|\mathbb{X}_i)}$. This is no longer a strict equality due to the approximation error in using compression length as an estimate for information; as mentioned above, a poor choice of compression algorithm will result in poor approximations to the optimal classification rules. Note that due to the summations, we can no longer take the log transformation of both sides as in Eq. (6), and must rely on exponentiating the approximate information as given by Eq. (5).

Similarly, using the approximation given in Eq. (5), this decision rule can be approximated as

$$R_k \approx \left\{ x : \sum_{i=1}^{K} \pi_i 2^{C(\mathbb{X}_i)-C(concat(\mathbb{X}_i,x))} cost(k \mid i) < \sum_{\ell=1}^{K} \pi_\ell 2^{C(\mathbb{X}_\ell)-C(concat(\mathbb{X}_\ell,x))} cost(j \mid \ell) \right\}, \ \forall j = 1,2,...,K, \ j \neq k, \quad (9)$$

which follows because $C(x \mid \text{class}\, i)$ can be approximated by $C(concat(\mathbb{X}_i,x)) - C(\mathbb{X}_i)$.

2.3. Probabilities of Class Membership

In addition to deriving optimal classification rules, we can use the relationship in Eq. (4) to derive posterior probabilities of class membership. We begin with the probability of an observation x drawn from random variable X conditional on (given) membership in class i,

$$P(X = x \mid x \in \text{class}\, i) := P(X = x \mid C = i).$$

We seek the probability of class membership conditional on our observation, $P(C = i \mid X = x)$.[5] We proceed with a generic application of Bayes' rule to obtain what we want ($P(C \mid X)$) from what we have in hand ($P(X \mid C)$):

$$\begin{aligned} P(C = i \mid X = x) &= \frac{P(X = x \mid C = i)P(C = i)}{\sum_j P(X = x \mid C = j)P(C = j)} \\ &\approx \frac{2^{-C(x|\mathbb{X}_i)}\pi_i}{\sum_j 2^{-C(x|\mathbb{X}_j)}\pi_j} \end{aligned} \quad (10)$$

where the summation is over all classes in the training set.

2.4. A Note on Prior Probabilities and Class Imbalance

The prior probabilities of class membership, π_i, denote the probability of observing a member of class i *before* observing your data. For example, if we are classifying (diagnosing) individuals based on the presence or absence of a pathogen, the priors would reflect the overall prevalence of the disease. Priors play an important role in deriving probabilities of class membership by enforcing the need for stronger evidence for identification of rare categories. In terms of Eq. (10), this means we require a higher $P(X = x \mid C = i)$ for classifying into class i if $P(C = i) := \pi_i$ is low. In cases where the true prior probabilities are highly imbalanced, it is critical to use methods that take them into account for valid classification probabilities.

In practice, priors may be obtained from subject matter expertise, estimated from a random sample from the population, or simply set to be equal in the absence of any relevant knowledge. Note that the priors in Eqs. (8) and (10) cancel out if they are all equal.

[5] The distinction between $P(X = x \mid C = i)$ and $P(C = i \mid X = x)$ is a critical one; the latter is the quantity we must obtain to calculate "class membership probabilities."

2.5. Comparison with Prior Work

A commonly used (but flawed) classification rule (e.g., Sect. 6 of [5]) is to allocate an observation x to class k such that

$$\begin{aligned} k &= \arg\max_i P(x \mid \text{class } i) \\ &\approx \arg\min_i C(x \mid \mathbb{X}_i). \end{aligned}$$

In contrast to Eq. (6), this classification rule implicitly assumes that all classes are equal, i.e. $\pi_1 = \cdots \pi_K$. This assumption can be correct in some circumstances, but in many others (particularly when the classes are imbalanced) it will lead to sub-optimal classification decisions. Additionally, this decision rule cannot be used to minimize cost of misclassification as described above.

Equation (5) is similar in spirit to expressions used in [22], particularly the compression distance

$$C(concat(x,y)) - \min\{C(x), C(y)\}.$$

In fact, these expressions are identical if we set $y = \mathbb{X}$. The primary difference is that Eq. (5) assumes that $\mathbb{X}$ is a large representative sample from a random variable X, whereas [22] assumes x and y are two comparable samples. The reference [22] goes on to normalize this difference by $\max\{C(x), C(y)\}$ in order to construct a distance metric that can be used to quantify the similarity between x and y. Our Eq. (5) is instead adapted for classification.

3. A STATISTICAL INTERPRETATION OF PPM

3.1. A Statistical Interpretation of PPM Probabilities

In this document, we show that PPM probabilities can be properly understood as Bayesian estimates of multinomial parameters with specific Dirichlet priors. We further show that the probabilities in the original PPM paper are only recoverable with mutually exclusive choices of prior parameters, suggesting a minor inconsistency in the original formulation. Finally, we discuss the implications of this finding, in particular, that it offers a statistically defensible way to incorporate UQ and informative prior information into PPM calculations.

3.1.1. The Core Derivation

In the 1984 paper [10] by Cleary and Witten describing PPM, they consider estimating the probability of a character that has never been seen. Specifically, in the second-to-last equation of their paper, they describe "the estimated escape probability of a novel character occurring relative to order m" as

$$e_m = \frac{1}{1+C_m},$$

where e_m gives the escape probability of interest, and C_m is the total count of observed characters whose highest context is of length m. For example, if for the context ab ($m = 2$) we have observed $\{abc, abd, abe\}$, then $C_m = 3$, we would escape if we attempt to predict on abf, and $e_m = 0.25$.

We can reformulate this in terms of a statistical Markov model. Consider a sequence of characters $X_{i-m}, X_{i-m+1}, ..., X_i$, in which X_i denotes a categorical random variable taking on a finite set of k_m possible values A_m. For simplicity, consider $A_m = \{1, 2, ..., k_m\}$. Consider the problem of estimating the distribution of X_i given a fixed history

$$H = \{X_{i-m} = x_{i-m}, X_{i-m+1} = x_{i-m+1}, ..., X_{i-1} = x_{i-1}\},$$

that is, we would like to estimate

$$Pr(X_i = \phi \,|\, H),$$

where we stay (somewhat) consistent with [10] by using ϕ to represent the next character in a sequence.

A standard approach is to model X_i using the multinomial distribution with $n = 1$ (AKA the categorical distribution) and probability vector $\vec{p}$:

$$X_i \,|\, H \sim Multin(n = 1, \vec{p}),$$

which leads to the MLE (maximum likelihood estimator) for

$$\hat{p}_j = \frac{c_m(\phi = j \,|\, H)}{C_m(\cdot \,|\, H)},$$

where we extend the notation (c_m and C_m from [10]) to allow for additional precision of notation as follows. The quantity $c_m(\phi = j \,|\, H)$ denotes the count of the number of occurrences j whose highest context length m consists of state H, and

$$C_m(\cdot \,|\, H) = \sum_{j=1}^{k_m} c_m(\phi = j \,|\, H).$$

We now move to a Bayesian inferential scheme for $\vec{p}$ using the (conjugate) Dirichlet prior $Dir(\alpha_1, \alpha_2, ..., \alpha_k)$, where α_j denotes pseudocounts for category j. The posterior distribution on $\vec{p}$ also follows the Dirichlet distribution with parameters updated as

$$\alpha_j + c_m(\phi = j \,|\, H), \quad \forall\, j = 1, 2, \ldots, k,$$

that is, the posterior distribution is

$$\vec{p} \,|\, \vec{\alpha}, X_i, H \sim Dir(c_m(\phi = 1 \,|\, H) + \alpha_1, c_m(\phi = 2 \,|\, H) + \alpha_2, \ldots, c_m(\phi = k \,|\, H) + \alpha_k). \tag{11}$$

Using basic properties of the Dirichlet distribution, this leads to conditional expectations

$$\begin{aligned} E[p_j \,|\, \vec{\alpha}, X_i, H] &= \frac{c_m(\phi = j \,|\, H) + \alpha_j}{\sum_{\ell=1}^{k} c_m(\phi = \ell \,|\, H) + \alpha_\ell} \\ &= \frac{c_m(\phi = j \,|\, H) + \alpha_j}{C_m(\cdot \,|\, H) + \sum_{\ell=1}^{k} \alpha_\ell} \end{aligned}$$

For particular values of the $\vec{\alpha}$ prior parameters, we can recover the original expressions in [10] as expectations of the Dirichlet distribution. The probability of a particular character $\phi = j$ (notated as $p_m(\phi)$ in the appendix of [10]) is recovered if we set $\alpha_j = 0$ and $\sum_{\ell=1}^{k} \alpha_\ell = 1$:

$$E[p_j \,|\, \vec{\alpha}, X_i, H] = \frac{c_m(\phi = j \,|\, H) + 0}{C_m(\cdot \,|\, H) + 1},$$

and the escape probability (e_m in the original) is recovered for a $\phi = j$ which has never occurred for history H (that is, $c_m(\phi = j \,|\, H) = 0$) if we set $\alpha_j = 1$ and all other α's to zero:

$$e_m = E[p_j \,|\, \vec{\alpha}, X_i, H] = \frac{1}{C_m(\cdot \,|\, H) + 1}.$$

Thus, we see that while we can recover the exact probability estimators presented in [10], they require incompatible prior specifications. These incompatible settings result in an algorithm that uses different levels of prior uncertainty for identical data depending upon whether an unrelated character has been observed. It would be more consistent to set all alphas equal to 1 or $1/k$, for example. While these observations are interesting from a mathematical perspective, the nature of the potential corrections is not expected to have a large effect on PPM probabilities, with the possible exception of cases where inferences must be drawn from a very small training dataset.

Informative priors: More impactful would be to use this Bayesian framework to incorporate informative prior information into a PPM analysis. For example, the $\vec{\alpha}$ parameters could be derived based on counts from a relevant corpus, resulting in a compressor that is highly tuned to a particular domain (e.g., an algorithm that is already tuned to handle English language journal articles, Swedish blog posts, C++ source code, etc.). This setup could be used to implement a transfer learning framework, in which information from a large semi-relevant corpus is built into a PPM compressor which is then further trained on a smaller dataset of high interest.

3.2. Uncertainty Quantification: Credible Intervals on PPM Probabilities

Here we consider using the posterior Dirichlet distributions to derive credible intervals (CI) on the individual probability estimates, and consider extending this to products of the probability estimates. Here we may search for equal-tailed CI, or HDR (highest density region), which will be different since the Dirichlet distribution can be asymmetric.

Let us initially consider calculating the covariance of $\vec{p}$ from equation (11):

$$Cov[p_i, p_j \,|\, \vec{\alpha}, \vec{X}, H] = \frac{I\{i=j\}E[p_i] - E[p_i]E[p_j]}{1+\sum_\ell^k \alpha_\ell}.$$

Using established techniques, CI or HDR can be derived. Exploring trade-offs of differing approaches here is beyond the scope of the current project.

3.2.1. Delta Method for Products of Binomial Probabilities

Individual probabilities are rarely of interest in PPM; rather, we must aggregate probability estimates across a document (where document refers to some aggregated sequence of tokens). Here we seek a method for deriving confidence intervals for a product of the k binomial probabilities $\theta = \prod_i^k p_i$ that emerge from this setting. Here, $\hat{p}_i = X_i/n_i$, where $X_i \sim Binom(n_i, p_i)$, with $X_i \perp X_j$ for $i \neq j$[6].

Interestingly, the product of Bernoulli random variables is itself Bernoulli. However, binomial random variables are more complicated. An approach for $k = 2$ was described in [6].

We first appeal to the multivariate delta method, which states that

$$\sqrt{n}(h(\hat{\vec{\beta}}) - h(\vec{\beta})) \xrightarrow{D} MVN\left(\vec{0}, \nabla h(\vec{\beta})^\top \Sigma \nabla h(\vec{\beta})\right),$$

for an arbitrary function $h(\cdot)$, where MVN denotes the multivariate normal distribution, $\xrightarrow{D}$ denotes convergence in distribution, and $\Sigma = Cov(\hat{\vec{\beta}})$. An approach based on using the delta method to derive the asymptotic distribution of the sum of log probabilities (e.g., [1]) results in the $\alpha \times 100\%$ confidence interval

$$\hat{\theta}\exp(\pm z_{1-(\alpha/2)}\hat{\sigma}_{ln(\hat{\theta})}),$$

[6]Here we use $\perp$ to denote statistical independence.

where $\theta = \sum_{i=1}^{k} \log(p_i)$ and

$$\begin{aligned}\hat{\sigma}^2_{ln(\hat{\theta})} &= \sum_{i=1}^{k} (1-\hat{p}_i)/(\hat{p}_i n_i) \\ &= \vec{1}_k^{\top} (I_k - P) N^{-1} P^{-1} \vec{1}_k. \end{aligned} \tag{12}$$

where $P = diag(p_i)$, $N = diag(n_i)$, I_k denotes the k-dimensional identity matrix, and $\vec{1}_k$ denotes the one vector of length k. Note the similarity between this and confidence intervals for relative risk in standard use.

A similar approach using the multivariate delta method on the product of probabilities directly yields

$$\begin{aligned}\nabla h(\vec{p})^{\top} \Sigma \nabla h(\vec{p}) &= |P|^2 \vec{1}_k^{\top} (I_k - P) N^{-1} P^{-1} \vec{1}_k, \\ &= |P|^2 \hat{\sigma}^2_{ln(\hat{\theta})} \end{aligned} \tag{13}$$

4. MODELS FOR SLOW TIME-VARYING TRENDS

4.1. General Problem

The problem we investigate here is modeling Markov-style transition probabilities that vary gradually over time. For a time-ordered sequence of tokens $X_t \in \{1, 2, \ldots, \Omega\}$, we seek to model

$$Pr(X_t = i \mid X_{t-1}, X_{t-2}, \ldots X_{t-c}) = p_{it}$$

for some context length c, where p_{it} is the probability of observing token i given the history at time point t.

4.2. DFT of Categorical Sequential Data

Fourier analysis is one approach to measure linear dependence in sequential data. It is most commonly applied to real-valued time series, but there has been some research in the bioinformatics community and elsewhere to develop methods for Fourier analysis of categorical data [30, 29]. In this section, we apply Fourier methods to categorical data in an attempt to discover any non-local dependence.

The basic idea is to take a sequence of tokens, encode each token as a numerical value, then take the discrete Fourier transform (DFT) of the real-valued sequence. Peaks in the DFT can indicate dependencies in the original sequence; the frequency axis in this case represents cycles per token. The sequence must be weakly stationary in order for the DFT results to have meaning. Extensions to non-stationary sequences using the short-time DFT are possible, which are very helpful when dependencies can change over the length of the sequence.

The choice of encoding method is fundamental to this idea, and the results of any Fourier analysis will depend on the encoding technique. A unique, optimal encoding scheme can be obtained for some applications, but optimal encoding methods are not available in general; this is an area of open research. We provide a few examples below to illustrate the concepts of Fourier analysis of sequences of tokens.

Example 1 A single strand of DNA consists of many linked, smaller components called nucleotides. Each nucleotide is one of four possible types designated by the letters A (adenine), T (thymine), C (cytosine), and G (guanine), so that any fragment of DNA can be represented by a string of these 4 letters.

