A scalable approach to prediction from multiple data sources with varying availability

Stephen Miller PhD, Felipe Avila, Lewis Jordan PhD, Greg Kwiatkowski. Equifax Inc.

Abstract

As the volume of data available grows, credit decisions should become more accurate benefiting lenders and borrowers alike. However, when that data is drawn from multiple sources with different levels of coverage or availability, modelers face a challenge to make use of all the data that is available in each instance. Common approaches to multi-data modelling fall into two camps: we either fit a separate model for each possible combination of data sources, or we rely on missing flags and trust the model (such as a GBM) to fit all possibilities. Both approaches have weaknesses as the number of data sources increases: the former approach leads to an exponential increase in the number of models, and exponential decrease in the sample size for each; the latter may overfit to the combinations of data sources represented in the training data, and fail to generalize well to new combinations.

We introduce a probabilistic approach that enables a single model to integrate information from multiple data sources in a principled way, that will support better generalization while avoiding the need for multiple models. The model allows information to be added from each data source in any order and as they arrive, and allows local explanations to be generated in terms of the data sources available for each decision. We also show how the model may be extended to incorporate new data sources without needing to refit to all existing data sources from scratch. We show benchmarked results integrating standard credit attributes with 3 new data sources representing alternative finance and telco & utilities. Our model is implemented in Tensorflow and hence suitable for rapid cloud deployment and monitoring.

Contents

Introduction

We consider the problem of building models to make predictions of an unknown quantity from multiple data sources, where it may be the case that only a subset of those data sources is available for any given observation, either in model training or for prediction. The distinct combinations of data sources present per-observation in the training data may not cover all the possible combinations that may be seen at prediction time. The challenge is to implement a model that can make statistically robust predictions from any combination of the training data sources, including where that particular combination does not exist or is not well represented in the training data, and where the total number of data sources may be large. In particular, we wish to avoid an exponential increase in model complexity as the number of data sources increases linearly.

We are specifically interested in a consumer credit origination use case, where the quantity to be predicted is a good/bad credit outcome defined over some period of time after account opening, and the data sources represent previous credit performance and other relevant and permissible information related to the applicant at the time of application. We will therefore refer to the data subject as an individual or applicant where it is natural to do so, although our solution could be applied to other use cases. This use case also imposes an explainability requirement on the model, which we shall achieve through monotonicity constraints.

Examples of data sources for the consumer credit origination use case include:

- Traditional credit file data
- Alternative data sources providing coverage of other sources of borrowing, e.g. alternative finance
- Closed user group data, available for decisioning only in certain circumstances or for use by certain lenders
- Bank transactional data
- Social media or other non-financial data that may be permissible for use in some geographies

Each of these data sources may or may not be available for a given application, either at the point of decisioning or in the historic data used to create a training sample. It is worthwhile noting some reasons why data may have varying availability:

- Consent. For example, the use of bank transactional data in credit decisioning in the UK requires explicit consent from the applicant. This consent may or may not be sought: if the

data already available is sufficient for a lender to make a clear decision to accept or reject the application, there is no need to burden the applicant with the consent process.

- Contractual restrictions. This applies to closed user group data, which may be shared under conditions that restrict its use for some or all purposes to members of the group.
- Coverage. While it may be more correct to treat e.g. a number of accounts variable as zero if there is no data available from a particular source, it can also be useful to treat the data as missing if, for example, there is known to be incomplete coverage of a market segment. For example, if we observe no utilities accounts at an address, and data coverage is known to be incomplete, it may make more sense to treat the status of utilities as unknown rather than to assume no account exists.
- Opt-out. By this we mean the decision of the lender to not use a particular data source that may have an increased cost associated with it.
- Keying and linking. The process of associating data with an individual is imperfect, especially for individuals with a small data footprint, e.g. young people and those new to the country. Therefore, even in the case of traditional credit file data, the absence of any matched account data may not necessarily indicate that no such accounts exist.

Given all of the above reasons why data may be unavailable, it is clear that the presence or absence of a particular data source cannot be assumed to be independent of the content of that data, nor of the applicant's credit and financial situation or their application risk. In other words, the data sources must be considered *missing not at random* (MNAR). Our solution will explicitly account for the presence or absence of each data source.

Before presenting our solution in detail, we note alternative solutions to the problem and their drawbacks.

