Statistical Properties of Compression Analytics

Kurtis Shuler [1]

[1]Statistical Sciences
Sandia National Laboratories

Aug 2, 2023

Alexander Foss[1], Christina Ting[1], Travis Bauer[1], Richard Field[1]

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Secu DE-NA0003525.

wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. SAND NO.

What is compression analytics?

- Compression Analytics (CA) uses compression algorithms to achieve machine learning tasks: Anomaly detection, classification, clustering, etc.
  - ⇒ This is accomplished primarily through using file compression and the corresponding file size (in bytes) to measure “similarity,” or shared information, across items
- CA is “featureless” in the sense that it can be applied to any byte stream
  - ⇒ Some compression algorithms are black-box (to varying degrees), some are based on probability/statistical models
- Because basic analysis only requires a compressor, CA is available without any additional software, on virtually all systems, even distributed ones
- **Key Question: Can we say something about the statistical properties of these models when they are used for ML?**

How do we use compression for machine learning?

Use file compression algorithms to produce metrics on bit streams, e.g. file compression ratio, normalized compression distance (Ming et al. 2004) etc.

For clustering:

- Compute pairwise distances between sequences
- Produce/cut a dendrogram, favorite hierarchical clustering algorithm, qualitatively via multidimensional scaling, etc.

For classification:

- “Train” different compression models on corpuses of labeled data
- Assign classes to new documents based on which model achieves the best compression ratio

How many bits does it take to store a symbol?

By the **Source Coding Theorem**, the optimal number of bits required to store each symbol ψ in a stream is

$$-\log_2(p(\psi))$$

The lower bound for file size is nearly achievable by arithmetic coding and other entropy encodings

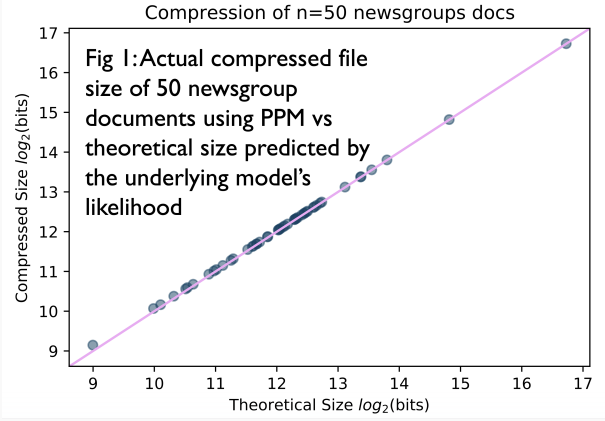


Figure: Comparison of predicted file size using log-likelihood from PPM to actual file size

How many bits does it take to store a symbol?

The compression ratio R_d of a sequence of symbols $\boldsymbol{\psi}_d$ is (approximately) given by

$$R_d = \frac{-\log_2(p(\boldsymbol{\psi}_d))}{S_d}$$

$\rightarrow$ The compression ratio is approximately the (base-2) negative log-likelihood of the sequence $\boldsymbol{\psi}_d$ scaled by the reciprocal of the document size, S_d

$\rightarrow$ $\star$Bit of a simplification, because it ignores the size of the decompression program

$\rightarrow$ **Implies a "psesudo-likelihood" $\log_2(p(\boldsymbol{\psi}_d)) = -R_d S_d$ (even for black-box compression models, i.e. those not based on a statistical model) which can be used for other tasks, e.g. model selection, LR ratio tests, etc.**

- We will show an example using an EM-like algorithm for clustering

Compression Analytics for Classification

Goal: Assign class label $m^\star \in \{1, \ldots, M\}$ to a new, unlabeled document d

- The goal is to assign class label $m^\star \in \{1, \ldots, M\}$ to a new, unlabeled document d
- For each class a different compression/statistical model will be produced ("trained") using the labeled training data
- Let R_{dm} denote the compression ratio coming from applying the m^{th} model to the document

$\rightarrow$ **Relationship between compression ratio and log-likelihood implies the classification rule $m^\star = \underset{m}{\operatorname{argmin}}\, R_{dm}$ is equivalent to $m^\star = \underset{m}{\operatorname{argmax}}\, p_m(\boldsymbol{\psi}_d)$, i.e. assign d to model with the highest likelihood**

Compression Analytics for Classification

There are immediate implications to this connection between CA for ML tasks and the underlying statistical model

- Choosing a compression algorithm that is effective for classification is closely related to choosing better statistical models to represent the different classes
- Optimal compression is related to the effectiveness of the underlying statistical model
  - ⇒ ⋆Ignoring the size of the decompression program
  - ⇒ Including the model size adds a “model complexity” penalty
  - ⇒ For optimal compression purposes, this produces an expression similar to information criterion like AIC/BIC
- The usual caveats of using plug-in parameter estimates and classification by fitting separate models and picking the class with the highest likelihood apply
  - ⇒ Bias/overfitting is may be a problem, classes without much data (e.g. small number of documents in a corpus) may cause difficulties, etc.