Single DNA strands tend to form double helices with other single DNA strands. The pair of strands are complementary to each other because each nucleotide of one strand is linked to a nucleotide of the other strand by a chemical bond: A is linked to T and vice versa, and C is linked to G and vice versa. An encoding scheme that takes advantage of these relationships is [2]

$$\begin{aligned} \mathrm{A} &\mapsto 1 + i & \mathrm{C} &\mapsto -1 - i \\ \mathrm{T} &\mapsto 1 - i & \mathrm{G} &\mapsto -1 + i \end{aligned} \tag{14}$$

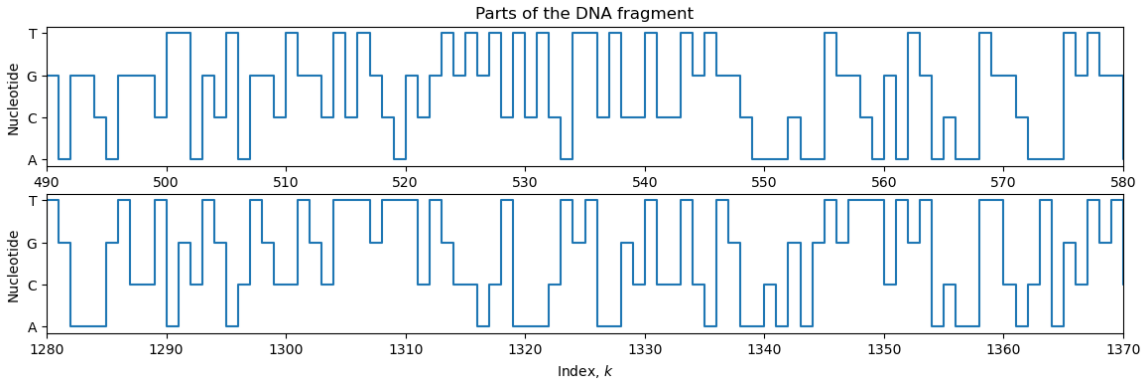


Figure 4-1 Parts of the DNA fragment Yh1F5 taken from [31].

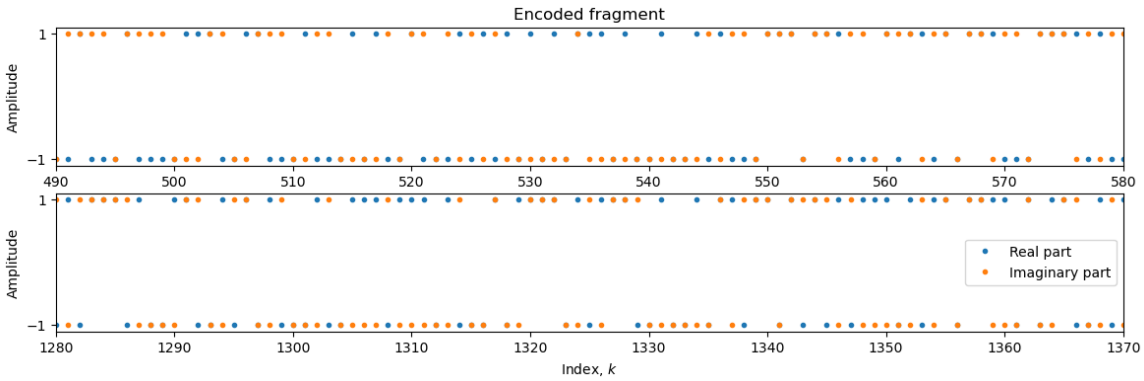


Figure 4-2 DNA fragment illustrated by Fig. 4-1 encoded according to Eq. (14).

where $i = \sqrt{-1}$ is the imaginary unit. By Eq.(14), (A, T) and (C, G) are encoded as complex conjugate pairs.

To illustrate, we consider the Y-chromosonal DNA fragment Yh1F5 [31], consisting of 4,156 nucleotides. Parts of this fragment are illustrated by Fig. 4-1; the corresponding encoded version according to Eq. (14) is illustrated by Fig. 4-2. The short-time DFT of the encoded DNA fragment is illustrated by Fig. 4-3. There is a clear peak in the amplitude of the short-time DFT near positions k such that $3{,}100 < k < 3{,}400$; this peak indicates some structure in this region of the DNA fragment.

Example 2 Next, let a_i be a token, and let $\mathscr{A} = \{a_1, \ldots, a_n\}$ be a set of distinct tokens, i.e., an alphabet. One approach to encode the tokens in $\mathscr{A}$ is simple integer encoding, i.e., $a_1 \mapsto 0, a_2 \mapsto 2, \ldots, a_n \mapsto n-1$. Now define

$$x_k = \left\lfloor \frac{n-1}{2} \left(1 + \cos\left(2\pi \frac{kp}{m} \right) \right) \right\rceil, \; k = 0, \ldots, m-1, \tag{15}$$

to be a sequence of length m, where $\lfloor z \rceil$ denotes the nearest integer to real-valued number z, and $p > 0$ is a parameter. By construction, $\{x_k\}$ is an integer-valued cosine wave that takes values in $\{0, \ldots, n-1\}$ with period p.

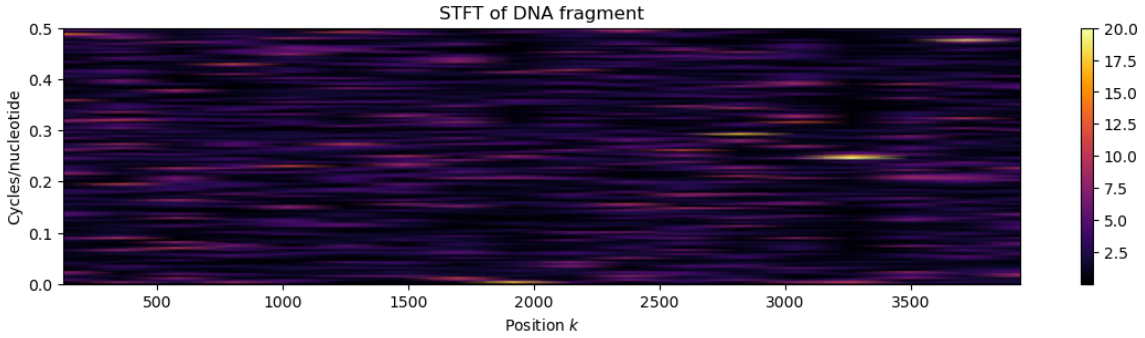


Figure 4-3 Short-time DFT of the Y-chromosonal DNA fragment Yh1F5.

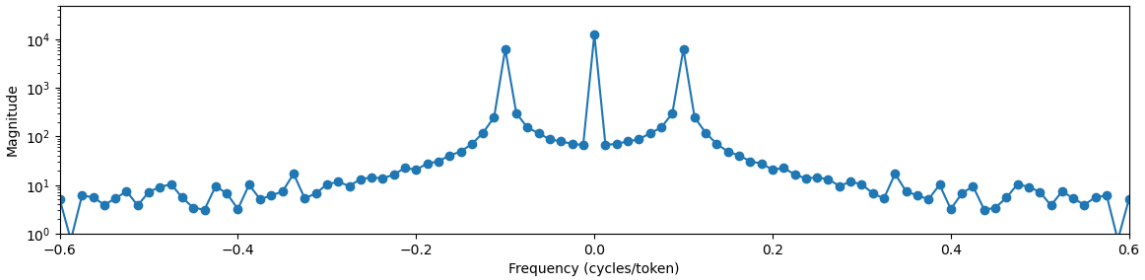


Figure 4-4 DFT of the sequence of tokens defined by Eq. (17).

The DFT of $\{x_k\}$ is given by

$$X_k = \frac{m(n-1)}{4}\left(2\delta(k) + \delta\left(k - \frac{mp}{2\pi}\right) + \delta\left(k + \frac{mp}{2\pi}\right)\right),\ k = -m/2, \ldots, m/2-1, \qquad (16)$$

where $\delta(z)$ is the Dirac delta function, equal to one if $z = 0$ and equal to zero otherwise. If we interpret $\{x_k\}$ defined by Eq. (15) to be the encoded version of an underlying sequence of tokens from $\mathscr{A}$, then Eq. (16) defines the Fourier amplitudes of this sequence of tokens.

$$\mathscr{A} = \{\mathrm{A}, \ldots, \mathrm{Z}, \mathrm{a}, \ldots, \mathrm{z}\}$$

be an alphabet with $n = 52$ tokens. The sequence

$$\mathrm{zzyyxwvtrpnljgebZWURPNKJHFECBBAAAABCDEFHJLNPSUXZce}\ldots \qquad (17)$$

can be encoded by Eq. (15) with $p = 1/10$. The DFT of this sequence is illustrated by Fig. 4-4; there are peaks at frequencies 0 and $\pm p$ according to Eq. (16).

Despite the open questions regarding how best to conduct the integer-coding step described above, we think this approach could be a promising technique for identifying periodicity in categorical data. We prioritized most development work on other thrusts during this project, but future work should continue to assess this technique for handling categorical sequences with slowly evolving dependence structures.

4.3. Multinomial Regression with Sinusoidal Basis Functions

Let X_t denote a random variable describing token observed at time t, with $X_t \in \{1, 2, \ldots, \Omega\}$. This idea involves modeling the transition probabilities with a collection of multinomial regression models with a fixed set of sinusoidal basis functions as covariates, that is,

$$Pr(X_t = i \mid X_{t-1}, X_{t-2}, \ldots X_{t-c}) = p_{it} = softmax(\vec{w}_t^{\top} \beta_i),$$

where β_i is a vector of unknown feature weights corresponding to outcome i, and $\vec{w}_t$ is a vector of sinusoidal basis functions

$$w_{jt} = \sin\left(\frac{\pi j(t - min)}{max - min}\right),$$

where $j = \{1, 2, 3, \ldots\}$, *min* is the earliest time point in the dataset, and *max* is the latest time point. The above equations show the 0th order model. The full form of this model would need a separate β weight vector for each combination of items in the history:

$$Pr(X_t = i \mid X_{t-1} - j, X_{t-2} - k, \ldots X_{t-c} - z) = p_{it} = softmax(\vec{w}_t^{\top} \beta_{ijk\cdots z}),$$

which could potentially be quite messy to implement.

Example We implemented the 0th order model in `R`, with a simple usage example shown here. We constructed a string of 1000 characters drawn uniformly at random from the lowercase English alphabet. We then replaced a random 20% of the letters in the middle 50% of the string with the letter "b" (see Figure 4-5). Using a small set of sinusoidal basis functions (see Figure 4-6), we modeled the presence of each of the 26 letters using the 0th order model as described above. As is typical in standard multinomial regression, a baseline category must be specified – the character "a" in this case. Results are shown in Figures 4-7 and 4-8. Inspecting the fitted model parameters, we see that the model correctly predicts a higher likelihood of the character "b" at the center of the string (Figures 4-7), and gives reasonable-looking estimates for other characters (Figure 4-8).

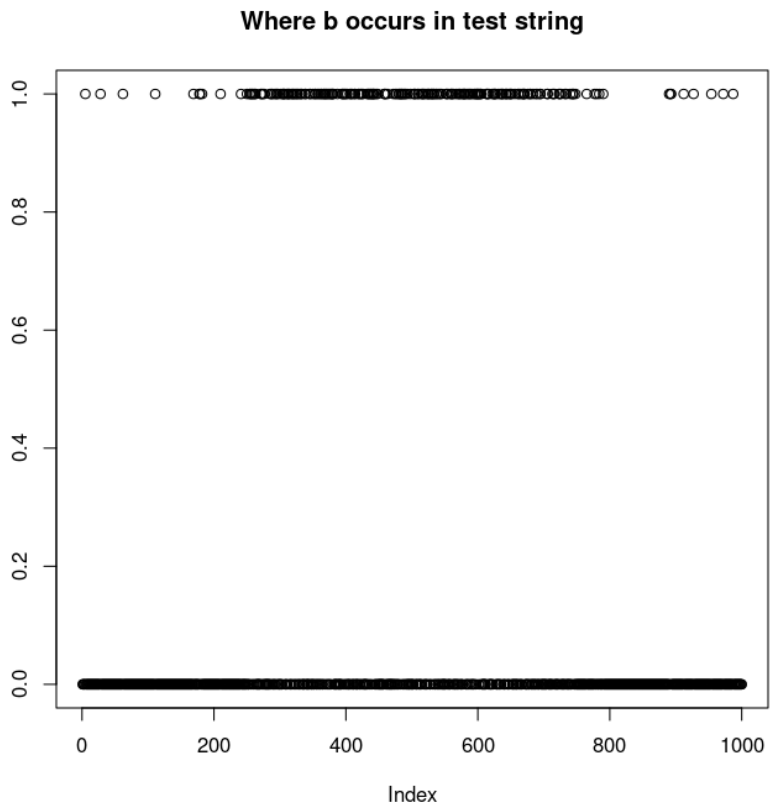


Figure 4-5 Structure of the test string. The index of the string is shown along the horizontal axis, and the vertical axis shows the presence (1.0) or absence (0.0) of the character “b”.

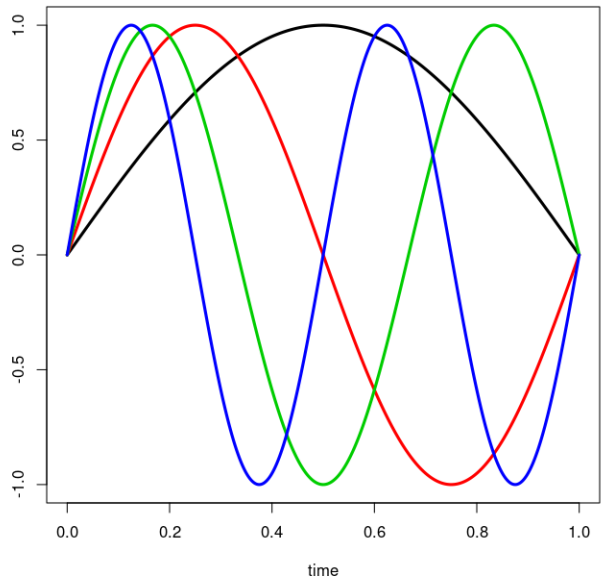


Figure 4-6 Sinusoidal basis for the multinomial regression design matrix.

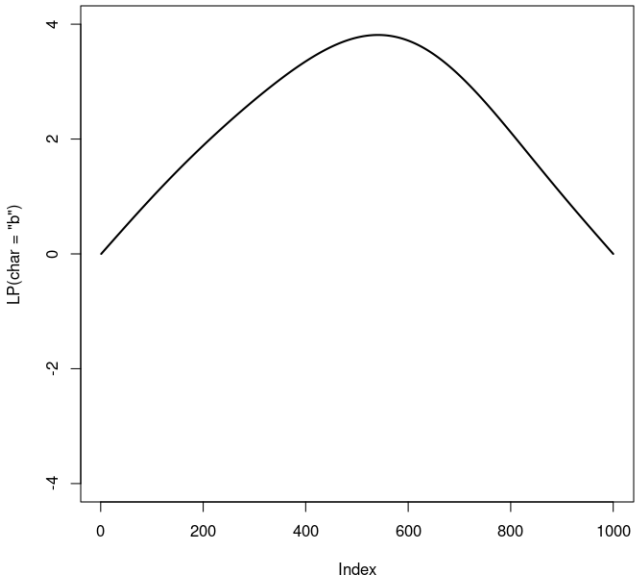
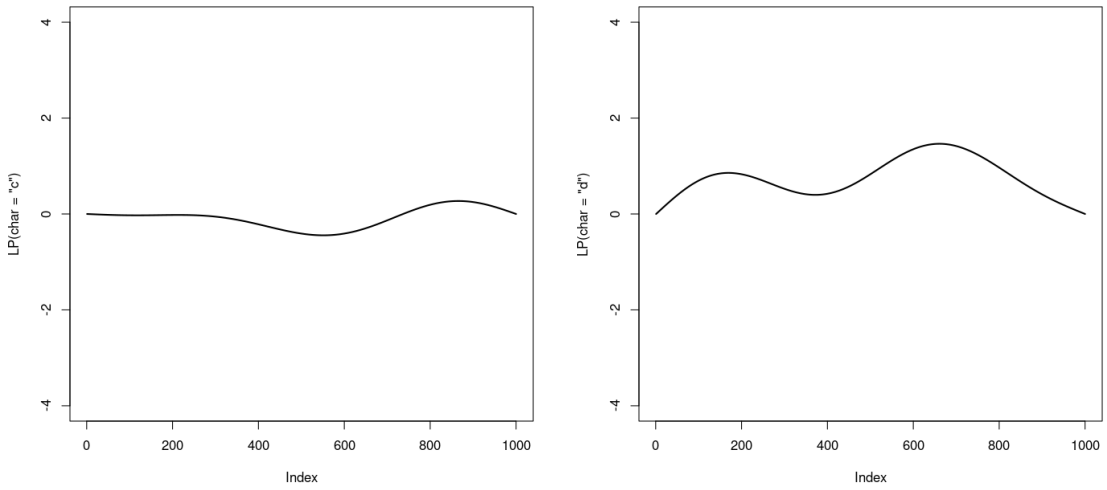


Figure 4-7 Linear predictor from fitted sinusoidal multinomial model. This plot shows the linear predictor estimated for the likelihood of observing the character "b" at any point along the sequence. This quantity is used by the softmax function to calculate probabilities of observing a character. Higher values in the middle 50% of the data correctly reflect the underlying data structure as described in the text body. Values close to zero reflect probabilities close to the baseline ("a") in this case.