The first common solution is segmentation. This requires a separate model to be built for each possible combination of data sources. The models may be built from scratch, or methods may be used to combine models built separately for each data source to cover the combinations. The methods include fusion models, where a new model is formed from a combination of existing model scores, and embedded models, where new predictors are combined with an existing score.

Fusion and embedded models have theoretical flaws when the different data sources have information in common (see below), but they suffer primarily from the curse of dimensionality: as the number of data sources in scope grows linearly, the number of possible combinations or segments grows exponentially. So, for example, if we have 3 data sources there are only 7 possible combinations (not including the empty set); but if we have 10 data sources there are 1,023 possible combinations. Building and monitoring this number of separate models is a substantial operational burden. In addition, as the number of segments grows exponentially, so the expected number of observations in each segment shrinks exponentially for a fixed sample size. Model training will be compromised by the shrinking sample sizes.

The second common solution can be considered a 'black box' approach. Rather than build a separate model for each combination of data sources, we could fit a single model taking variables from all the data sources as input (with default values used for missing data sources) as well as a mask, consisting of binary indicators for the presence or absence of each data source. Such a model could take any common form, such as a neural network or GBM, and necessary constraints could be imposed, such as monotonicity in the values of the input variables (but not the mask). We would hope that the model would learn to accurately reflect the information contained in the mask as well as the available data sources, to make accurate predictions for any combination of data sources.

While this second approach does not suffer from a proliferation of models, the curse of dimensionality is still present. In fitting a non-linear response to the mask, a non-linear model is highly likely to over fit to the values of the mask represented in the training data. Without a statistically principled approach to incorporating the information contained in the mask, generalization to combinations of data sources that are either absent or infrequent in the training data is likely to be weak.

Finally, we note the possibility of carrying out data integration at the variable level. In this approach, data from multiple sources is aggregated into a single set of independent 'multi-data' variables that are then used as inputs to the model. For example, an aggregate 'number of trades' variable could be defined as a sum of separate 'number of trades' variables from each data source. From the model's point of view, all the variables are always available. This approach can work well when the data sources are of a similar type (e.g. different sources of credit tradeline data) and especially when the absence of a data source has a clear quantitative meaning (e.g. no tradelines exist of this type). However, this approach is not applicable when the different data sources contain fundamentally different types of information that cannot be summarized by the same independent variables.

Requirements and solution in brief

Based on the discussion above, we establish the following requirements for a predictive model to be trained to predict from multiple data sources with varying availability:

- Single model. Predictions given any combination of data sources should be obtained from a single model, without fragmentation into multiple segments for different combinations.
- Statistically principled. To improve the chances of good generalization to unseen combinations of data sources, the model should account for the availability of each data source in a statistically principled way.
- Missing-not-at-random. The model should account for the relationship between the presence or absence of each data source and the observed variables and outcome.
- Explainability. For use in a credit application decisioning, the model should support the generation of accurate local explanations. Specifically, it should be possible to impose a monotonic relationship between the model prediction and the values of the observed variables.

Our solution can be described as a deep hybrid generative/discriminative model (Kuleshov & Ermon, 2017), using a monotonically constrained partial inference net encoder (Ma, et al., 2019) for amortized variational inference, a monotonic scorer and an unconstrained decoder. The model is fit using doubly stochastic gradient descent as in the seminal (Kingma & Welling, 2013).

More meaningfully, we model common information from the model data sources via a multi-dimensional latent variable z, which is assumed to be distributed standard normal in the population. Neural network modules are fit to capture the relationships between z, the observed variables x_i, and the outcome variable y, taking into account the mask m indicating which of the data sources are present as in (Collier, Nazabal, & Williams, 2020). The values of the separate data vectors x_i, one for each source, the mask m and the outcome y are assumed to be conditionally independent given z, and it is this assumption that allows us to incorporate information from the available data sources and the mask in a principled way.

Model formulation

Our model represents information from multiple data sources in a multi-dimensional latent variable z, whose value for an individual can be calculated (with uncertainty) given the values of the available

data sources and the mask. The binary good/bad outcome is then predicted by a function of z. We think of z as capturing the key factors related to credit behaviour that are represented across the data sources, and that are relevant to predicting the outcome. The model can be interpreted as a non-linear factor analysis, with the addition that the factor values are required to be predictive of an outcome variable. Each data source (or its absence) provides information about the value of the latent factor variable.