KL Divergence

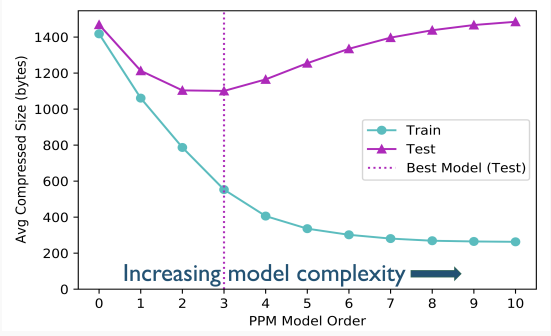
KL divergence gives the expected number of extra bits required to encode a stream when the true (typically unknown) byte stream distribution is $f(x)$, but is encoded using $g(x|\theta)$

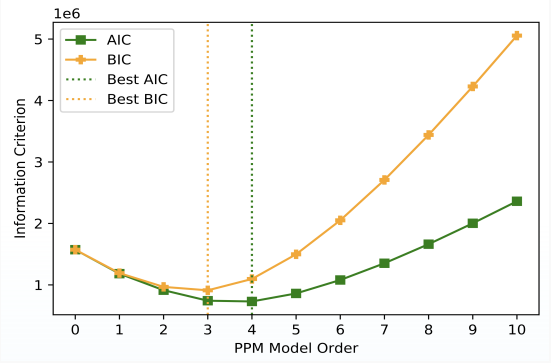
$$I(f,g) = \sum_x \log_2 \left(\frac{f(x)}{g(x|\theta)} \right)$$

- Akaike Information Criterion (AIC) provides an approximately unbiased estimator of expected relative KL information
- For entropy encoders, model selection criteria can be used to select an optimal compression algorithm

Model Selection

Consider the problem of choosing an optimal model for file compression, using either information criterion or cross-validation





Compressed size of 100 test/train documents from the 20-newsgroups dataset

EM Algorithm for CA

Model definition

- 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 $Z_j \in \{1, 2, ..., K\}$
- $Pr(Z_j = k) = \pi_k$ denotes the prior probability that a randomly selected document is drawn from class k.

We assume

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

where C is some sequence model, e.g., a compression model or Markov model. For simplicity, denote $p(x_{ij} \mid z_j, h_{ij}) = q(ijk)$, 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}.

EM Algorithm for CA

E-step: $Q(\Theta \mid \hat{\Theta}^{(t)}) = \mathrm{E}_{\mathbb{Z}}\left[\ell(\Theta; \mathbb{X}, \mathbb{Z})\right] = \sum_{j=1}^{d} \sum_{i=1}^{t_j} \sum_{k=1}^{K} \log(q(ijk)) w_{jk}$

$\Rightarrow$ Weighted log-likelihood expression with weights w_{jk}

M-step: $\hat{\theta}^{(t+1)} = \arg\max_{\Theta} Q(\Theta \mid \hat{\Theta}^{(t)})$

$\Rightarrow$ Not all compression algorithms have natural way to handle weights on input sequences

One way to handle this for the general case is to use “hard” weights,

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

In practice, when applying this algorithm to clustering, rare sequences can cause an item to get ‘stuck’ within a cluster. To combat this effect, we apply the algorithm in a k-fold fashion, leaving out the items of interest in the M-step

Clustering Log Files using EM Algorithm

We apply this algorithm to perform clustering on Mac/Linux log files from LogHub

Linux Log Snippet:

```
Jul 27 14:41:58 combo kernel: CPU: L1 I cache: 16K, L1 D cache: 16K
Jul 27 14:41:58 combo kernel: CPU: L2 cache: 256K
Jul 27 14:41:58 combo kernel: Intel machine check architecture supported.
```

