SAMPLE SUMMARY WITH GENERATIVE ENCODING

David Banh  
Data Science  
AskExplain  
St Lucia, Brisbane, Australia  
david.b@askexplain.com

ABSTRACT

With increasing sample sizes, all algorithms require longer run times that scales at best logarithmically. A concept that summarises the sample space to reduce the total number of samples into a core set that can be used for regression tasks is introduced. This idea of summarisation is called folding - the technique for projecting data into a lower dimensional subspace, whereas unfolding projects it back into the original space. Results for a prediction task show that information is retained during folding as accuracy after unfolding is still comparable to prediction without summarisation.

GitHub link for summary sampling at:  
https://github.com/AskExplain/summary_sampling_via_folding

GitHub link for Generative Encoder at:  
https://github.com/AskExplain/gcode

1 Introduction

Large sample sizes are important in detecting meaningful effect sizes that are statistically significant. With more features, the covariance between samples contains structured information that can be analysed using machine learning methods. The limitations of methods that treat samples as being independent such as in ordinary least squares, do not account for the immense amount of structure and variance in Big Data that modern day techniques such as neural networks can. An alternative to working with Big Data is proposed by making summaries of the samples.

Sample reduction has been studied as a form of data sketching, or data transformation. Data sketching involves selecting a number of samples that are representative of other data points calculated via distance based neighbouring metrics [1], considering the spatial covering over which the points span [2], or by considering geometric segments over which the sample space covers [3]. Data transformation involves dimensionality reduction to a reduced space by accounting for minimal loss of spectral properties, carrying out analyses in the reduced space, and then projecting learned information back onto the original samples [4] [5].

Summarisation of samples takes the form of data transformation, whereby information is encoded into a reduced space, analysed, and then decoded back into the original sample space. This encoder-decoder technique follows on from the idea of Autoencoders and models the encoding as a transformation and the decoding as a decomposition. By simultaneously treating transformation and decomposition in a single model, the samples and features of the data can be generated via the learned encoding and decoding functions.
2 Method

The aim of Generative Encoding is to integrate data from several datasets, \( Y, X \) and \( A \), into an aligned and reduced dimensional space \( d \) via a transformation. An example of Generative Encoding that is implemented here:

\[
\alpha_{Y_{k,i}}Y_{i,l}\beta_{Y_{l,c}} = \alpha_{X_{k,i}}X_{i,j}\beta_{X_{j,c}} = \alpha_{A_{k,i}}A_{i,d}\beta_{A_{d,c}}
\]

Here, \( k \) and \( c \) are the dimensions of the reduced dimensional encoding. The sample size \( i \) is shared between the datasets \( Y, X, \) and \( A \). However, each dataset contains differing feature sizes \( l, j, d \). For example, this could be of a single cell experiment where each dataset contains the profile of gene expression, protein and chromatin accessibility, respectively for \( Y, X \) and \( A \).

Figure 1: Encoding the dataset using two sets of parameters - an encoding of the sample space and an encoding of the feature space. Multiple sets of data can be aligned to the same space, provided there is a proportional relationship between the two encoding spaces, for example, an equality.

The \( \alpha \) parameter transforms the samples into the same reduced dimensional plane along the sample space. Furthermore, the \( \beta \) parameter transforms the features across the datasets into the same space. The matrix \( \alpha X \) are the encoded features where the sample space has been encoded leaving only the features in the original space. The matrix \( X\beta \) are the encoded samples where the feature space has been encoded leaving only the samples in the original space.

By simultaneously transforming the samples and features into a reduced dimensional code representation, similar to Autoencoders, Generative Encoders learn a set of parameters for both features and samples that can encode both datasets into a reduced dimensional space.

The concept here is a form of Generalised Procrustes with Canonical parameters, similar to the rigid geometric method, Procrustes analysis, which aligns a set of shapes via a rotation transform, and Canonical Correlation Analysis, which seeks to find canonical parameters which maximise the correlation transformation of two data matrices.