Motivation

To illustrate the need for a multi-dimensional latent variable, we consider a contrived example that also shows the weakness of combining one-dimensional scores from different sources. Suppose we have two data sources, each consisting of two binary independent variables. We will call the variables x_{11}, x_{12}, x_{21} and x_{22}. Now suppose x_{11} and x_{21} capture information unique to data sources 1 and 2 respectively, but x_{12} and x_{22} represent the same information and in fact are perfectly correlated.

Suppose we fit a score based on data source 1 of the form $s_1 = 400 + 20.x_{11} + 20.x_{12}$, and a score based on data source 2 of the form $s_2 = 400 + 20.x_{21} + 20.x_{22}$. A reasonable combined score would reflect the correlation between x_{12} and x_{22}, and might take the form $s_c = 400 + 20.x_{11} + 20.x_{21} + 20.x_{12}$ or $s_c = 400 + 20.x_{11} + 20.x_{21} + 20.x_{22}$, these being equivalent.

However, suppose we only observe the values of s_1 and s_2, and these are both equal to 420. What should be the combined score? If $x_{11} = x_{21} = 1$ and $x_{12} = x_{22} = 0$, then we should calculate $s_c = 440$. On the other hand, if $x_{11} = x_{21} = 0$ and $x_{12} = x_{22} = 1$ then we should have $s_c = 420$. The per-data source scores do not contain enough information for us to combine them correctly.

On the other hand, suppose we calculate a 3-dimensional vector from each data source. From data source 1 we return $(x_{11}, 0, x_{12})$ while from data source 2 we return $(0, x_{21}, x_{22})$. We now have enough information to calculate the combined score accurately. The 3 components of the returned vectors correspond to 3 common factors represented across the 2 data sources. In practice the returned vectors from each data source may consist of non-linear combinations of variables, not only single values, and may also contain information about the uncertainty, expressed as precision or covariance, associated with those estimates of the factor values.

Mathematical formulation

Figure 1 shows the structure of the model as a probabilistic graphical model (PGM).

- x'_r for $r = 1, \ldots, R$ are vector-valued latent variables representing the ”true” values of the partially observed independent variables, one per data source.
- x_r for $r = 1, \ldots, R$ are the observed ”corrupted” values of the same independent variables. If data source r is available, then $x_r = x'_r$; otherwise x_r is set to some default value D_r, such as zero.
- m is a binary mask vector of dimension R. Each coordinate m_r is equal to one if data source r is available, and zero otherwise. Therefore we have $x_r = m_r x'_r + (1 - m_r)D_r$ where D_r is the default value for data source r.
- y is a dependent variable. It may be discrete or continuous and may be multi-valued. In the credit risk use case, y is a binary variable representing good/bad credit outcome.
- z is a latent variable with standard multivariate normal distribution of some dimension K. For each observation, z captures the fundamental characteristics of the entity (individual borrower, in the credit risk use case) that are observed through the independent and dependent variables and the mask. The values of the mask m, the partially observed

independent variables x_r and the dependent variable y are conditionally dependent on the value of z.

The arrows in Figure 1 represent the conditional dependency relationships in the model. In particular, note that x_r', m and y are conditionally independent given z. In other words, they are statistically related only through their common dependence on z. In addition, though it is not clear in the diagram, we assume the separate coordinates of the mask are conditionally independent given z.

In order to complete the mathematical description of the model, we need to specify the conditional probability distributions $p(x_r'|z)$, $p(m|z)$ and $p(y|z)$.

We use neural networks to flexibly model the parameters of the conditional pdfs $p(x_r'|z)$, $p(m|z)$ and $p(y|z)$. Treating x_r' as a continuous variable we use $p(x_r'|z) = \mathcal{N}(\mu_r(z), \Sigma_r)$ and treating y as a binary variable, we use $p(y = 1|z) = \mu_y(z)$, where $\mu_r, r = 1, \dots, R$ and μ_y are vector valued functions and Σ_r is a constant diagonal covariance matrix. In practice we may use a single neural network to model the concatenation $\left(\mu_1(z), \cdots, \mu_R(z)\right)$, and a separate neural network with sigmoid activation on the output layer to model $\mu_y(z)$. In the credit risk use case, we can constrain the function $\mu_y(z)$ to be monotonic.

Alternative specifications exist. For example, we can model binary coordinates x_r^i using $p\left(x_r^i = 1\middle|z\right) = \mu_r^i(z)$ with a sigmoid transformation. Nominal and ordinal categorical variables can be also be handled by using appropriate conditional probability distributions for $p\left(x_r^i = 1\middle|z\right)$. If the dependent variable is continuous, we can specify an appropriate conditional distribution for y, such as a Gaussian $p(y|z) = \mathcal{N}\left(\mu_y(z), \sigma_y^2\right)$.