Mac Log Snippet:

```
Jul 3 12:36:01 authorMacBook-Pro kernel[0]: Unexpected payload found for
message 9, dataLen 0
Jul 3 12:36:38 calvisitor-10-105-160-237 corecaptured[33373]:
CCFile::captureLog
```

Clustering Log Files using EM Algorithm

We apply the algorithm to a set of 52 log files, using both a model based compression algorithm, NgramPPM, and using the pseudo-likelihood from zlib

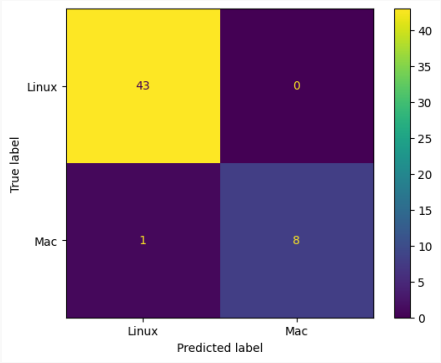


Figure: zlib

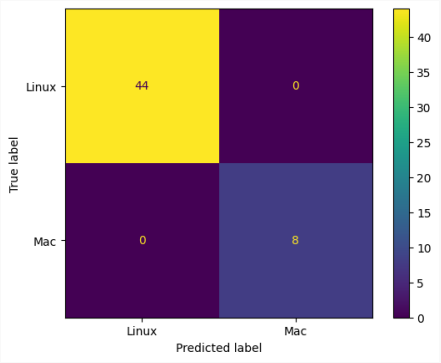


Figure: NgramPPM

Extension: Semi-Supervised Approach

Consider the case where cluster labels Z_j are partially observed

- 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 $J_\ell \cup J_u = \{1, 2, ..., d\}$

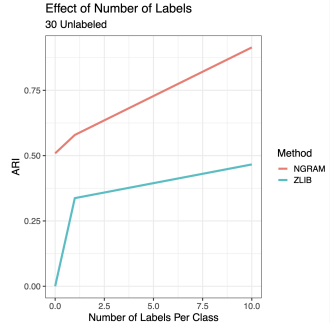
We update the EM algorithm above in light of partially observed labels:

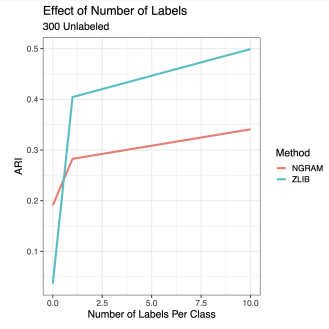
$$
\begin{aligned}
Q(\Theta \mid \hat{\Theta}^{(t)}) &= \mathrm{E}_{\mathbb{Z}} \left[\ell(\Theta; \mathbb{X}, \mathbb{Z}) \right] \\
&= \sum_{j \in J_\ell} \sum_{i=1}^{t_j} \log(q(ijZ_j)) + \sum_{j \in J_u} \sum_{i=1}^{t_j} \sum_{k=1}^{K} \log(q(ijk)) w_{jk}
\end{aligned}
$$

Semi-Supervised Clustering

We apply the semi-supervised clustering approach to Windows/Mac/Linux logs from LogHub, but this time at the line level

- We measure performance using the adjusted rand index
- Adding labels generally increases performance and class separability





References

- Jiang, Zhiying, et al. "'Low-Resource' Text Classification: A Parameter-Free Classification Method with Compressors." Findings of the Association for Computational Linguistics: ACL 2023. 2023.
- Li, Ming, et al. "The similarity metric." IEEE transactions on Information Theory 50.12 (2004): 3250-3264.
- Shannon, Claude Elwood. "A mathematical theory of communication." The Bell system technical journal 27.3 (1948): 379-423.
- Shilin He, Jieming Zhu, Pinjia He, Michael R. Lyu. Loghub: A Large Collection of System Log Datasets towards Automated Log Analytics. Arxiv, 2020.
- Ting, Christina, et al. "Compression analytics for classification and anomaly detection within network communication." IEEE Transactions on Information Forensics and Security 14.5 (2018): 1366-1376.
- 20 Newsgroups: `http://qwone.com/~jason/20Newsgroups/`

Thanks for Listening

Questions?