(a) Linear predictor for character “c”. **(b) Linear predictor for character “d”.**

Figure 4-8 Linear predictor from fitted sinusoidal multinomial model. This plot shows the linear predictor estimated for the likelihood of observing the characters “c” (left) and “d” (right) at any point along the sequence. This quantity is used by the softmax function to calculate probabilities of observing a character. Values close to zero reflect probabilities close to the baseline (“a”) in this case.

4.4. Markov Transition Regression with Periodic GP Latent Factors

Here, we assume that

$$Pr(X_t = i \mid X_{t-1} = j) = C_{ijt},$$

with

$$(X_t \mid X_{t-1} = j) \sim Multinomial(n = 1, \theta = C_{:jt}),$$

where C_t is a time-varying $\Omega \times \Omega$ matrix of transition probabilities C_{ijt}, and each column of C_t is given by

$$C_{:jt} = SoftMax(\vec{Z}_{jt}),$$

where for some kernel $k(|t-t'|)$,

$$Z_{ijt} \sim GP(\vec{0}, k)$$
$$Cov(Z_{ijt}, Z_{i'j't}) = 0 \quad \forall i \neq i', \forall j \neq j'$$
$$Cov(Z_{ijt}, Z_{ijt'}) = k(|t-t'|)$$
$$k(|t-t'|) = \sigma^2 \exp\left\{-\frac{2\sin^2(\pi|t-t'|/p)}{\ell^2}\right\} \tag{18}$$

where $p > 0$ is the period, and ℓ is the "lengthscale" governing how fast the GP can fluctuate (independent of periodicity).

4.5. Markov Transition Regression with GP-Varying Transition Matrices

Let X_t denote a random variable describing token observed at time t, with $X_t \in \{1, 2, \ldots, \Omega\}$ and $t = 1, \ldots, n$. Here, we assume that

$$Pr(X_t = i \mid X_{t-1} = j) = C_{ij}^{w_t} D_{ij}^{1-w_t}$$

$$W_t = logistic(Z_t)$$
$$Z_t \sim GP(\vec{0}, k(|t-t'|))$$

where matrices $C = \{C_{ij}\}$ and $D = \{D_{ij}\}$ are not time-varying, and where the kernel k is chosen to be the periodic kernel defined in Eq. (18). This model has complete data log-likelihood

$$\ell(\vec{X}, \vec{Z}; C, D) = \log \prod_{t=1}^{n} Pr(X_t = i \mid X_{t-1} = j)$$
$$= \sum_{t=1}^{n} (W_t \log C_{ij} + (1 - W_t) \log D_{ij})$$

Since the likelihood includes random variables W_t, we derive an expectation-maximization (EM; [12]) algorithm for maximizing the likelihood. The E-step is

$$\underset{Z|\Theta=\Theta^{(t)},X}{E}[\ell(\vec{X}, \vec{Z}; C, D)] = \sum_{t=1}^{n} \left(\log(C_{ij}) E[W_t] + \log(D_{ij})(1 - E[W_t])\right)$$

which can be written as two inner products. The expectation is:

$$\underset{Z|\Theta=\Theta^{(t)},X}{E}[logistic(\vec{Z})] = \int\int_{\vec{Z}}\cdots\int logistic(\vec{Z})\frac{f(X|\vec{Z},\hat{\Theta}^{(t)})f(\vec{Z})}{\int\int_{\vec{y}}\cdots\int f(X\mid\vec{y},\hat{\Theta}^{(t)})f(\vec{y})\mathrm{d}\vec{y}}\mathrm{d}\vec{Z}$$

where we abuse notation slightly and use f to denote the marginal and conditional density functions of the variables therein.

While the above derivation was one nonlocal model we considered investigating, we ended up focusing our efforts on other models due to potential concerns about convergence issues in handling these GP-based models. It is possible that this approach would be viable, however. We would also be interested in covariance functions that lead to low-rank covariance matrices (and ideally low-rank inverse covariance matrices), but did not spend time on this project investigating that. Also considered for the latent variable were spline basis functions, as well as dynamic linear models with multinomial observation variables (while some R packages supported GLM DLMs, none we tested appeared to adequately support multinomial observation variables[7]).

[7]We investigated the R packages `dlmodeler`, `KFAS`, `FKF`, and `sspir`.

5. GENERAL EM-BASED CLUSTERING

5.1. Introduction

While there are many methods for cluster analysis on continuous numeric features, there are fewer techniques for clustering mixed-type data (e.g., combinations of continuous, ordinal, categorical, etc.) [16]. Clustering raw sequence data (e.g., collections of text, DNA sequences, raw binary blobs, etc.) has received less attention still (e.g., [9]). In this section, we seek to derive and validate a compression-analytics based clustering algorithm that is specialized for these latter data structures.

We aim to derive a technique based on the statistical interpretation of compression analytics described in Section 2.1. Our technique is completely agnostic with regard to the actual compression algorithm that is used – in fact, any reasonable compression algorithm can be used interchangeably in our implementation. As described below, the main idea is to derive a complete-data log-likelihood for a clustered dataset, and then use the Expectation-Maximization algorithm (EM; [12]) to predict the unobserved cluster labels.

Our method is different from the work of [9], which uses normalized compression distance (NCD) to build an $n \times n$ matrix of distances for n input items, followed by a hierarchical clustering step. Computation of the distance matrix requires computing $\binom{n}{2} = n(n-1)/2$ distances, and thus requires quadratic time and memory. Our technique, like k-means and Gaussian mixture models, avoids this by computing distances/similarities between n items and k clusters (with $k < n$), resulting in an algorithm that has complexity proportional to $i \cdot n \cdot k$, where i is the number of iterations required by the EM algorithm.

5.2. Methods

5.2.1. Model Definition

Define the following quantities:

- X_{ij}, an observed random variable denoting token i from document j, with t_j tokens in document j;
- Z_j, an unobserved RV denoting class of document j, with d documents in total, and with $Z_j \in \{1, 2, ..., K\}$; and
- $\Pr(Z_j = k) = \pi_k$, the prior probability that a randomly selected document is drawn from class k.

We assume that

$$X_{ij} \mid Z_j = k, x_{i-1,j}, x_{i-2,j}, ... \sim C(\theta_k),$$

where C is some sequence model with parameters θ_k, e.g., a compression model or Markov model, with PMF

$$p(x_{ij} \mid Z_j = k, x_{i-1,j}, x_{i-2,j}, \ldots).$$

This can be simplified to

$$p(x_{ij} \mid z_j, h_{ij}) = q(ijk),$$

for notational convenience, where h_{ij} is the relevant history for x_{ij} and $q(ijk)$ is the probability for x_{ij} given z_j and h_{ij}.

The complete data log-likelihood can be written as

$$\begin{aligned}\ell(\Theta; \mathbb{X}, \mathbb{Z}) &= \log\left(\prod_{j=1}^{d}\prod_{i=1}^{t_j} p(x_{ij} \mid z_j, h_{ij})\right) \\ &= \log\left(\prod_{j=1}^{d}\prod_{i=1}^{t_j} q(ijk)\right)\end{aligned}$$

where Θ denotes the collection of parameters for the sequence models, and $\mathbb{X}$ and $\mathbb{Z}$ denote the collections of all observations of X_{ij} and Z_j, respectively.

5.2.2. Model Fitting with the EM Algorithm

Define the collection of all unknown parameters $\Theta = \{\theta_1, \theta_2, \ldots, \theta_K, \pi_1, \pi_2, \ldots, \pi_K\}$. Then, the E-step is defined as

$$\begin{aligned}Q(\Theta \mid \hat{\Theta}^{(t)}) &= \underset{\mathbb{Z}|\Theta=\hat{\Theta}^{(t)},\mathbb{X}}{E}[\ell(\Theta; \mathbb{X}, \mathbb{Z})] \\ &= \sum_{j=1}^{d}\sum_{i=1}^{t_j} \underset{\mathbb{Z}|\Theta=\hat{\Theta}^{(t)},\mathbb{X}}{E}\left[\log p(x_{ij} \mid Z_j, h_{ij})\right] \\ &= \sum_{j=1}^{d}\sum_{i=1}^{t_j}\sum_{k=1}^{K} \log(q(ijk)) \Pr(Z_j = k \mid \mathbb{X}, \Theta = \hat{\Theta}^{(t)}) \\ &= \sum_{j=1}^{d}\sum_{i=1}^{t_j}\sum_{k=1}^{K} \log(q(ijk)) \frac{\Pr(Z_j = k, X_{1j}, X_{2j}, \ldots, X_{t_j j} \mid \Theta = \hat{\Theta}^{(t)})}{\Pr(X_{1j}, X_{2j}, \ldots, X_{t_j j} \mid \Theta = \hat{\Theta}^{(t)})} \\ &= \sum_{j=1}^{d}\sum_{i=1}^{t_j}\sum_{k=1}^{K} \log(q(ijk)) \frac{\Pr(Z_j = k) \Pr(X_{1j}, X_{2j}, \ldots, X_{t_j j} \mid Z_j = k, \Theta = \hat{\Theta}^{(t)})}{\sum_{\ell=1}^{K} \Pr(Z_j = \ell) \Pr(X_{1j}, X_{2j}, \ldots, X_{t_j j} \mid Z_j = \ell, \Theta = \hat{\Theta}^{(t)})} \\ &= \sum_{j=1}^{d}\sum_{i=1}^{t_j}\sum_{k=1}^{K} \log(q(ijk)) \frac{\pi_k \prod_{i=1}^{t_j} \hat{q}^{(t)}(ijk)}{\sum_{\ell=1}^{K} \pi_\ell \prod_{i=1}^{t_j} \hat{q}^{(t)}(ij\ell)} \\ &= \sum_{j=1}^{d}\sum_{i=1}^{t_j}\sum_{k=1}^{K} \log(q(ijk)) w_{jk}\end{aligned}$$

where $\hat{\Theta}^{(t)}$ denotes the estimate of Θ at iteration t, and $\hat{q}^{(t)}$ denotes the estimate of q at iteration t. Note that this is a weighted log-likelihood expression with weights w_{jk}.

The M-step is defined as

$$\hat{\theta}^{(t+1)} = \arg\max_{\Theta} Q(\Theta \mid \hat{\Theta}^{(t)}).$$

This step would amount to a weighted maximum likelihood problem, except that most compression algorithms don't have a natural way to handle weights on input sequences. Some options for proceeding with the M-step are

1. Use standard statistical tools with maximum likelihood estimation for $\hat{q}$.
2. Approximate the effect of weights by upsampling documents for the training set in proportion to w_{jk}.
3. Use a hard EM algorithm (similar to k-means) by using modified weights w^*_{jk} as defined below. Then, the M-step simply amounts to classifying every item based on w^* and performing standard optimization within these estimated classes. It's a bit of a leap to equate the MLE with whatever default optimization is used by the chosen compression technique, but the consequences of this should be studied carefully. If we use log base 2, then the Q function involves a weighted number of encoded bits for the full dataset, which suggests a direct linkage to standard compression goals.

Hard weights can be defined as

$$w^*_{jk} = \begin{cases} 1 & \text{if} \quad w_{jk} > w_{jk'} \; \forall \, k \neq k' \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

Note that the $\{\pi_k\}$ can be estimated by simply summing up the "number" of documents in that class by taking $\hat{pi}_k = \sum_{j=1}^{d} w_{jk}$.

5.2.3. Methods for Avoiding Within-Cluster Overfitting

We have found in experimentation that the framework as defined above tends to overfit within each cluster, often in a fairly detrimental way. This leads to each cluster getting "stuck" in local optima that are substantially worse than the global optimum.

This problem is especially acute when using the hard EM algorithm, which effectively assigns every document to a cluster based on w^*. Using the hard EM algorithm, a document with a rare sequence will be assigned to a single cluster k. The model representing this cluster will "learn" the rare sequence much better than the other $K-1$ models. That is, θ_k will be updated in the M-step in such a way that the likelihood of the rare sequence will be much higher for the model that has seen that document before. The effect of the rare sequence on the document's likelihood can be so significant that it will be highest in class k due to the rare sequence, even if overall the document has many more commonalities with documents in one of the other $K-1$ classes. As a result, documents with rare sequences can get "stuck" within whatever cluster the document was initially assigned when using the hard EM algorithm, and never get re-assigned to a new cluster.

For similar reasons, the EM algorithm with hard weights can sometimes form larger clusters than are desirable, leaving some of the clusters empty or with very few documents. In this case, the large cluster "sees" many documents in the E-step (because many documents have already been assigned to that cluster), which leads to good estimates for θ_k which well-capture common

sequences. These common sequences may not be useful for distinguishing which documents belong to which cluster, but nonetheless contribute to the documents' overall likelihood. Thus, all else being equal, models (clusters) which have "seen" many documents will tend to have a higher likelihood for a new, unseen document than models which have seen fewer documents. The result is a self-reinforcing cycle where a large cluster k assigned many documents in the M-step will have estimates for θ_k which better capture commons sequence, which leads to more documents being assigned to that cluster in the M-step, and so on.

To combat these effects, we employ three strategies: 1. A K'-fold[8] like approach to fitting in the M-step, 2. A "cluster reset" step which allows small/empty clusters to learn from documents even if they are not assigned to that cluster, and 3. Restricting learning from the largest clusters by only allowing them to learn from some subset of the documents assigned to them.

The K'-fold strategy involves fitting each cluster in a cross-validation like fashion. Under this procedure, the corpus is partitioned into K' groups. Only $K' - 1$ of these groups are used to fit each model, with the held out group remaining "unseen." Cluster assignment for the held out group is performed using the models fit on the documents from the $K' - 1$ other groups. The hold-out procedure ensures that rare sequences will not be overly influential during cluster assignment, which alleviates the problem of documents containing rare sequences not changing clusters.