Where it is convenient, we will use the single vector valued variable x to represent the concatenation of the observed vectors x_r, one for each data source, that include default values for unavailable data sources. We will also use $\mu(z)$ and Σ to represent the concatenations of the mean vectors and diagonal covariance matrices $\mu_r(z)$ and Σ_r.

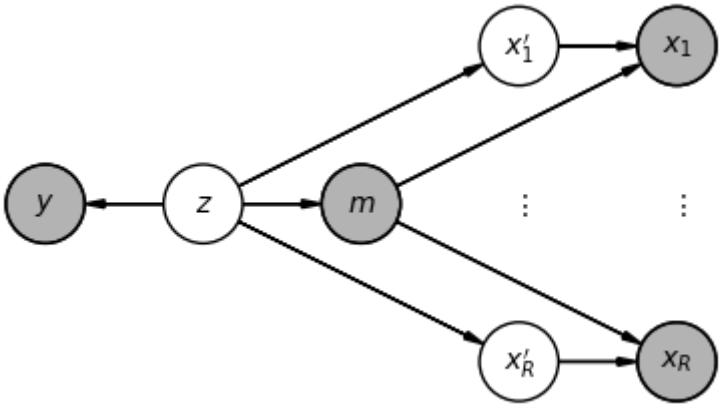

Figure 1: Hybrid model as a PGM

Model fitting by variational inference.

Given the mathematical description of the model in the previous section, the likelihood function for a single observation of the triple (x, m, y) is given by an integral over the density of *z*:

$$p(x, m, y) = \int p(x, m, y|z)p(z)dz = \int p(x, m|z)p(y|z)dz$$

(Collier, Nazabal, & Williams, 2020) note that, if we always have access to the mask m, we could optimize the model weights to maximize a variation on the conditional likelihood $p(x, y|m)$ rather than the full likelihood $p(x, m, y)$. They introduce an additional latent variable z_m that influences only the mask, which is integrated away in $p(x, y|m)$. However, we have found better generalization results when fitting $p(x, m, y)$, forcing the model to learn $p(m|z)$ and hence incorporating information from the mask in z. We omit z_m in our model.

Following the hybrid model approach of (Kuleshov & Ermon, 2017), we allow for a weight to be applied to the likelihood of y, forming a *multi-conditional likelihood*:

$$l(x, m, y; \lambda) = \int p(x, m|z)p(y|z)^{\lambda}\, dz$$

The parameter λ allows us to assign higher importance to the prediction of y than to the reconstruction of x in the model loss function. Typically, x will be a vector of dimension in the hundreds or thousands, and y will be a scalar. Without this importance weighting, y would be treated as just another variable in a generative model. It is the importance weight λ that allows the model to perform predictively.

Analytical evaluation of this integral is intractable in general, and we wish to avoid simulation over the high-dimensional z, both in training and prediction. Instead, following (Kuleshov & Ermon, 2017) we seek to maximise a *variational lower bound* to the log-likelihood given by

$$log\ \ l(x, m, y; \lambda) \geq E_{q(z|x,m)}(\lambda \log\ p(y|z)\ +\ \ \log\ p(x, m|z)) - D_{KL}(q(z|x, m)\ ||\ p(z))$$

Here, $q(z|x, m)$ is a variational approximation to the true conditional distribution $p(z|x, m)$, the latter being intractable to compute in general. The model therefore consists of three modules, each implemented via neural networks:

- A *decoder* that computes $p(x, m|z) = p(x|z, m)p(m|z)$ via the conditional mean $\mu(z)$ and fixed covariance Σ of $p(x'|z)$, accounting for the default values D_r for missing data sources.
- An *encoder* that computes $q(z|x, m)$. In the simplest formulation, $q(z|x, m)$ is assumed to be multivariate normal, and the conditional mean and variance are computed.
- A *scorer* that computes $p(y|z)$. In the case that y is binary, this uses a sigmoid activation to output $p(y = 1|z)$.

This architecture is represented in Figure 2.