2.1 Further details of the Generative Encoder

Inspection of the parameters and how they are linked can be taken a further step. By expressing \( Y, X, \) and \( A \) as a function of the parameters, it is possible to extract the core common element shared between all datasets called the code. For example, let the code be \( Z \) of reduced dimension \( k \) by \( c \).

For \( Y \) the decoded estimate is \( \hat{Y} \),

\[
\hat{Y} = \alpha_{Y_{k,i}}^T Z_{k,c} \beta_{Y_{c,l}}^T
\]

Likewise, \( X \) can be recovered by the decoded estimate \( \hat{X} \)

\[
\hat{X} = \alpha_{X_{k,i}}^T Z_{k,c} \beta_{X_{c,j}}^T
\]

Finally, \( A \) can be recovered by the decoded estimate \( \hat{A} \)

\[
\hat{A} = \alpha_{A_{k,i}}^T Z_{k,c} \beta_{A_{c,d}}^T
\]
These expressions using the code $Z$ can recover the original datasets, with the key important concept that the latent space shared amongst the datasets are given as $Z$.

Furthermore, it can be limiting to consider the datasets as matrices - they can be of any size, including three dimensional objects or tensors.

### 2.2 Properties of the Generative Encoder

**Samples can be encoded**: Sample sizes can be reduced and thus save computational resources when running prediction or nearest-neighbour algorithms. For example, by first reducing the large sample size (several million data points) to a smaller encoded set (several thousand), an algorithm can be run on the smaller set and then decoded back into the original space.

**Data can be recovered and imputed**: Given all data points can be fully expressed by the model, then iterating over updates of the model while simultaneously updating any missing data points can recover the structure of the data and impute missing points. This is similar to the concept of a bootstrap, yet for prediction.

**Transformations can be transferred**: If of similar sample ID the sample parameter $\alpha$ can be transferred, if of similar feature label the feature parameter $\beta$ can be transferred. For this reason, a transform learned to reduced the dimensions from a set of genes can be trained on one dataset, and transferred to the second dataset, provided the gene list is identical.

**Parameters can be joined**: If multiple datasets share the same sample ID, the sample parameter can be shared. Otherwise, if multiple datasets have overlapping feature labels, the feature parameter can be shared. This provides an extra level of interpretability.

### 2.3 Coordinate descent updates for learning parameters

The coordinate descent update to estimate the parameters, iterates through multiple steps outlined by Algorithm 1 until convergence. The full step loops over all datasets, and is run in a while loop within gcode.

#### Algorithm 1 Generative Encoding via Generalised Canonical Procrustes

**Input**: $D_L$ Dataset per modality $L$

**Output**: $\alpha_L$ sample parameters, $\beta_L$ feature parameters, $Z$ code

**procedure** COORDINATE DESCENT UPDATE($D_L$)  

for $L$ in $D_L$ do  
  $\alpha_L = D_L^T(Z\beta_L^T)((Z\beta_L^T)(Z\beta_L^T)^{-1}$  
  $\beta_L = ((\alpha_L^T Z^T(\alpha_L^T Z))^{-1}(\alpha_L^T Z)^T D_L$  
  $Z = (\alpha_L^T\alpha_L)^{-1}\alpha_L^T D_L\beta_L^T(\beta_L\beta_L^T)^{-1}$  
end for

end procedure

### 2.4 Why learn in a subspace?

Learning in a reduced subspace enables a greater level of interpretability when evaluating parameters, and more computational efficiency as the parameters are learned in the smaller reduced dimension. Both of these are benefits from reducing the sample and feature size to an encoded dimension, equivalent to the encoding space.

For example, the code can represent biological information of living cells from different datasets of varying modalities (gene expression, protein etc.). Here, the sample and feature parameters represent a transform of the code of biological cell information into actual realisations of observed samples. This parameterises a full data generating process: encompassing measurement specific noise ($\alpha$, sample space), relevant signal ($\beta$, feature space), and the underlying phenomena ($Z$, the code).