When using the hard EM algorithm, the cluster reset strategy allows a model to learn from documents not assigned to the model's corresponding cluster. Once the total number of documents assigned to a particular cluster falls below some threshold, the model is allowed to update θ_k using documents not assigned to it. Consider a cluster that has no documents assigned to it. Such a cluster has no information to use to update θ_k, leading to poor model fit and low likelihood values for new documents, regardless of the documents' content. As a result, in subsequent iterations of the EM algorithm the likelihood for *every* document in the corpus will be small for this model, which again leads to no documents being assigned to that cluster, and so on. The cluster reset strategy allows this model to use some documents not assigned to it to update θ_k. In the E-step, for example, this cluster might treat Z_j as if every document in the corpus were assigned to it. This procedure can "jump start" the small/empty cluster, letting it improve its estimates of θ_k and get some documents assigned to it. After the cluster has been reset and the number of documents assigned to the previously small/empty cluster has surpassed the cluster reset threshold, the EM algorithm continues to proceed iteratively in the usual fashion.

The final strategy for avoiding within-cluster overfitting, restricted learning, restricts models (clusters) so that they may only "learn" from some maximum number of documents M_d. When using the hard EM algorithm, this corresponds to only using M_d documents to estimate θ_k, even if the number of documents assigned to that cluster is greater than M_d. The restricted learning procedure helps prevent the effect where large clusters get even larger because they have seen more documents and, accordingly, have better estimates for θ_k for many sequences.

Particularly when used in conjunction, these procedures can substantially reduce within-cluster overfitting. The K'-fold strategy helps mitigate documents with rare sequences not getting

[8] K' refers to the number of folds in the common cross-validation strategy named 'K-folds,' not the K clusters. We use K' instead of K to avoid ambiguity.

Cluster	Text
1	Sandia Nation lndia Naboratoryraxyies is a multional L
1	Sandia multimississ multimiss al Laboratorinaboratorie
1	Sand laboratoryaboratoriesLaltorieLorie myoratorya mto
1	...
2	Hurrard Cantd elburda in Fiona Canane Fiona barrels to
2	Hurricanada in arrels tonarre Fiona barrenther Canada
2	Hurricane Fiona ranada barrene weanare war her hiCowar
2	...

Table 5-1 Synthetic text generated from two clusters

re-assigned to a more appropriate cluster, and the cluster reset and restricted learning strategies encourage better cluster separation. We have seen the benefits of these strategies in preventing within-cluster overfitting both in simulated datasets where the cluster assignment is known, and in real world datasets measuring clustering performance using the adjusted Rand Index (ARI; [19]).

As an example, we applied cluster reset and restricted learning to a set of synthetic data generated using a modified probability model similar to the probability model specific by PPM. Markov models were built up using seed sentences, and then random sequences of text were generated by drawing from these Markov models. In this simulated dataset, two seed sentences were used, and 25 random sequences were generated from each seed sentence. Examples of the generated sequences are shown in Table 5-1. Because the sequences were generated from known probability models, each sequence can be associated with a known cluster label. In some cases applying the EM clustering algorithm without overfitting adjustments can result in the sequences being inseparable, with all 50 sequences falling into a single cluster, as shown in the confusion matrix in Figure 5-2. Applying either cluster reset using a threshold of zero documents, or restricting learning to 5 documents results in perfect separability, with the algorithm able to identify the true cluster labels. We used these strategies using two different compression algorithms, NgramPPM [4] and zlib [17], and found improvements in the performance of both.

Other strategies we tried to reduce within-cluster overfitting were less successful, including sampling Z_j using weights derived from the model likelihoods, and alternative initialization schemes.

In our exploration of the EM-based clustering algorithm's performance and characteristics we considered two initialization schemes: Random partition, and a method inspired by the Forgy initialization algorithm for k-means [15]. Of the two, random partition is the simpler method to initialize the cluster labels. Using random partition initialization each document is randomly assigned a cluster label by sampling Z_j uniformly from $\{1,2,\ldots,K\}$ before iterating between the E-step and M-step as usual.

k-means, like our EM clustering algorithm, may use random partition to randomly assign cluster labels before iteratively fitting the model. The k-means algorithm performs "hard clustering," where each datapoint is assigned to a cluster with probability 1, as opposed to "soft clustering"

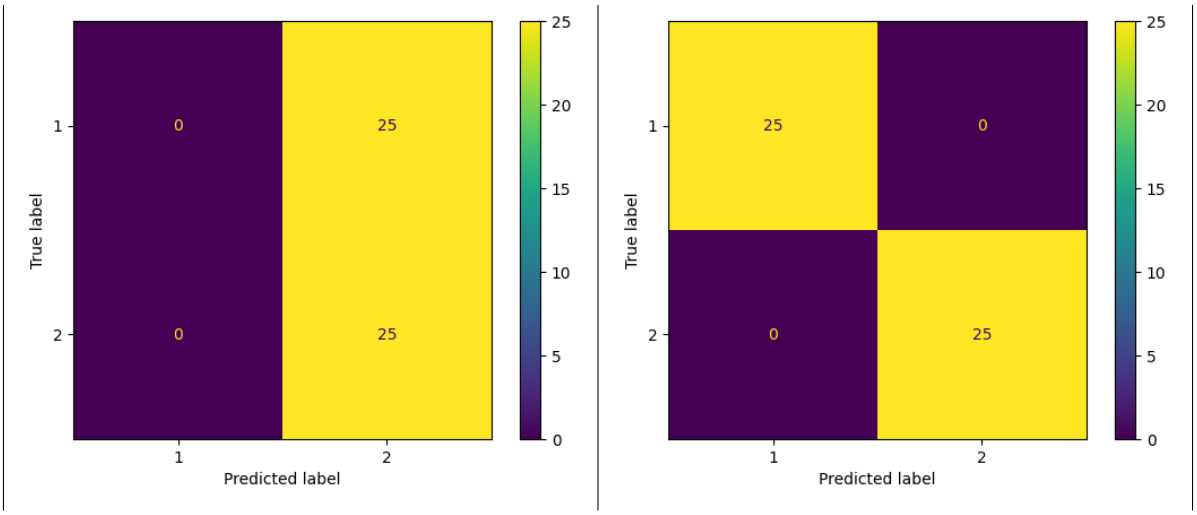


Table 5-2 Confusion matrices of predicted/true cluster labels with (right) and without (left) applying strategies for mitigating within-cluster overfitting

(like a Gaussian mixture model) which assigns a probability or score to each datapoint for belonging to each of the K clusters. Because it uses hard clustering a commonly observed phenomenon in k-means is empty clusters containing no data points, depending on the initial centers assigned to each cluster. To combat the empty cluster problem, [15] developed an alternative cluster initialization algorithm which assigns the initial cluster centers to locations chosen by randomly sampling K datapoints from the dataset. The cluster centers are chosen to be the same as the locations of those K randomly selected datapoints. After initialization the k-means algorithm then proceeds iteratively as usual, alternatively assigning cluster labels and moving the cluster centroids. Although the Forgy algorithm does not guarantee global convergence, it does help eliminate the problem of empty clusters by reducing within-cluster heterogeneity at the algorithm's initialization.

We developed an alternative initialization scheme inspired by the Forgy algorithm in an effort to eliminate empty clusters when using the hard EM algorithm. Under this initialization scheme, K documents from the corpus were selected at random, and each of the K compression models was fit to one of these documents. In this fashion, each compression model was allowed to "learn" from a single document selected at random from the corpus. The initial cluster labels were then generated by applying the K models, each of which only had learned from a single document in the corpus, and selecting Z_j for each document in the usual way, by assigning the cluster label according to which cluster had the highest likelihood/best compression ratio (assuming uniform prior cluster membership probabilities, i.e $\Pr(Z_j = k)$ the same for all k). By using this initialization method we hoped to reduce within-cluster heterogeneity and, like the Forgy initialization for k-means, eliminate or reduce the number of empty/small clusters observed after applying the EM-based clustering algorithm. Unfortunately we did not observe substantial improvement in reducing the number of small/empty clusters using this alternative initialization strategy over the simpler and faster random partition method.

Another method we tried to reduce within-cluster overfitting was sampling Z_j using weights derived from the model likelihoods rather than selecting Z_j according to Eq. (19). When the prior

cluster probabilities $\Pr(Z_j = k)$ are the same for all K classes the hard weight assignment rule is equivalent to choosing cluster membership based on which model has the highest log likelihood for the document under consideration. An alternative scheme is to choose Z_j randomly using weights $w_{jk}/\sum_{k'} w_{jk'}$. Using this procedure a document is most likely to be assigned to the class for which it has the highest likelihood, but there is non-zero probability that it may also may be assigned to one of the other $K-1$ classes. The motivation for changing the algorithm in this way when using the hard EM algorithm was that it allows some probability for a document to "escape" and be re-assigned to a more appropriate cluster. In particular, this change targeted the problem where documents with rare sequences could get stuck inside whatever cluster they were initially assigned. In our testing on simulated datasets however, we did not see significant improvement indicating documents with rare sequences would successfully change labels and join their "true" clusters.

5.3. Extension to Semisupervised learning

One way to extend the framework from the previous section to accommodate the semi-supervised paradigm is to assume a partially-labeled dataset. Here we describe what that model might look like.

5.3.1. Model Definition

In this case, we use the model as defined in Sect. 5.2.1, except that the random variables Z_j are partially observed. Specifically, we assume that Z_j is observed for all $j \in J_\ell$, $J_\ell \subset \{1,2,...,d\}$. Let J_u denote the unlabeled documents, with $J_\ell \cup J_u = \{1,2,...,d\}$.

5.3.2. Semisupervised Model Fitting with the EM Algorithm

The complete data log-likelihood is the same as above. We update the EM algorithm above in light of partially observed labels:

$$\begin{aligned} Q(\Theta \mid \hat{\Theta}^{(t)}) &= \underset{\mathbb{Z}|\Theta=\hat{\Theta}^{(t)},\mathbb{X}}{E}[\ell(\Theta;\mathbb{X},\mathbb{Z})] \\ &= \sum_{j\in J_\ell}\sum_{i=1}^{t_j}\log(q(ijZ_j)) + \sum_{j\in J_u}\sum_{i=1}^{t_j}\sum_{k=1}^{K}\log(q(ijk))w_{jk} \end{aligned}$$

5.4. Results

We tested our algorithm on simulated data, and a cybersecurity logfile dataset [33].

The logfile dataset involved samples of Linux, Mac, and Windows host-based logs, and the goal was to successfully cluster the logs into these three groups based on raw log content alone. We ran experiments in which the semisupervised algorithm described above was used with two distinct underlying clustering algorithms, NgramPPM [4] and the Python zlib library, based on gzip, a Lempel-Ziv-based compressor [17]. Our initial pilot study involved clustering a small sample of either 30 or 60 unlabeled items, 0 or 10 labeled items, a varying number of initializations and iterations, and 10 replicate (duplicate) analyses per factorial combination of conditions. We quantified the performance of the algorithm by taking the Adjusted Rand Index (ARI) between the true class labels and the predicted cluster labels [19].

As shown in Figure 5-1, including labeled items improved the results, and the clustering technique based on NgramPPM outperformed the zlib-based method. Increasing the number of iterations had the predictable effect of improving the solution, while increasing the number of initializations had a small effect on performance.

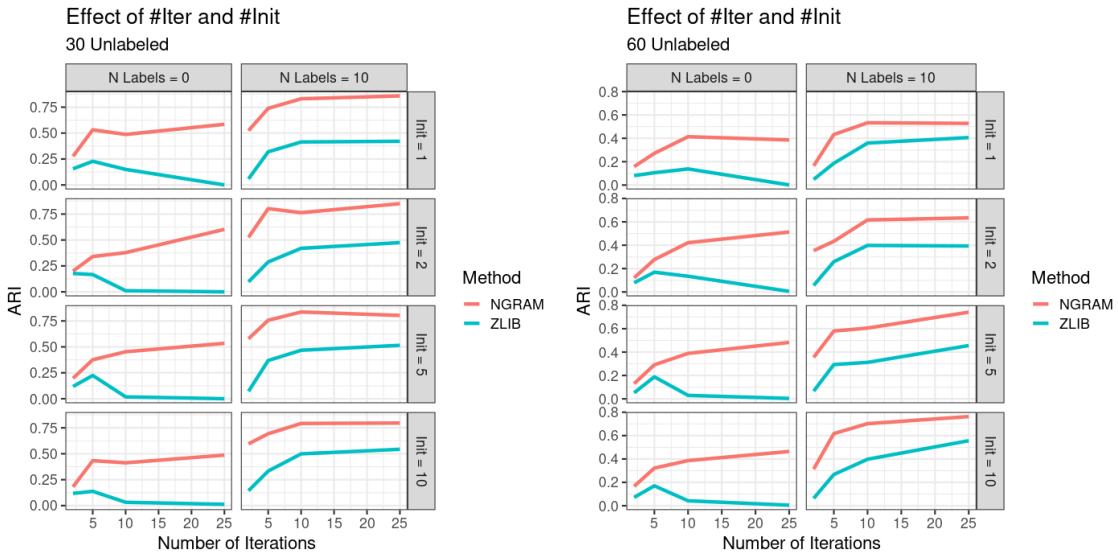


Figure 5-1 Results of semisupervised clustering, initial simulation. These plots show performance (ARI) of the semisupervised technique for two different dataset sizes (30 unlabeled logs per class, left; 60 unlabeled per class, right), different numbers of labeled samples (0 versus 10), different numbers of initializations per run (panel rows), and different numbers of iterations of the EM algorithm (horizontal axis).

Figure 5-2 shows the effect of increasing the number of labeled samples for two different dataset sizes. Although the performance of the zlib algorithm is particularly poor without any labeled samples (ARI close to zero), adding one labeled sample greatly improves its performance. In these conditions, PPM appears to outperform zlib in most conditions.