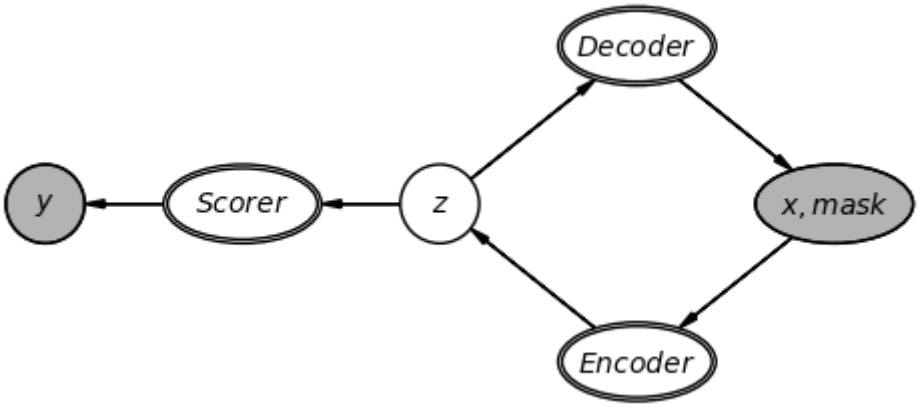

Figure 2: Modular representation of hybrid model

The model is fit using doubly stochastic gradient descent (DSGD) to optimize the weights of the modules to minimize a loss function that includes the negative of the variational bound expressed above, and may also include additional regularization losses. That is, we minimize

$$E_{q(z|x,m)}(-\lambda \log p(y|z) - \log p(x, m|z) + D_{KL}(q(z|x, m) \,||\, p(z)) + \cdots$$
$$= E_{q(z|x,m)}\big(-\lambda \log p(y|z) - \log p(x, m|z) + \log q(z|x, m) - \log p(z)\big) + \cdots$$

where the ellipses represent any additional losses, such as regularization terms. The expectation term is evaluated stochastically by sampling from the posterior $q(z|x, m)$, and the reparameterization trick is used to obtain a gradient. The gradient is aggregated over mini-batches of data before each gradient step is taken. Hence the term doubly stochastic: the first source of stochasticity is sampling from $q(z|x, m)$, and the second is the use of mini-batches. Any standard gradient descent algorithm can be used to implement the gradient step, such as Adam.

Partial inference net

The purpose of choosing a latent variable formulation for the model is to enable information from the different data sources and the mask to be integrated in statistically principled way, enabling better generalization to unseen combinations of the data sources. This is achieved in two ways: first, through the mathematical specification in which the data source values and mask are conditionally independent given the value of the latent variable; and second through the use of an encoder structure that respects this assumption.

Our encoder uses a *partial inference net* (PIN) of the type introduced by (Ma, et al., 2019) with the addition that mask information is also incorporated to account for the MNAR assumption. The PIN encoder is distinguished by an aggregation layer (an additive layer in our case) in which information from each available data source is aggregated before being transformed to compute the conditional

mean and covariance of z. Unavailable data sources provide zero information, except through the mask. This additive formulation can be motivated in two related ways, as explained below. Note that we do *not* impose either of these specific interpretations of the partial inference net's function; we allow an arbitrary (though possibly monotonically constrained) non-linear transformation of the output of the additive layer.

Additive log-probability

Applying Bayes' rule to the conditional independence formula for x and m given z

$$p(x, m|z) = \prod_r p(x_r, m_r|z)$$

we obtain

$$p(z|x, m) \propto \prod_r p(x_r, m_r|z)\, p(z)$$

where the constant of proportionality is $1/p(x, m)$, which does not depend on z. On the log scale, we therefore have the additive formula

$$log\ p\ (z|x, m) = \sum_r log\ p\ (x_r, m_r|z) + log\ p(z) + C$$

Now suppose we choose a collection of representative points z^j in the sample space for z. Defining for simplicity $p\left(z^j\middle|x, m\right) = p\left(z = z^j\middle|x, m\right)$, we can express the conditional mean of z given x approximately as a weighted average, in terms of the probability distribution at these points:

$$E(z|x, m) \cong \frac{\sum_j z^j p\left(z^j\middle|x, m\right)}{\sum_j p\left(z^j\middle|x, m\right)} = \frac{\sum_j z^j exp\ \left(log\ p(z^j) + \sum_r log\ p(x_r, m_r|z^j)\right)}{\sum_j exp\ (log\ p(z^j) + \sum_r log\ p(x_r, m_r|z^j))}$$

As a function of x and m, this is a non-linear transformation of the j terms $\sum_r log\ p\left(x_r, m_r\middle|z^j\right)$, each being a sum of contributions from each data source x_r and its associated mask coordinate m_r. This can be implemented in an additive layer in a neural network, whose inputs are j-dimensional vectors, one for each data source. Under this interpretation, those j-dimensional inputs are expected to learn an approximation to $log\ p\left(x_r, m_r\middle|z^j\right)$ at j representative values of z.