Given the dimensions of both sample and features are encoded down into a reduced dimension, it is possible to learn in the encoded space. The process involves projecting the data into this reduced dimension, after which the main code is learned in sync with the other datasets. The code allows the
learning of both the dimensionally reduced sample and feature spaces after projecting to a shared space informed by the code. These sample and feature projections are parameterised by the matrices: $\alpha$ and $\beta$.

3 Sample Summaries and related work

The optimisation function for a regression with l2-loss, is given by:

$$\|Y - f(X, \theta)\|^2_2$$

where $Y$ represents the data to be predicted, $X$ represents a dataset, and $f$ is a function that is learned with parameters $\theta$ to minimise the l2-loss term.

In Generative Encoding, first learn the parameters for both the feature and sample space of $X$, which represents the original dataset. Following, it is known that the samples are identical between the predicted value $Y$ and the covariates $X$, enabling the encoding of both $Y$ and $X$ along the sample space via $\alpha$.

Given $\alpha$ is the matrix parametric form of a projective function that operates on $Y$ and $X$ simultaneously, the expression can then be interpreted as:

$$\|\alpha Y - f(\alpha X)\|^2_2 = \|g(Y) - f(g(X))\|^2_2$$

It is known that $X = \alpha^T Z \beta^T$ provided by the decomposition of the Generative Encoder, can be considered as the inverse function $g^{-1}$ via $\alpha^T$ (given $\alpha$ is orthogonal, $\alpha\alpha^T = I$, where $g(g^{-1}(I)) = I$), it can be expressed such that $\hat{X} = g^{-1}(Z \beta^T)$ is an estimation of $X$.

Substituting $\hat{X}$ into the loss function:

$$\|g(Y) - f(g(\hat{X}))\|^2_2 = \|g(Y) - f(g(g^{-1}(Z \beta^T)))\|^2_2 = \|g(Y) - f(Z \beta^T)\|^2_2$$

Seemingly, the known function $g$ parameterised by $\alpha$ encodes $Y$ via $g(Y)$ with the purpose of the optimisation algorithm to find a function $f$ that minimises the loss with $g(Y)$.

Now let the function $f$ be any function that gives parameters $\theta$. This can include a random forest regression function that minimises the loss function. The solution using the known $g(Y)$ data, and the known $Z \beta^T$ data (both learned in the Generative Encoder step) can be unfolded with the function $g^{-1}$ that operates on

$$\|g^{-1}(g(Y)) - g^{-1}(f(Z \beta^T, \theta))\|^2_2 = \|Y - g^{-1}(f(Z \beta^T, \theta))\|^2_2 = \|Y - \hat{Y}\|^2_2$$

Furthermore, if $f$ and $g^{-1}$ are linear functions, the composition of $g^{-1}(f(\cdot)) = f(g^{-1}(\cdot))$ enables the expression:

$$\|Y - \hat{Y}\|^2_2 = \|Y - f(g^{-1}(Z \beta^T), \theta)\|^2_2 = \|Y - f(\hat{X}, \theta)\|^2_2$$

The last line follows from the estimate of $X$, given as $\hat{X} = g^{-1}(Z \beta^T) = \alpha^T Z \beta^T$.

Of great importance, notice in the expression $\|Y - f(g^{-1}(Z \beta^T), \theta)\|^2_2$ that $\theta$ is the learned parameterisation of the function $f$ in $g^{-1}(f(Z \beta^T, \theta)) = f(g^{-1}(Z \beta^T), \theta) = f(\hat{X}, \theta)$. This means that the parameters $\theta$ apply to both $Z \beta^T$ and notably $\hat{X}$, the estimation of $X$.

Given an estimation of $Y$ is estimated via $\hat{Y} = g^{-1}(f(Z \beta^T, \theta)) = f(g^{-1}(Z \beta^T)) = f(X, \theta)$, the aim of the original optimisation function is reached.