Backup

The Similarity Metric

Ming et al. 2004 suggested using compression as an approximation to Kolmogorov complexity to produce what they call the **Normalized Compression Distance (NCD)** between two sequences in their paper *The Similarity Metric*

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

- $C(xy)$: Compressed size of the concatenation of x and y
- $C(x)$ and $C(y)$: Compressed size of x/y
- $0 \leq \text{NCD} \leq 1 + \epsilon$, with smaller numbers indicating more similar files

Prediction by Partial Matching

PPM is a compression algorithm introduced by Cleary and Witten[1] uses a series of nested Markov models as the underlying statistical model for entropy coding

- PPM uses the “context” immediately preceding a byte/character/word for more effective prediction
- This method is very effective for compressing English text and is highly competitive as a compression method even today when benchmarked against newer methods like transformer NNs[2]
- NgramPPM produces the same CA results as PPM (but with less computational burden)

[4] Cleary & Witten. (1984). *Data compression using adaptive coding and partial string matching.*
[5] Large Text Compression Benchmark, http://mattmahoney.net/dc/text.html

Prediction by Partial Matching

The probability of a symbol *relative to a model of order k* is given by:

$$p^{(k)}(\psi|\cdot) = \frac{c_k(\psi)}{1 + C_k}, \quad \psi \in A_k,$$

where

- $c_k(\psi)$ is a count of the number of times the character has occured in the current context
- C_k is the total count for characters first predicted by the model of order k
- A_k is the set of characters with counts greater than 0 in model order k, less those with counts greater than 0 in higher order models.

In other words, $p^{(k)}(\psi)$ is *almost* simply the proportion of times that ψ has been seen in the context

PPM: An Example

Say in our PPM model with max order $K = 2$ we have processed the string "abracadabra".

What is the probability, relative to model order $k = 2$, of observing a ψ='a' given context 'br'? Abuse of notation: Call this $p^{(k)}(\text{a}|\text{br})$

- We use the $k = 2$ model because we have seen 'a' in this context
- We have seen 'a' twice in context, so $c_k(\psi) = 2$
- 'a' is the only character predicted by the $k = 2$ model, so $C_k = 2$

$$p^{(k)}(\text{a}|\text{br}) = \frac{c_k(\psi)}{1 + C_k} = \frac{2}{1 + 2}$$

PPM: An Example

Say in our PPM model with max order $K = 2$ we have processed the string "abracadabra".

What is the probability of observing a ψ='b' given context 'ta'?

- We use the $k = 1$ model because we have never seen 'b' in the second order context
- We have seen 'b' twice in context, so $c_k(\psi) = 2$
- Three characters (b, c and d) are predicted by the $k = 1$ model, and $C_k = c_k(b) + c_k(c) + c_k(d) = 2 + 1 + 1 = 4$

$$p^{(k)}(\text{b}|\text{ta}) = \frac{c_k(\psi)}{1 + C_k} = \frac{2}{1 + 4}$$

Likelihood of a sequence

To evaluate the statistical model's relevance to CA we would like the likelihood of an entire sequence (or document).

Let i index a character's position in some document d of length N_d, such that the document may be thought of as an ordered sequence of symbols $\boldsymbol{\psi}_d = \psi_1 \psi_2 \dots \psi_{N_d}$.

We would like an expression for $p(\boldsymbol{\psi}_d)$

Likelihood of a sequence

To get a valid PMF over the alphabet for a given context we must adjust the probabilities to account for the several nested Markov models for a given context.

- This is done by using “escape probabilities” $e_\ell = \frac{1}{1+C_k}$
- In PPM compression actual “escape” symbols are encoded into in the compressed document to indicate which Markov model to use

Likelihood of a sequence

A valid PMF for a symbol ψ_i in context is given by

$$p(\psi_i|\psi_{i-1}, \psi_{i-2}, \ldots, \psi_{i-K}) = p^{(k)}(\psi_i|\psi_{i-1}, \psi_{i-2}, \ldots, \psi_{i-K}) \prod_{\ell=k+1}^{K} e_\ell, \quad \psi_i \in A_k$$

The expression above is equivalent to the last equation in the appendix of Clearly and Witten, 1984

Likelihood of a sequence

Using the conditional independence implied by PPM's Markov structure, the likelihood of the sequence is given by

$$p(\boldsymbol{\psi}_d) = \prod_{i=1}^{N_d} p(\psi_i|\cdot),$$

where $p(\psi_i|\cdot)$ indicates the probability implied by automatically escaping to the Markov model in the hierarchy corresponding to the amount of available context up to that point

From here we can evaluate the sequence likelihoods and compare them to the ML results obtained by CA