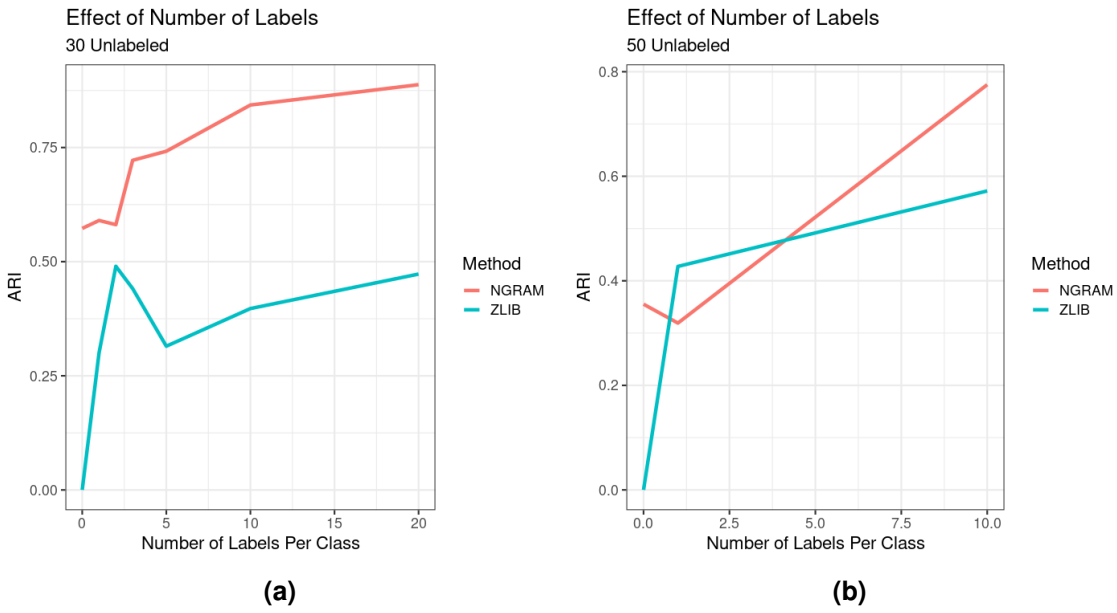


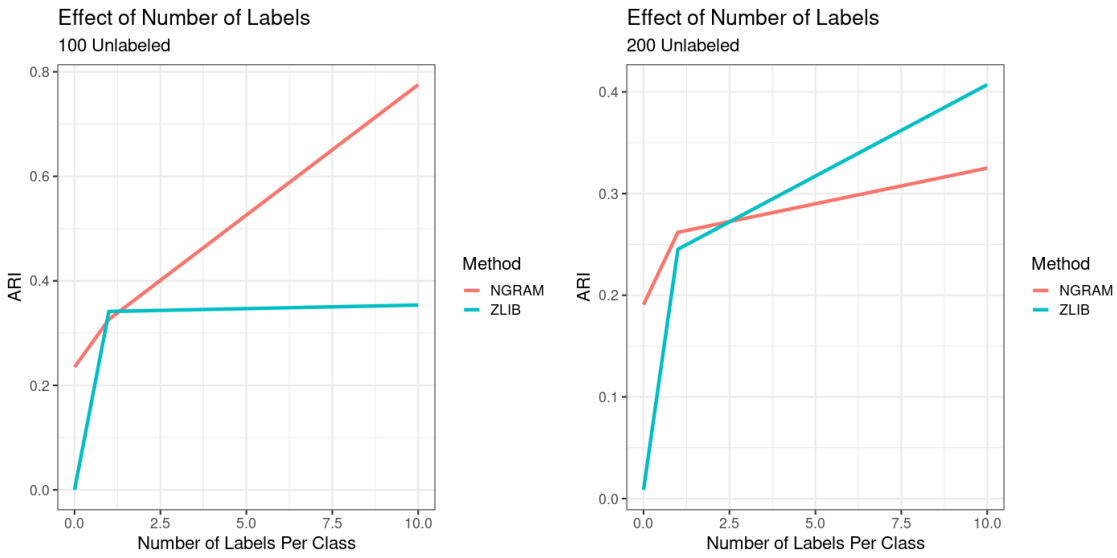
Figure 5-2 Effect of number of labeled samples on performance of the semisupervised algorithm. This plot shows the effect of increasing the number of labeled samples on algorithm performance (ARI).

Figure 5-3 shows results on larger dataset sizes (100 and 200 logs per class). Increasing the number of labels improves performance of both compression techniques in these conditions. Interestingly, for datasets with 200 logs per class, zlib outperforms PPM.

Due to the fact that the zlib algorithm was about 20 times faster than the PPM algorithm, we were able to run on larger dataset sizes for the zlib method only. These results are shown in Fig. 5-4; 30 replicates were taken per experimental condition. For smaller dataset sizes (less than about 1000 records per class), more labeled samples predictably improved performance. However, puzzlingly, including more labeled samples (10 or 50 per class) showed worse performance than one labeled sample per class for some larger dataset sizes. We suspect this may be related to unstable algorithm performance (ARI fluctuated between favorable values above 0.5 and below 0.1), and 30 replicates was not sufficient to stabilize the average performance across the simulations. The unstable algorithm performance appears to be related to an issue we have observed in numerous scenarios, in which one large cluster may come to capture a large majority of the dataset. Figure 5-5 illustrates this phenomenon. In this plot, we see that clustering performance tends to be poorer for increasing size of the largest cluster. This result appears to hold regardless of the number of labeled samples in the input data.

5.4.1. Results on Simulated Data

We also tested the semi-supervised algorithm on a set of simulated text sequences where the true probability distributions and cluster labels for each generated sequence was known. These simulated sequences were generated in the same manner as the data simulation process described in Section 5.2.3, where sequences were created using Markov model probabilities built up from



(a) **(b)**

Figure 5-3 Performance of the semisupervised algorithm on moderately sized datasets.

Cluster	Text
1	Sandia National Laboratories is a multimission laboratory
2	Los Alamos Laboratories is a single mission laboratory
3	The quick brown fox jumps over the lazy dog

Table 5-3 Seed sentences used to generate synthetic sequences

seed sentences. Here, we simulated data from three clusters, where two of the clusters deliberately had similar probability distributions to make disambiguating them more challenging. The three seed sentences used are shown in Table 5-3. Because of the similarities in the seed sentences, the generated sequences in clusters 1 and 2 tend to be similar. 25 synthetic sequences were generated from each of the three seed sentences; some examples of the sequences generated are shown in Table 5-4.

Figure 5-5 shows the results of applying the EM clustering algorithm using the zlib library without any labeled sequences. Without any labels, the clustering algorithm incorrectly groups the sequences from clusters 1 and 2 into a single cluster. By providing some labeled data, however, it the EM algorithm is able to distinguish these two clusters. Using the semi-supervised approach we provided labels for 20% of the sequences, chosen at random. After providing these labels the algorithm is able to correctly identify the true clusters membership of each of the 75 sequences, as shown in the confusion matrix in the right panel of Figure 5-5.

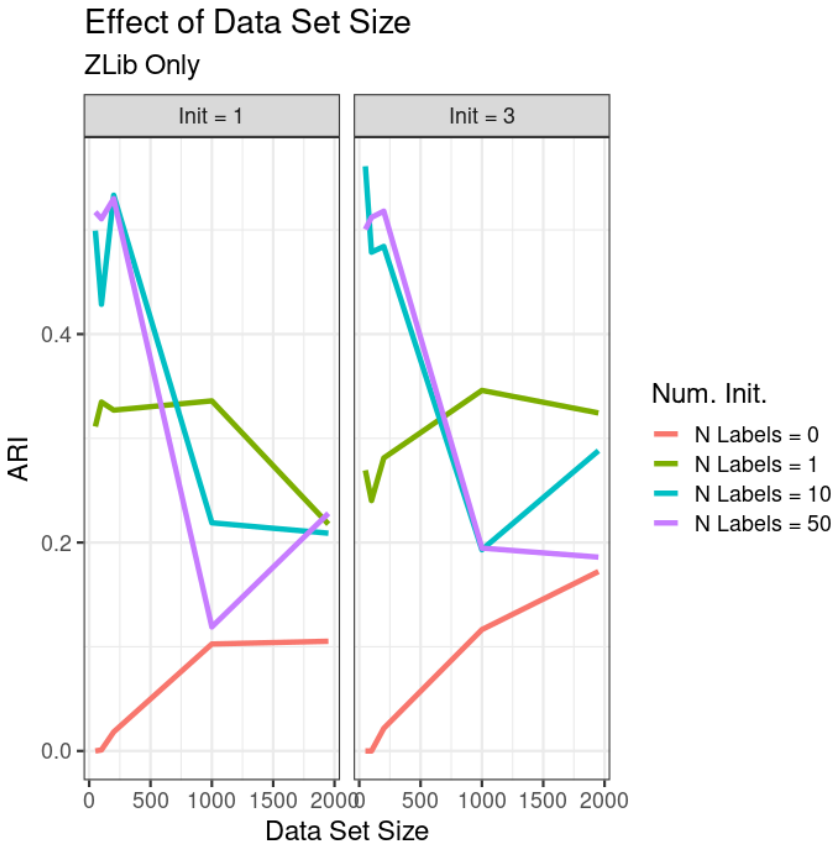


Figure 5-4 Semisupervised zlib performance with varying dataset sizes and varying numbers of labeled samples

Cluster	Text
1	Sandia National Laboratoryboratories multimultional La
1	Sandia Nationa Natio mul ission laboratormltories a mu
2	Los Alamossingle mingle mis a single single mission la
2	Los Labories ion labory Ala single mismingle missiLos
3	The lazy dogqex btqer lazy dogabrox jumps over the qui
3	The q quick brown fox jumps over m brown fox jumps ovx

Table 5-4 Examples of generated sequences with known cluster labels

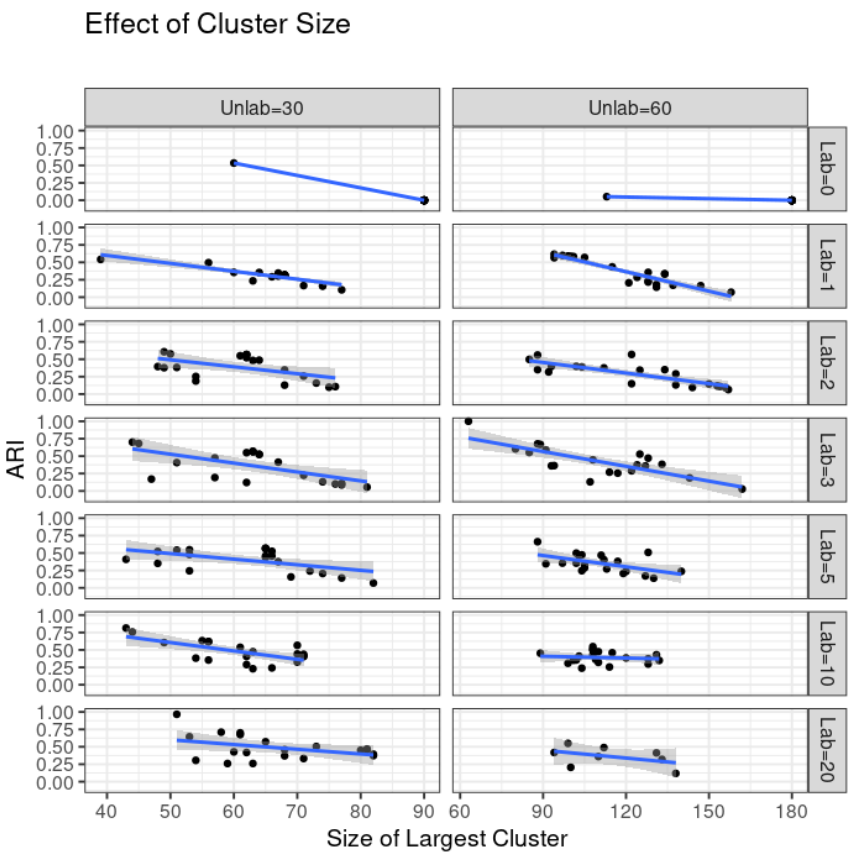


Figure 5-5 Semisupervised zlib clustering performance as a function of the largest cluster size. This plot shows that clustering solutions in which a single cluster engulfs the majority of the observations are associated with poor recovery of true classes.

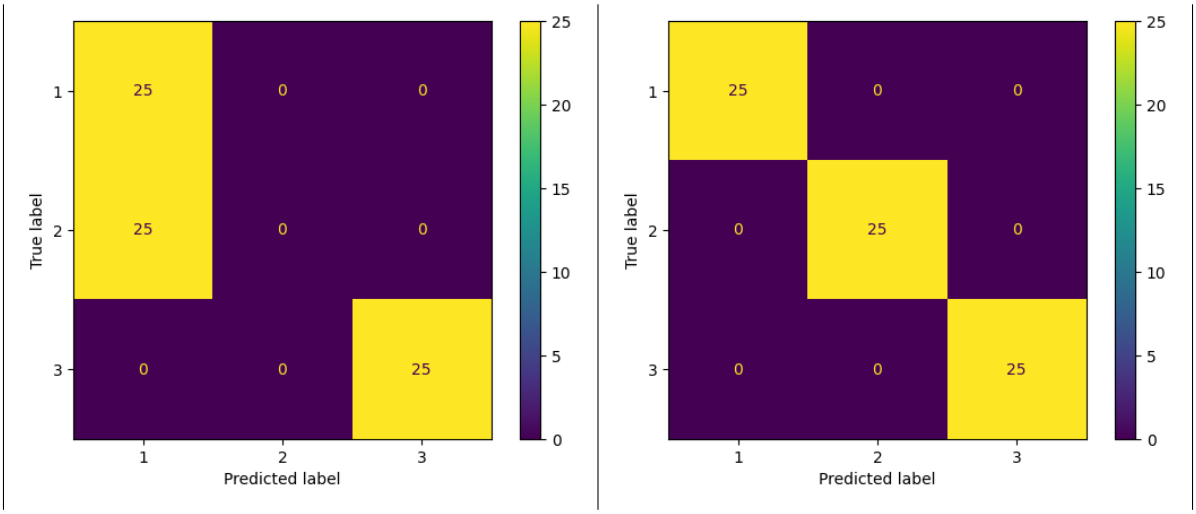


Table 5-5 Confusion matrices of predicted/true cluster labels without any labeled sequences (left) and with 20% of the sequences labeled (right)

5.5. Discussion

These results show feasibility of our clustering technique using two very different compression algorithms. It also shows that using even a small amount of labeled data in the semisupervised framework can greatly improve cluster separation. Future work should continue to evaluate these methods, and further understand the relationship between data structure and favorable choices of compression algorithm.

6. NONLOCAL CA WITH PPM

6.1. Motivation

Prediction by Partial Matching (PPM) with Arithmetic Coding (AC) [25] is an adaptive statistical data compression algorithm based on context modeling and prediction. Given a sequence

$$x = z_0 \dots z_{i-k} \dots z_{i-1} z_i \dots z_n,$$

PPM tries to predict the token z_i by estimating the conditional probability $p(z_i|z_{i-k} \dots z_{i-1})$, where $z_{i-k} \dots z_{i-1}$ denotes the context of size k, also referred to as the depth of the PPM model. For many data types, taking into account the preceding context allows PPM to provide a better estimation of the token probabilities to the encoder (AC), resulting in better prediction, better data compression, and better utility for CA.

However, there are certain instances where the immediately preceding (or local) context is somewhat irrelevant and does not improve PPM performance. For example, in HTML, we often observe the pattern <...>, and in computer code we observe statements like `if...else`. In both of these examples, the local context contained within the `...` can vary from one usage to another, and is not necessarily useful for predicting the > or `else` tokens, i.e., does not necessarily yield better data compression or utility for CA. Instead, it would be preferable to estimate the probabilities for the > or `else` tokens using the more distant, < or `if`, respectively. We refer to this as nonlocal context. The goal in this section is to account for these nonlocal contexts, thereby extending PPM for nonlocal CA.

6.2. Methods

6.2.1. Background

PPM-AC has been used as the data compression algorithm for computing the normalized compression distance (NCD), a pairwise similarity between two data items x_1, x_2. NCD was defined by Eq. (3), but is restated here for convenience:

$$d(x_1,x_2) = \frac{C(x_1x_2) - min\{C(x_1),C(x_2)\}}{max\{C(x_1),C(x_2)\}}, \tag{20}$$

where $C(*)$ denotes the compressed size of $*$ after applying a compression algorithm C, and x_1x_2 denotes x_1 and x_2 concatenated. The key term in this expression is $C(x_1x_2)$, the compressed size of the two items concatenated; the remaining terms are used as normalization factors. Specifically, if x_1 and x_2 are similar, the probabilities learned in x_1 can be used to encode x_2 and the compressed size of the two items concatenated is close to the compressed size of each item alone and $d(x_1,x_2) \approx 0$. If instead x_1 and x_2 are dissimilar, the probabilities learned in x_1 cannot be used to encode x_2 and the compressed size of the two items concatenated is close to the sum of the compressed size of each item alone and $d(x_1,x_2) \approx 1$.