Note, for reference, $\alpha Y$ and $\alpha X$ are considered to be the summary form of the full sample datasets $Y$ and $X$ respectively. The action of $\alpha$ as a function $g$ is called "folding". The use of $g^{-1}$ operation
Generative Encoding on $g(Y)$, is called the "unfolding" (here it derives the identity matrix). The whole process is called "sample summarisation" and generates "sample summaries".

4 Numerical Results

The results contain two parts - a simulation to reveal the straightforward properties of sample and feature encoding, and, a showcase of sample summarisation by testing on regression algorithms.

- A simulation of graphical shapes undergoing an alignment into the same subspace
- Testing several regression algorithms on the sample summaries of a dataset

4.1 Simulation of shape alignment

Generative encoding shares ideas with Procrustes, a geometric based method designed to match points optimally in a rigid formulaic expression. Generative Encoding can match points similarly, however, it is more flexible as the main goal is to align all datasets involved into the same subspace, rather than aligning one dataset to another.

Here, two datasets are simulated - both visually inspected as heart shape structures when represented in the euclidean plane. When these datasets are fully aligned, it is expected that the parameters transform one dataset to another such that the structures align in axis and match in location [6].

The encoded features align in location, while the encoded samples align along the vertical axis. Then, by simultaneously aligning the encoded samples (aligned along y-axis) and encoded features (aligned at center of shape), the datasets integrate and match point-by-point.

4.2 Sample summary evaluation

Using the Satellite data from the mlbench library in the R statistical language, the categorical feature is excluded leaving 36 variables and 6435 samples. Permutations ($N = 100$) are run to sample 80...
percent of the total samples as the training set, leaving the other 20 percent as the test set. Each feature of the 36 variables are treated as the y-variable in a full run of 36 rotations of the full feature set ($36 \times N = 3600$).

The sample summarisation is run via Generative Encoding (gcode) or via Singular Value Decomposition (SVD) to reduce the total number of samples via folding. Regression with random forests via the ranger package, and the base linear model are used as the optimisation functions to estimate $\hat{y}$.

Both methods for regression: a random forest and a linear model have the Mean Absolute Error loss term measured, as well as the computational run time - with or without summarisation. Individually, the random forest method has an extra metric comparing the correlation of Gini Importance values with or without summarisation, whereas the linear model has the extra metric of correlation for comparing coefficients (weights corresponding to $\beta$ in $Y = X\beta^T$) with or without summarising.

Figure 3: Metrics run on linear regression (left column) and a fast implementation of random forest regression (right column). First row (A, and, B) represents the correlation of the feature weights (Gini Importance in random forests, and beta coefficients in linear regression) learned in each respective function with and without summarisation (more is better). Second row (C, and, D) represents the Mean Absolute Error (less is better), and the third row (E, and F) represents the computational runtime (less is better). Note that two runs of each algorithm was run on the full samples with different seeds - this is to ensure reproducibility.
5 Related Work

5.1 Similarities to Singular Value Decomposition

When the model is expressed as $\alpha Y \beta$ it is similar to a Procrustes Analysis. However, the detailed model outlined in further detail (Section 2.1), $Y = \alpha^T Z \beta^T$ shows an expression depicting Singular Value Decomposition.

5.2 Similarities to Procrustes

In order to align two sets of shapes or points within a dataset, the method centers and scales the points within each set, and then takes a reference set upon which the other set is rotated and fit to. The expression for the centering and rescaling with rotation is: $Y_{i,l} = k_{k,i} X_{i,j} Q_{j,c}$ [7].

5.3 Similarities to Canonical Correlation Analysis

The aim is to find a set of component vectors for each of the reference and the non-reference dataset which maximise the correlation. The generative encoder borrows ideas, by finding sets of parameters for each dataset that maximises the similarity of the structures of the encoded datasets.

5.4 Similarities to Neural Networks and Autoencoders

Deep Learning models are optimisation methods that use a non-linear composition of weights to minimise a loss function [8]. The style of using many weights that scales with both features and samples to analyse images is capable with both Deep Learning and Generative Encoding.