Importantly, PPM-AC allows one to extend the application of data compression algorithms beyond their use in computing a pairwise similarity metric between two data items x_1 and x_2. Instead of compressing a data item x_2 using an adaptive context model of the current data item x_1, denoted by $C(x_1x_2)$, we compress a data item using a fixed, previously-trained context model, denoted by $C_M(x)$. Here, C_M is a trained PPM context model that has learned the probabilities from data items in a training set. This has utility for classification if the training set represents data items belonging to a common class. That is, for PPM context models, $M_1, \ldots, M_m$, each trained on a different class of data items, we can define a decision function

$$\hat{y} = \underset{i=1,\ldots,m}{\operatorname{argmin}} C_{M_i}(x), \tag{21}$$

which assigns item x the class label $\hat{y}$ of the PPM context model that produces the best compression.[9]

In our work, we use a modification of PPM-AC, called n-gram PPM [4]. Instead of applying AC to do the actual compression to obtain $C_M(x)$, n-gram PPM uses the PPM context model directly to compute a score $s(x)$ that indicates how well (or poorly) an item would be compressed using the probabilities in the model M. By skipping the actual compression step, n-Gram PPM is significantly faster than PPM-AC while providing equivalent results.

6.2.2. Description of n-gram PPM

In the original n-gram PPM description, the score for a sequence $x = z_0z_1 \ldots z_n$ is related to the product of the conditional probabilities for each of the tokens in the sequence, and is defined by

$$s(x) = -\log_2 \prod_{i=0}^{n} p(z_i|c_{ik}), \tag{22}$$

where the application of $-\log_2$ converts the probabilities to the compressibility (in bits) for the sequence. Note that the more likely a sequence is to occur, the lower its score. The probability $p(z_i|c_{ik})$ is the conditional probability of z_i given the context $c_{ik} \equiv z_{i-k} \ldots z_{i-1}$, and can be estimated from the training data by

$$p(z_i|c_{ik}) = \frac{N(c_{ik}z_i)}{N(c_{ik}) + 1}, \tag{23}$$

where $N(*)$ denotes the number of times $*$ has been observed in the training set.

This expression in Eq. (23) appears straightforward; it would be if the token and its context of length k had been accumulated by the counter, i.e., observed in the training set. However, this is not always the case. How do you estimate the probability for something that has not been observed? PPM handles this problem by *combining probabilities from different contexts, until the token is observed.* Specifically, starting from the maximum context length k, PPM "escapes" to a

[9] For extensions to this decision rule that can handle prior probabilities of class membership and costs of misclassification, see Section 2.

context of length $k-1$, repeating this escape process until it finds the largest context $j^* \leq k$ where the token is observed. Importantly, the context lengths $j > j^*$ before the token is observed are also accounted for by estimating the probabilities for not observing the token in that context, referred to as the escape probability.

The probability for a token in Eq. (23) is therefore modified to account for all contributions starting from a maximum context length k and escaping to lower contexts until the token is observed in $j^* \leq k$. This probability can be estimated by

$$p(z_i|c_{ik}) = \prod_{j=k,k-1,\ldots,j^*} p_j(z_i), \tag{24}$$

where the index j starts from the PPM model depth k and is decremented until it reaches $j^* \leq k$. The $p_j(z_i)$ are defined by

$$p_j(z_i) = \begin{cases} p(z_i|c_{ij}) = N(c_{ij}z_i)/(N(c_{ij})+1) & \text{if observed in context, i.e., } j = j^* \text{ and } j \neq 0; \\ p(\text{esc}|c_{ij}) = 1/(N(c_{ij})+1) & \text{if not observed in context, i.e., } j > j^* \text{ and } j \neq 0; \\ p(z_i) = N(z_i)/\sum_{a\in A} N(a) & \text{if no context, i.e., } j = 0\ , \end{cases} \tag{25}$$

where "esc" denotes that an escape event occurs, and A is the alphabet of the training set. The last condition is an unconditional probability that accounts for the case where the token is not observed in any context.

Algorithm 1 Compute n-gram PPM token probability.

```
function TOKENPROBABILITY(sequence x, position i, depth k, model)
    p ← list
    for j = k, k−1, ..., 0 do
        if j ≠ 0 then
            α ← model.count(x[i − j : i + 1])
            β ← model.count(x[i − j : i])
        else
            α ← model.count(x[i])
            β ← model.total_count                ▷ count of all tokens occurring
        end if
        if α > 0 then
            p.append(α/(β + 1))
            return product(p)
        else
            p.append(1/(β + 1))
        end if
    end for
end function
```

Pseudocode for obtaining the token probability in Eq. (24) is provided by Algorithm 1. The software library, `Romans`, developed at Sandia National Laboratories, includes code for computing the token probability and subsequent sequence score (Eq. (22)). Also included is code for training multiple n-gram PPM models and utilizing the sequence score for classification.

Algorithm 2 Compute nonlocal n-gram PPM token probability.

function NONLOCALTOKENPROBABILITY(sequence s, position i, depth k, nonlocal model)
    $p \leftarrow$ list
    $p_{escape} \leftarrow$ list
    for $j = k, k-1, \ldots, 1$ **do**
        $\alpha \leftarrow$ model.count($x[i-j], x[i]$)
        $\beta \leftarrow$ model.count($x[i-j]$)
        p.append($\alpha/(\beta+1)$)
        p_{escape}.append($1/(\beta+1)$)
    end for
    if $\max(p) > 0$ **then**
        return $\max(p)$
    else
        $\alpha \leftarrow$ model.count($x[i]$)
        $\beta \leftarrow$ model.total_count ▷ count of all tokens occurring
        return $\max(p_{escape}) \times (\alpha/(\beta+1))$
    end if
end function

6.2.3. Extension of n-gram PPM to Nonlocal Contexts

As discussed in Section 6.1, there are many applications and corresponding data sets where the sequences of interest contain nonlocal dependencies. Consider, for example, sequences with the following structure:

$$x = \ldots <r_{i-n} \ldots r_{i-1}> \ldots .$$

This sequence contains open and closed brackets separated by a random sequence of tokens $r_{i-n} \ldots r_{i-1}$. The best probability for > is not given by the context in the preceding random tokens, but by the < some distance $n+1$ ahead. This is an example of a nonlocal dependence.

Noting that the local context, in this case the random sequence of tokens $r_{i-n} \ldots r_{i-1}$, does not improve the probability for the > token, our general approach is to skip-ahead until we find the nonlocal context with the highest probability and to use that instead. The definition of the score for a sequence $s(x)$ remains the same as the definition provided by Eq. (22), except that the token probability is modified to take the nonlocal context with the highest probability and can be written as

$$p_{nonlocal}(z_i|c_{ik}) = \begin{cases} \max\limits_{j=1,\ldots,k} p(z_i|z_{i-j}) & \text{if observed in any nonlocal context;} \\ \max\limits_{j=1,\ldots,k} p(\text{esc}|z_{i-j}) \times p(z_i) & \text{if not observed in any nonlocal context.} \end{cases} \tag{26}$$

The definitions for $p(z_i|z_{i-j})$, $p(\text{esc}|z_{i-j})$, and $p(z_i)$ are the same as those given in Eq. (25). However, instead of counting the occurrences of z_i given the full context c_{ij}, we count only the pairwise occurrences of z_i given the nonlocal token z_{i-j}. Importantly, because the nonlocal n-gram PPM model is keeping track of only pairwise counts within some distance k, it does not

include the same memory overhead as the standard n-gram PPM model, which needs to keep track of counts for all contexts up to size k. In practice, the standard n-gram PPM model is generally limited to $k < 10$. For the nonlocal n-gram PPM model, the primary computational consideration for setting k is the time required to accumulate all the pairwise counts within some distance of size k. Pseudocode for obtaining the token probability in Eq. (26) is provided by Algorithm 2.

The next sections describe the data sets used to explore the utility of nonlocal n-gram PPM for CA and present the results of our study. We conclude with future work.

6.3. Data Sets

6.3.1. Synthetic Data

To test the nonlocal n-gram PPM context model, a synthetic data set of 1,000 sequences is first constructed. We consider two classes of sequences. Both classes of sequences are of length 25, generated by selecting tokens uniformly at random from the alphabet `A = [a,b,c,d,e,f,g]` and containing a single occurrence of the pattern `<...>`. The only difference between the two classes is the distance separating the < and >. Class 1 is separated by 3 tokens; class 2 is separated by 6 tokens. Examples of class 1 and class 2 are:

1. `eedcee<feg>cccdcbbdf`
2. `cecged<agcdga>dfgaca`

6.3.2. Code Snippets

To evaluate nonlocal n-gram PPM on a real data set, a GitHub Code Snippets Development sample was obtained from Kaggle[10]. This data set is a sample of approximately 5% of the full GitHub Code Snippets data set, also available on Kaggle. The full data set contains over 97,000,000 snippets of code from various GitHub repositories with more than 10,000 stars. For each repository, snippets were extracted from the default branch by going through each text file and extracting 5-line chunks of text every 5 lines. File extensions were used to associate snippets with a programming language. When the language could not be inferred from the file extension, `UNKNOWN` is assigned as the language. The following languages are available: `Bash, C, C++, CSV, DOTFILE, Go, HTML, JSON, Java, JavaScript, Jupyter, Markdown, PowerShell, Python, Ruby, Rust, Shell, TSV, Text, UNKNOWN, YAML`. We choose a random sample of 1,000 snippets from the `C` and `C++` programming languages, and filter to snippets with lengths between between 100 and 200 tokens.

[10] https://www.kaggle.com/datasets/simiotic/github-code-snippets-development-sample

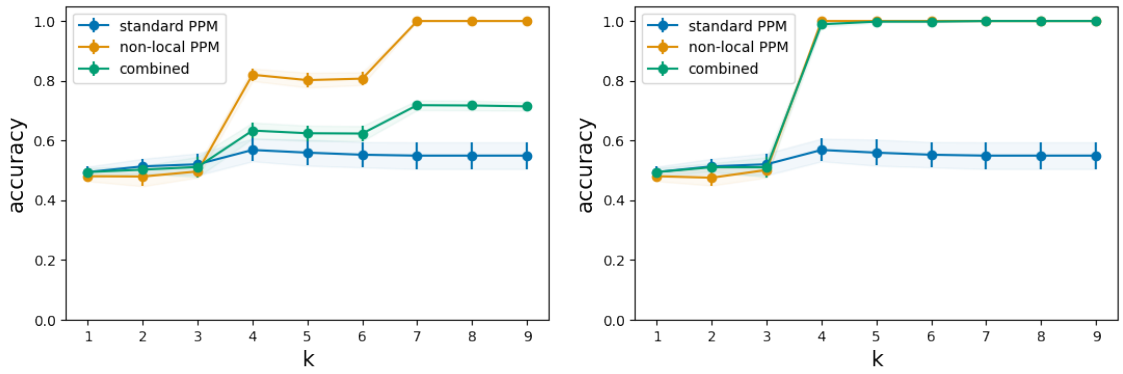
6.4. Results

In addition to comparing the standard n-gram PPM and the nonlocal n-gram PPM classifiers, we consider a third classifier that combines the standard and non-local models by taking the mean of the two context models for each token:

$$s_{combined}(x) = -\log_2 \prod_{i=0}^{n} (p(z_i|c_{ik}) + p_{nonlocal}(z_i|c_{ik}))/2, \tag{27}$$

where $p_{standard}(z_i)$ is given by Eq. (24) and $p_{nonlocal}(z_i)$ is given by Eq. (26). The performances of the three models are evaluated on the synthetic and code snippets data sets. We use a 5-fold cross validation split, training on the large split, to evaluate the performance of our models. An n-gram PPM context model is trained on each class within the train split and applied to sequences in the test split. The sequences are classified according to Eq. (21), using the sequence score in place of the actual compressed size of the sequence. The accuracy is reported as a measure of performance.

A small variation to the nonlocal n-gram PPM is considered. In Eq. (26), the nonlocal context z_{i-j} with the largest conditional probability for observing the z_i is used for estimating $p(z_i)$. However, it may be the case that a token has a high conditional probability for being observed in a context, but the probability of observing the context is in itself not very high. Therefore, in addition to taking the non-local context with the highest conditional probability, we also explore taking the context with the highest lower bound on its confidence interval. Specifically, we use the value for the lower bound of the 95^{th} percentile of the Clopper-Pearson interval, which is a numerical method for calculating the binomial confidence interval and is available in `scipy.stats.beta`. The result using the context with the maximum lower bound for its confidence interval is reported in addition to the result using the context with the maximum probability.

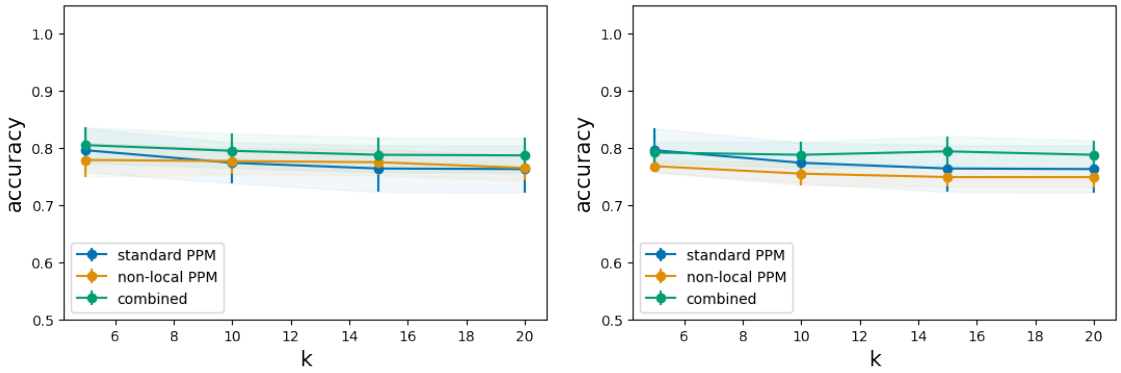


(a) Probabilities **(b) Confidence intervals**

Figure 6-1 Comparison of the performance of standard, nonlocal, and combined n-gram PPM on synthetic data. In (a) and (b), different methods are used for selecting the context for nonlocal n-gram PPM.

Figure 6-1 compares the accuracies of the standard, nonlocal, and combined n-gram PPM context models on the synthetic data set. Using the probabilities (a), it can be seen that the nonlocal

n-gram PPM model out-performs the standard n-gram PPM model for $k \geq 4$ and then jumps to near 100% accuracy for $k \geq 7$. Using the confidence intervals (b), the nonlocal n-gram PPM model does even better, achieving near 100% accuracy for $k \geq 4$. Recalling that this data set contains < and > separated by 3 and 6 tokens for the two classes, it makes sense that at these two context levels, the nonlocal n-gram PPM model is able to use the < token as the context for predicting the > token. In contrast, the standard n-gram PPM model includes the random tokens in the context used to calculate the probability for the > token. As a result, the accuracy of the standard n-gram PPM model is no better than chance (50%). The combined model is able to leverage the discriminatory power of the nonlocal n-gram PPM model while ignoring the standard n-gram PPM model.



(a) Probabilities **(b) Confidence intervals**

Figure 6-2 Comparison of the performance of standard, nonlocal, and combined n-gram PPM on the code snippets data. In (a) and (b), different methods are used for selecting the context for nonlocal n-gram PPM.

Figure 6-2 compares the accuracies of the standard, nonlocal, and combined n-gram PPM context models on the code snippets data set. In contrast to the synthetic data set, nonlocal n-gram PPM does not demonstrate any advantage over standard n-gram PPM. Interestingly, all three models are comparable, and the performance does not improve with larger values of k. This result suggests that the best probabilities for predicting a token are obtained in the local context immediately preceding the token. In general, across a variety of real data sets, we observe this general trend that most of the predictive contexts are local. This result speaks to the general success of the PPM data compression algorithm across a wide variety of sequential data types.

6.5. Future Work

There are some potential avenues for future work. First, while we presented on only one real data set, several real data sets were explored in this study, including graphs, process chains, and natural language text. While nonlocal n-gram PPM demonstrated success on synthetic data, none of the real data sets considered showed similar success. Future work could identify alternative real data sets with more significant nonlocal relationships. Second, the current version of nonlocal n-gram PPM considers only pairwise nonlocal relationships. Future versions could extend nonlocal n-gram PPM to consider other types of nonlocal relationships, including higher order nonlocal

contexts, trends, and seasonal components. A final direction for future work could explore alternative methods for combining the local and nonlocal models, particularly if additional nonlocal models are developed.

7. NONLOCAL CA FOR VIDEO

7.1. Motivation

In Section 6, our approach to handle the nonlocal dependencies was to modify the CA. Here, we present an alternative approach to handle the nonlocal dependencies that is based on modifying the data representation that gets fed into the CA, and not the CA itself.

CA techniques are based on data compression algorithms that operate on data types where the relationships of interest are well-represented by the sequence of bytes; examples include natural language text, binaries, and time series. These techniques do not immediately extend to higher dimensional data types such as images and videos, where the temporal and spatial relationships of interest are not captured by the sequence of bytes. To overcome this limitation, prior work has explored the application of video compression algorithms for computing the NCD between images or frames in a video [8]. Another approach is to extract byte sequences from the video that capture the spatial and temporal relationships of interest. For example, in a FY22-FY23 LDRD, titled "Identifying and Explaining Anomalous Activity in Surveillance Video with Compression Algorithms", the authors converted each spatial region of the video into a temporally ordered byte sequence of pixel values. However, since each byte sequence knows only about the information in that local region of the image, this representation fails to capture any nonlocal spatial relationships.

The motivation for this section is to extend the application of CA to image and video data types with a specific focus on data representations that capture the nonlocal spatial dependencies.

7.2. Methods

Our approach is based on projecting the spatial features (pixels) of the video into a lower dimensional space, where each new direction is some linear combination of the original spatial features. In this way, the new directions represent a non-local combination of spatial features.

7.2.1. Principal Component Analysis

Specifically, we apply principal component analysis (PCA), a linear dimensionality reduction technique. We use the `sklearn.decomposition.PCA()` function, available in python's scikit-learn library.

Consider a video comprised of n_{frames} images or frames. Each frame of the video is described by a 2-d array of pixel values, which we can flatten into a row vector of length n_{pixels}. By repeating this process for the entire video, we obtain an $n_{frames} \times n_{pixels}$ data matrix denoted by $\mathbf{X}$, where each row corresponds to a video frame and each column corresponds to a pixel value.

The principal components of $\mathbf{X}$ can be obtained using its singular value decomposition (SVD), given by $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{W}^T$, where $\mathbf{S}$ is an $n_{frames} \times n_{pixels}$ rectangular diagonal matrix containing the singular values of $\mathbf{X}$, and $\mathbf{U}$ and $\mathbf{W}$ are $n_{frames} \times n_{frames}$ and $n_{pixels} \times n_{pixels}$ matrices containing

the left and right singular vectors, respectively. The SVD allows us to project the data matrix onto a new space as

$$\mathbf{T} = \mathbf{X}\mathbf{W}, \tag{28}$$

where $\mathbf{T}$ is an $n_{frames} \times n_{pixels}$ matrix of principal scores. Further, we can reduce the dimension of the new space by truncating according to singular value, i.e., by retaining the $p \leq n_{pixels}$ columns of $\mathbf{W}$ corresponding to the p largest singular values.

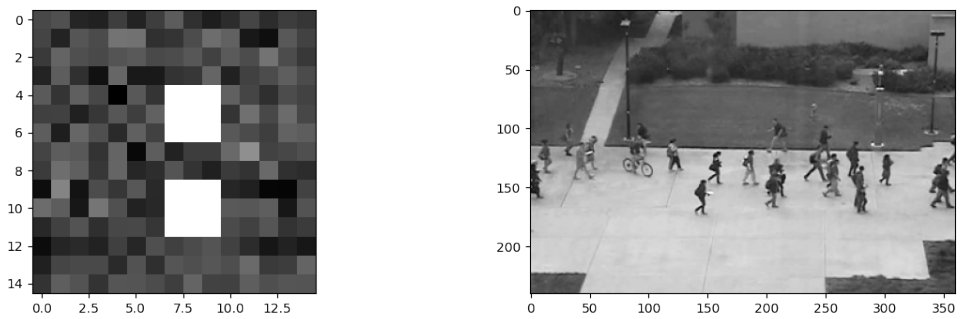
7.2.2. CA: Sliding Information Distance (SLID)

Rather than apply CA on the original representation of the data $\mathbf{X}$, we apply CA to $\mathbf{T}$ defined by Eq. (28). Specifically, the Sliding Information Distance (SLID) is a CA for change point detection[14]. SLID calculates the information distance between two adjacent sliding windows of a byte sequence. Large values in the SLID score indicate a change.

By applying SLID to the columns of $\mathbf{T}$, denoted by t_j for column j, we can investigate how the principal component scores, which can be thought of as a nonlocal linear combination of multiple pixel values, change over time from frame $i = 1, \ldots, n_{frames}$. This requires quantizing the real-valued sequence of scores into bytes:

$$x_j = \text{round}\left(\frac{t_j - \min(t_j)}{\max(t_j) - \min(t_j)} \times n_{bytes} \right). \tag{29}$$

The software library, `Romans`, developed at Sandia National Laboratories, includes code for computing SLID.



(a) Synthetic video **(b) UCSD pedestrian video**

Figure 7-1 Example frames from the two video data sets. The synthetic example shows the relative position of two objects. The UCSD pedestrian example shows an anomalous bicycle.

7.3. Data Sets

7.3.1. Synthetic Video

A synthetic video is constructed with $n_{frames} = 100$; Fig. 7-1a illustrates a single frame. Each frame is of size 15×15 pixels and consists of two 3×3 objects defined by pixel values equal to 100. The pixel values in the frame background are randomly generated from a normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 10$. At the start of the video, the two objects are separated by a vertical distance of $|y_1 - y_2| = 4$. At each frame, both objects move independently one pixel in a random direction, i.e., $y_1 \pm 1$ and $y_2 \pm 1$. In the middle of the video, the separation of the two objects changes instantaneously to $|y_1' - y_2'| = 9$ while the frame-to-frame movement remains unchanged.The goal is to detect this shift in relative position of the two objects using nonlocal CA.

7.3.2. Pedestrian Video

A non-synthetic video is also considered. The UCSD Pedestrians data set [24] is a video data set acquired from a mounted stationary camera overlooking pedestrian walkways on the campus of the University of California San Diego (UCSD). While most of the activity in this data set consists of pedestrians on a walkway, there are also non-pedestrian activities such as people riding bicycles or driving small carts, as well as motion patterns such as pedestrians walking outside the flow of normal traffic. Non-pedestrian activities are considered anomalies. The goal is to detect the anomalies in this data set using nonlocal CA.

7.4. Results

For the synthetic video, we run PCA with dimension $p = 50$. Figure 7-2 illustrates the top 10 principal component directions, after re-shaping the vectors back to images. The first principal component direction represents the relative position of the two objects in the video, the later principal component directions represent the background noise. Figure 7-3 (top) highlights the corresponding scores for the top 10 principal component directions for each frame in the video. The scores for the remaining 40 principal component directions are shown in gray in the background. It can be seen that the score for the first principal component direction shows an abrupt shift at frame 50, indicated by the horizontal red line. This is the frame where the relative positions of the two objects changes.

Figure 7-3 (bottom) shows the result of applying SLID to the sequence of principal component scores. For SLID, we have quantized the sequences into $n_{bytes} = 10$ and use a window size $w = 10$. A large value in the SLID score, particularly for the first principal component score, can be observed at the frame that corresponds to a change in the relative positions of the two objects.

The same analysis is repeated on the real video, also using $p = 50$. Figure 7-4 shows the principal component directions for the real video. In contrast to the synthetic video, none of the principal

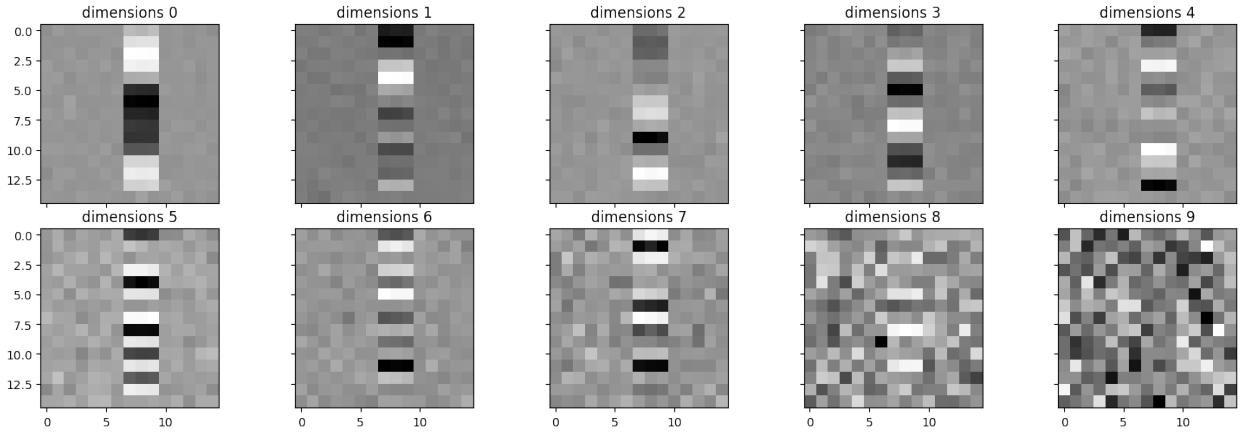


Figure 7-2 Principal component directions for the synthetic video.

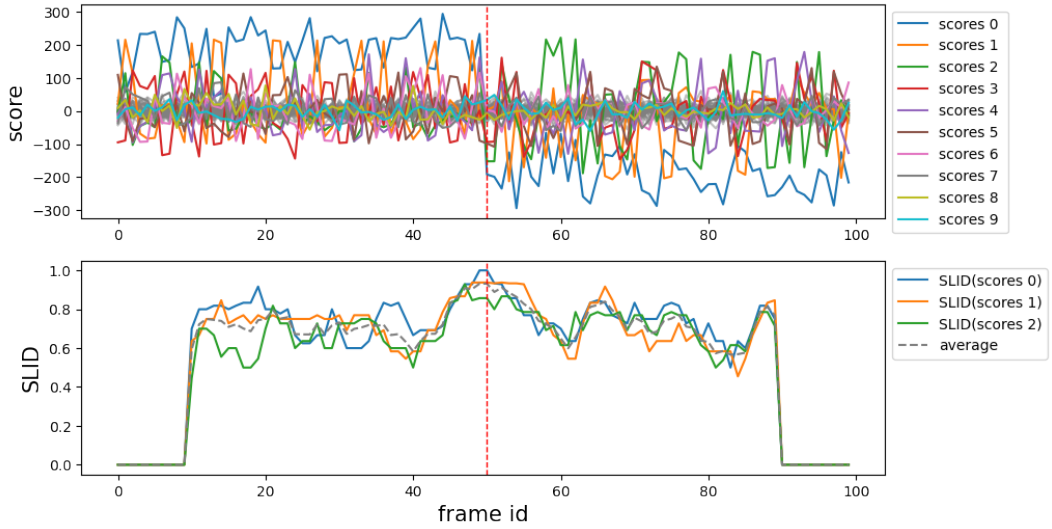


Figure 7-3 Analysis on the principal component scores for the synthetic video.

component directions for the real video captures the object of interest, the bicycle. At best, the components capture the general sense that there are pedestrians on the walkway across the middle of the video frame. As such, in Figure 7-5 (top) it is not surprising that there is not a single principal component score that represents the frames with the bicycle, indicated by the red shaded region. Figure 7-5 (bottom) illustrates that applying SLID to the principal component scores, with $n_{bytes} = 10$ and $w = 10$, fails to identify the frames with the bicycle.

7.5. Future Work

Future work could explore other dimensionality reduction techniques that may be more suitable for videos. For example, the Population Value Decomposition (PVD) method [11] is a technique for decomposing a set of images that takes into account the natural two-dimensional structure of

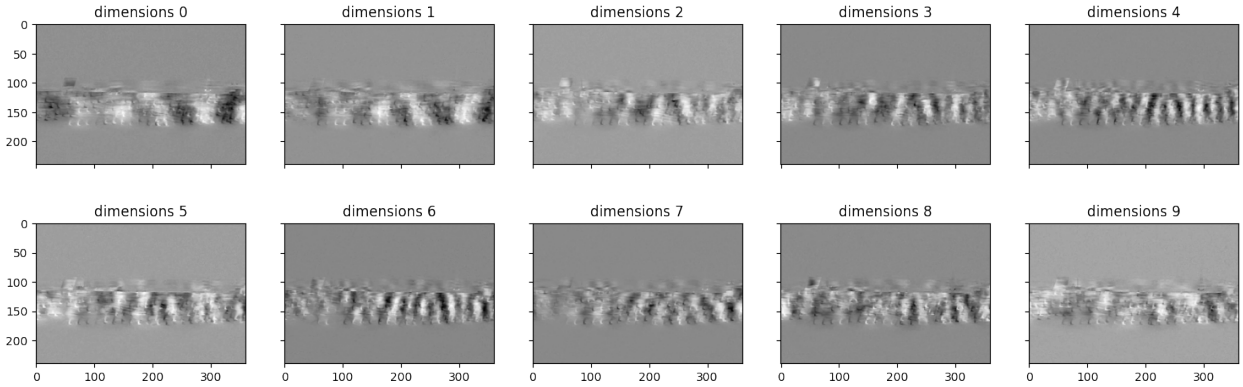


Figure 7-4 Principal component directions for the real video.

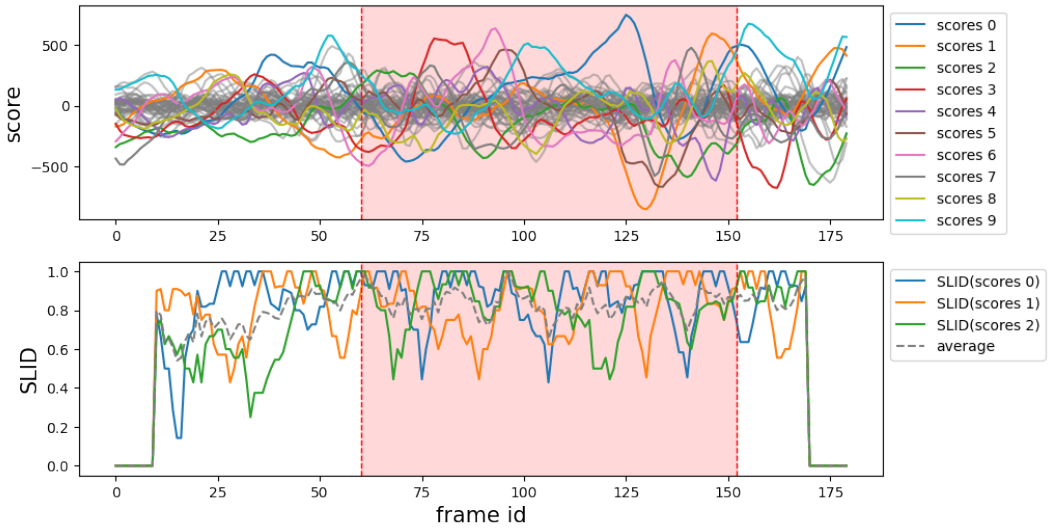


Figure 7-5 Analysis on the principal component scores for the real video.

the images. Additionally, the current implementation of SLID considers sequences for each component independently, i.e., considers univariate byte sequences. The resulting SLID scores are averaged after-the-fact. An alternative approach could be to consider a multivariate version of SLID that can handle the byte sequences for the principal component scores $x_0, x_1, \ldots, x_p$, defined by Eq. (29), simultaneously.