6 Limitations

6.1 Dimension Reduction

When the summary samples are folded and used for dimension reduction followed by an unfolding, the visualisation does not represent the same structures as compared to running on the full sample space. It is recommended that the features are encoded by using the feature parameter for dimension reduction, similar to how an embedding with SVD is done.

6.2 Inverse matrix operation

The size of the encoding space restricts the computational efficiency and quick runtime of the algorithm. Encoding dimensions greater than $R = 1000$ increases the computational complexity. Generally, the greater the encoding dimension, the more information is retained in summary sampling.

7 Extensions

7.1 Modelling the residual

Generative Encoding is currently not a complete statistical model, although it can incorporate residual terms and model the error distribution at various parts of the encoding or decoding steps. A link function can be used to model a variety of distributions.

7.2 Deep Generative Encoder

Similar to Deep Gaussian Mixture Models when a Generative Encoder incorporates residuals within the modelling - a deep version of Generative Encoding can be formulated [?]. The notion of a shared “code” in Generative Encoding represents a latent space which can be modelled similar to a factor analytic model (which the Deep Mixture Model is composed of).
Generative Encoding

8 Conclusion

Making summaries of samples improves computational runtime and efficiency, while also reducing computational memory usage. It can enable modern day machines to run on larger datasets, and handle Big Data with greater ease.

Summarisation via folding is a technique that reduces the total sample size, thus the term of summary samples. Making a summary of the data first involves learning the sample and feature parameters that can reduce the dimensionality of the data via Generative Encoding. The sample encoder is then used to encode the samples: this is the act of folding. While the samples are folded, a regression function can be run - with weights learned transferable to the full sample space. Unfolding treats the sample encoder as a function, and uses the inverse function to reverse the folding of the samples - back into the original space of all samples.

Acknowledgments

David Banh would like to acknowledge:

1 - Dr Alan Huang (University of Queensland, Australia) for being a good advisor for the Generative Encoder project.

2 - Cameron Gordon, Olivia Ou, and, Ryan Deslandes (University of Adelaide, Australia, and University of Queensland, Australia, and University of Queensland, Australia) for consistent support and advice throughout the project.

3 - Alex Alsaffar (University of Queensland, Australia) for discussions on the model of Generalised Canonical Procrustes as well as work on a preliminary Python version of gcproc (a prior version of gcode): [https://github.com/thisismygitrepo/gcprocpy](https://github.com/thisismygitrepo/gcprocpy)

4 - Dr Quan Nguyen (University of Queensland, Australia) for a discussion on the model of corevec (a prior version of gcproc, which is an earlier version of gcode) regarding imputation and alignment in early 2021 [https://github.com/AskExplain/corevec](https://github.com/AskExplain/corevec)

References

[1] Benjamin DeMeo and Bonnie Berger. Hopper: a mathematically optimal algorithm for sketching biological data. *Bioinformatics*, 36:i236–i241, 07 2020.

[2] Mostafa Rahmani and George K. Atia. Spatial random sampling: A structure-preserving data sketching tool. *IEEE Signal Processing Letters*, 24(9):1398–1402, 2017.

[3] Brian Hie, Hyunghoon Cho, Benjamin DeMeo, Bryan Bryson, and Bonnie Berger. Geometric sketching compactly summarizes the single-cell transcriptomic landscape. *Cell systems*, 8(6):483–493.e7, Jun 2019.

[4] Joel A. Tropp. *Column Subset Selection, Matrix Factorization, and Eigenvalue Optimization*, pages 978–986. Proceedings. Society for Industrial and Applied Mathematics, Jan 2009. 0.

[5] Petros Drineas, Michael Mahoney, and Senthilmurugan Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-row-based methods. volume 4110, pages 304–314, 09 2006.

[6] Meng Xu. Procrustes analysis. 2016.

[7] Colin Goodall. Procrustes methods in the statistical analysis of shape. *Journal of the Royal Statistical Society: Series B (Methodological)*, 53(2):285–321.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. [http://www.deeplearningbook.org](http://www.deeplearningbook.org)