8. BAYESIAN CONTEXT TREES FOR TEXT ANALYSIS

8.1. Main Idea

Compression algorithms involve predicting the next token based on some history; this generally aligns with the goals of Markov models. For some methods (e.g., PPM [10]), this connection is more explicit, whereas for others (e.g., run-length encoding or Lempel-Ziv [34]) this connection is more abstract. Here, we work to extend the capabilities of Markov models to more flexibly accommodate natural structure in a wide range of datasets. The work of [7] extended Markov models to handle *variable length* contexts through the use of trees with branches of variable length. This allows the modeling of both higher- and lower-order relationships with vocabulary size n and maximum order m without growing a full n-ary tree of depth m. The work of, e.g., [20] have extended these initial models to a Bayesian framework, allowing a more flexible tree-building procedure. Here we describe a novel Bayesian framework that allows parameters controlling tree growth to be estimated from the data rather than requiring a user-specified tuning parameter.

8.2. Model

Our goal is to classify documents into different classes, where a "document" is defined to be a series of tokens. Let X_i denote the ith document and let $Y_i \in \{1,\ldots,K\}$ denote the class to which the ith document belongs. We model X_i and Y_i as random quantities.

Given an observed document x, the Bayes classifier for determining the class to which x belongs is given by

$$\arg\max_k P(Y_i = k \mid X_i = x).$$

To implement this classifier we need to figure out how to calculate the posterior probability $P(Y_i = k \mid X_i)$. By the definition of conditional probability,

$$P(Y_i = k \mid X_i) = P(X_i = x \mid Y_i = k)P(Y_i = k)/P(X_i = x).$$

We will assume that the prior probabilities $P(Y_i = k)$ are known for all k. We can ignore the $P(X_i = x)$ since it doesn't affect the classification, because it doesn't depend on k.

For $P(X_i = x \mid Y_i = k)$, there are many different models to consider (e.g., bag of words, bigram models) but we will assume a more complex model. We assume that: (1) the tokens of the document are generated following a variable order Markov chain (e.g., [7]; and (2) for the ith document there corresponds an unobserved context tree T_i and unobserved probabilities $\theta_i = \{\theta_{is}, s \in T_i\}$. For an alphabet of size m, it follows that

$$P(X_i = x \mid Y_i = k) = \prod_{s \in T_i} \prod_{j=0}^{m-1} \theta_{is}(j)^{a_{sx}(j)},$$

where $a_{sx}(j)$ = number of times that token j follows context s in document x.

If we knew T_i and θ_i for all of our documents, we could implement the classifier. However, we don't know them; to solve this problem, we can take a Bayesian approach. We assume that: (1) conditional on $T_i = T$, the θ_{is} are independent across all $s \in T$; (2) conditional on $T_i = T$, for each $s \in T$, $\theta_{is} \mid T_i = T, Y_i = k \sim Dir(1/2, \ldots, 1/2)$ for all k; and (3) $T_i \mid Y_i = k \sim G_k$ for some class-specific prior G_k. Recall that the Bayes optimal classifier is

$$\arg\max_k P(X_i = x \mid Y_i = k) P(Y_i = k). \tag{30}$$

Because we are assuming that T_i and θ_i are random, the likelihood $P(X_i = x \mid Y_i = k)$ equals

$$\int P(X_i = x \mid T_i = T, \theta_i = \theta, Y_i = k) P(\theta_i = \theta \mid T_i = T, Y_i = k) P(T_i = T \mid Y_i = k) dF_{\theta, T \mid Y_i}(\theta, T \mid Y_i = k)$$

Continuing the derivation, from above we have

$$P(X_i = x \mid T_i = T, \theta_i = \theta, Y_i = k) = \prod_{s \in T} \prod_{j=0}^{m-1} \theta_s(j)^{a_{sx}(j)}.$$

By assumption,

$$P(\theta_i = \theta \mid T_i = T, Y_i = k) = \prod_{s \in T} \frac{\Gamma(m/2)}{\pi^{m/2}} \prod_{j=0}^{m-1} \theta_s(j)^{-1/2} = \frac{\Gamma(m/2)^{|T|}}{\pi^{m|T|/2}} \prod_{s \in T} \prod_{j=0}^{m-1} \theta_s(j)^{-1/2},$$

where $|T|$ is the number of leaves of T. Remember that the dimension of θ depends on the tree T through the number of contexts s. Therefore, the likelihood $P(X_i = x \mid Y_i = k)$ equals

$$\int \frac{\Gamma(m/2)^{|T|}}{\pi^{m|T|/2}} \left\{ \prod_{s \in T} \prod_{j=0}^{m-1} \theta_s(j)^{a_{sx}(j) - 1/2} \right\} g_k(T) d\theta dT,$$

where $g_k(T)$ is the probability mass function of $G_k(T)$. We know by the form of the Dirichlet distribution that

$$\int \prod_{j=0}^{m-1} \theta_s(j)^{(a_{sx}(j) + 1/2) - 1} d\theta_s = \frac{\prod_{j=0}^{m-1} \Gamma(a_{sx}(j) + 1/2)}{\Gamma(\sum_{j=0}^{m-1} a_{sx}(j) + m/2)}.$$

It follows that

$$P(X_i = x \mid Y_i = k) = \int \frac{\Gamma(m/2)^{|T|}}{\pi^{m|T|/2}} \frac{\prod_{s \in T} \prod_{j=0}^{m-1} \Gamma(a_{sx}(j) + 1/2)}{\prod_{s \in T} \Gamma(\sum_{j=0}^{m-1} a_{sx}(j) + m/2)} g_k(T) dT.$$

If we knew $g_k(T)$ for all k we could now implement the classifier. However, we don't know the prior distribution. In the past we've been following [20] and assuming that $g_k(T)$ takes the parametric form $\alpha_k^{|T|-1} \beta_k^{|T| - L_D(T)}$. However, we still don't know the value of β_k for the different classes $k = 1, \ldots, K$. We tried to estimate the β_k using maximum likelihood on the training documents in class k but we ran into a lot of computational problems.

To avoid these problems we can **approximate** $g_k(T)$ as follows.

1. Let $\mathscr{T}$ be the set of all possible trees in the sample space. Then

$$\sum_{T \in \mathscr{T}} g_k(T) = 1$$

and

$$P(X_i = x \mid Y_i = k) = \sum_{T \in \mathscr{T}} \frac{\Gamma(m/2)^{|T|}}{\pi^{m|T|/2}} \frac{\prod_{s \in T} \prod_{j=0}^{m-1} \Gamma(a_{sx}(j) + 1/2)}{\prod_{s \in T} \Gamma(\sum_{j=0}^{m-1} a_{sx}(j) + m/2)} g_k(T). \tag{31}$$

2. Now we will assume that $g_k(T)$ only has zero mass on a relatively small number of trees in $\mathscr{T}$. If this number is small enough, then we will be able to calculate the sum explicitly. This is an approximation to the true $g_k(T)$.

3. We need to figure out a reasonable set of trees $\widehat{\mathscr{T}_k}$ where we will assume $g_k(T) = 0$ for $T \notin \widehat{\mathscr{T}_k}$. One reasonable choice: $\widehat{\mathscr{T}} = \{\hat{T}_i : Y_i = k\}$, where $\hat{T}_i$ is the maximum a posteriori estimated tree using the `BCT()` function of the `BCT` package [26].

4. We can now estimate $g_k(T)$ under this approximation.

   a) Our estimate obeys $\hat{g}_k(T)$ equals zero for $T \notin \hat{\mathscr{T}}$.

   b) We can estimate $\hat{g}_k(T)$ for $T \in \widehat{\mathscr{T}}$ using maximum likelihood on all of the training documents in the kth class:

$$\{\hat{g}_k(T) : T \in \widehat{\mathscr{T}}\} = \operatorname*{arg\,max}_{g(T): T \in \widehat{\mathscr{T}}} \sum_{i: Y_i = k} \log \sum_{T \in \widehat{\mathscr{T}}} \frac{\Gamma(m/2)^{|T|}}{\pi^{m|T|/2}} \frac{\prod_{s \in T} \prod_{j=0}^{m-1} \Gamma(a_{sx_i}(j) + 1/2)}{\prod_{s \in T} \Gamma(\sum_{j=0}^{m-1} a_{sx_i}(j) + m/2)} g(T),$$

   where $a_{sx_i}(j)$ are the counts from the ith document. Note that the summation is over $\widehat{\mathscr{T}}$. The number of trees in this set equals the number of documents in class k, so this summation should be manageable.

   c) We can calculate the $\hat{g}_k(T)$ using the EM algorithm. To make the notation cleaner, define

$$f(x_i; T) = \frac{\Gamma(m/2)^{|T|}}{\pi^{m|T|/2}} \frac{\prod_{s \in T} \prod_{j=0}^{m-1} \Gamma(a_{sx_i}(j) + 1/2)}{\prod_{s \in T} \Gamma(\sum_{j=0}^{m-1} a_{sx_i}(j) + m/2)}.$$

   Then each EM iteration is given by

$$\hat{g}_k^{(t)}(T) = \frac{1}{\sum_i I(Y_i = k)} \sum_{i: Y_i = k} \frac{\hat{g}_k^{(t-1)}(T) f(x_i; T)}{\sum_{T \in \widehat{\mathscr{T}}} \hat{g}_k^{(t-1)}(T) f(x_i; T)}.$$

At the end of this procedure we have an estimate of $g_k(T)$ for each class $k = 1, \ldots, K$. We can then plug these into (31) and (30) to implement our classifier.

9. CONCLUSION

In this document we have described numerous advances in the field of compression analytics (CA). We have described a statistical perspective of CA that we have used for improved supervised classification, as well as improved unsupervised and semisupervised analysis. We note that our results are relevant for very general data structures – sequences of categorical tokens. This comprises text, cybersecurity log files, DNA sequences, sequences of musical notes, and many other data structures. While it is possible in some cases to construct embeddings of such data structures in a continuous (real-valued) space, any such transformation can run the risk of discarding information that is useful for the machine learning task at hand. Thus, methods such as CA that can handle sequences of categorical tokens in a natural manner are desirable.

While we have described a number of techniques aimed at extending CA to handle certain classes of nonlocal relationships, there are still many nonlocal relationships we have not covered. In the face of the (possibly uncountably) large number of possible nonlocal relationships imaginable, we have chosen to focus on a set of nonlocal relationships that are relevant to datasets of interest commonly encountered. However, future work could focus on nonlocal relationships not addressed in this report, such as structured nonlocal relationships known to exist across animal chromosomes.

Future work may consider CA techniques informed by recent advances in large language models (LLMs; [32]). Any model that can accurately predict a future token based on a history/context can be used to construct a compression algorithm; therefore, it should be possible to integrate LLMs into a CA framework.

REFERENCES

[1] K. A. Aho and R. T. Bowyer. Confidence intervals for a product of proportions: Application to importance values. *Ecosphere*, 6(11):1–7, 2015.

[2] D. Anastassiou. Genomic signal processing. *IEEE Signal Processing Magazine*, 18(4):8–20, July 2001.

[3] T. W. Anderson. *An Introduction to Multivariate Statistical Analysis, 3rd Edition*. Wiley, 2003.

[4] T. Bauer. NgramPPM: Compression analytics without compression. Technical report, Sandia National Lab (SNL-NM), Albuquerque, NM (United States), 2021.

[5] R. Begleiter, R. El-Yaniv, and G. Yona. On prediction using variable order Markov models. *Journal of Artificial Intelligence Research*, 22:385–421, 2004.

[6] R. J. Buehler. Confidence intervals for the product of two binomial parameters. *Journal of the American Statistical Association*, 52(280):482–493, 1957.

[7] P. Buhlmann and A. Wyner. Variable length Markov chains. *The Annals of Statistics*, 27(2):480–513, 1999.

[8] B. J. L. Campana and E. J. Keogh. A compression-based distance measure for texture. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 3(6):381–398, 2010.

[9] R. Cilibrasi and P.M.B. Vitanyi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

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

[11] Ciprian M Crainiceanu, Brian S Caffo, Sheng Luo, Vadim M Zipunnikov, and Naresh M Punjabi. Population value decomposition, a framework for the analysis of image populations. *Journal of the American Statistical Association*, 106(495):775–790, 2011.

[12] A.P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[13] B. Efron and C. Morris. Stein's estimation rule and its competitors – an empirical Bayes approach. *Journal of the American Statistical Association, Theory and Methods Section*, 68(341), 1973.

[14] R. Field, Jr., T.-T. Quach, and C. Ting. Efficient generalized boundary detection using a sliding information distance. *IEEE Transactions on Signal Processing*, 68:6394–6401, 2020.

[15] Edward W Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *biometrics*, 21:768–769, 1965.

[16] A.H. Foss, M. Markatou, and B. Ray. Distance metrics and clustering methods for mixed-type data. *International Statistical Review*, 87(1):80–109, 2019.

[17] Python Software Foundation. zlib – compression compatible with gzip. https://docs.python.org/3/library/zlib.html, Accessed 9-14-2023.

[18] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Rubin, D. Dunson, and A. Vehtari. *Bayesian Data Analysis*. Chapman and Hall/CRC, 3rd edition, Updated 2020.

[19] L. Hubert and P. Arabie. Comparing predictions. *Journal of Classification*, 2(1):193–218, 1985.

[20] I. Kontoyiannis, L. Mertzanis, A. Panotopoulou, I. Papageorgiou, and M. Skoularidou. Bayesian context trees: Modelling and exact inference for discrete time series. *arXiv*, 2022.

[21] 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.

[22] M. Li, X. Chen, X. Li, B. Ma, and P. Vitányi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.

[23] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag, 2nd edition, 1997.

[24] W. Li, V. Mahadevan, and N. Vasconcelos. Anomaly detection and localization in crowded scenes. *IEEE transactions on pattern analysis and machine intelligence*, 36(1):18–32, 2014.

[25] A. Moffat. Implementing the PPM data compression scheme. *IEEE Transactions on communications*, 38(11):1917–1921, 1990.

[26] I. Papageorgiou, V.M. Lungu, and I. Kontoyiannis. *BCT: Bayesian Context Trees for Discrete Time Series*, 2022. R package version 1.2.

[27] A. Robinson and C. Cherry. Results of a prototype television bandwidth compression scheme. *Proceedings of the IEEE*, 55(3):356–364, 1967.

[28] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[29] D. S. Stoffer and D. E. Tyler. Matching sequences: Cross-spectral analysis of categorical time series. *Biometrika*, 85(1):201–213, 1998.

[30] D. S. Stoffer, D. E. Tyler, and A. J. McDougall. Spectral analysis for categorical time series: Scaling and the spectral envelope. *Biometrika*, 80(3):611–622, September 1993.

[31] E. C. Whisenant, B. K. A. Rasheed, H. Ostrer, and Y. M. Bhatnagar. Evolution and sequence analysis of a human Y-chromosomal DNA fragment. *Journal of Molecular Evolution*, 33:133–141, 1991.

[32] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *arXiv*, 2023.

[33] Jieming Zhu, Shilin He, Pinjia He, Jinyang Liu, and Michael R. Lyu. Loghub: A large collection of system log datasets for ai-driven log analytics. In *IEEE International Symposium on Software Reliability Engineering (ISSRE)*, 2023.

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

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Technical Library	1911	sanddocs@sandia.gov

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.