Similarly, for the conditional covariance of z given x and m, we have:

$$E(zz^T|x, m) \cong \frac{\sum_j z^j \left(z^j\right)^T p\left(z^j\middle|x, m\right)}{\sum_j p\left(z^j\middle|x, m\right)}$$

which can be calculated in terms of the same inputs.

Mean and precision

Another interpretation of the additive layer arises naturally when the model is linear-Gaussian. That is, if the marginal $p(z)$ is Gaussian (as it is in our case) and $p(x_i|z)$ is Gaussian for independent variables x_i with mean an affine function of z (which it is not in our case, not least because we have binary mask variables), then we have:

$$\log p\ (z) = -\frac{1}{2}(z - \mu_0)^T \Sigma_0^{-1} (z - \mu_0) + C$$

$$\log p\,(x_i|z) = -\frac{1}{2}(f_i(x_i) - L_i z)^T \Sigma_i^{-1}(f_i(x_i) - L_i z) + C$$

where for additional flexibility, we have allowed for an arbitrary normalising transformation $f_i(x_i)$ of x_i.

Both $log\ p(z)$ and $log\ p(x_i|z)$ are quadratic in z, so $log\ p\ (z|x)$ is a sum of quadratics, which is again quadratic in z, so $p(z|x)$ is Gaussian. Its mean and covariance may be calculated by some standard linear algebra manipulations, and take the form

$$Cov[z|x] = \left(\Sigma_0^{-1} + \sum_i L_i^T\, \Sigma_i^{-1} L_i\right)^{-1}$$

$$E[z|x] = Cov[z|x]\left(\mu_0 + \sum_i L_i^T\, \Sigma_i^{-1} f_i(x_i)\right)$$

We see that the precision (inverse covariance) for z given x, and the product precision × mean, are both given by a sum of contributions from each of the observed variables. In this case, an interpretation of the role of the additive layer and its inputs is to learn, for each observable variable, a contribution to the conditional precision and precision weighted mean for z given x.

PIN encoder structure

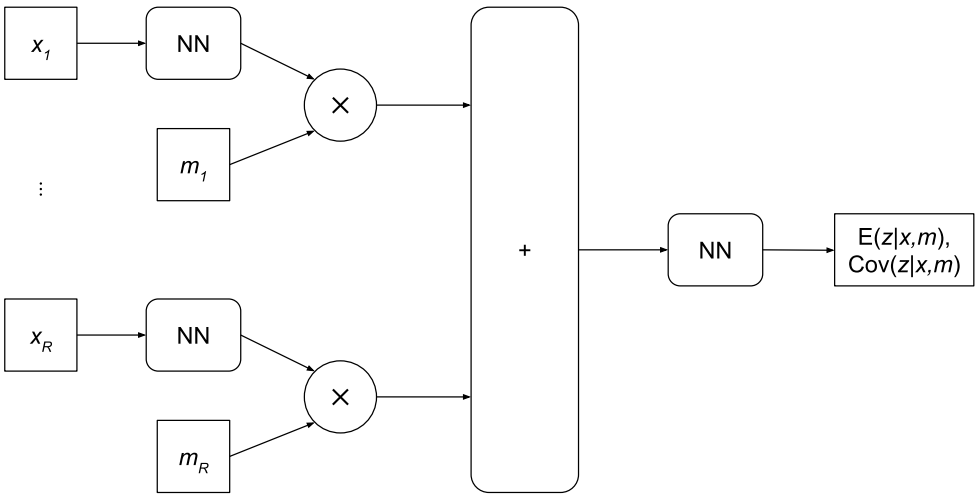

Figure 3: PIN encoder structure

Figure 3 shows the architecture of the PIN encoder. For each data source x_r there is a neural network taking x_r as input, with linear activation in output layer. The output of this neural network is then multiplied by the binary mask coordinate m_r. These outputs are then added, before being transformed by another neural network to compute $E(z|x,m)$ and (a representation of) $Cov(z|x,m)$. Note that the latter should be positive-definite, which is achieved by a bijective transformation. For example, Tensorflow Probability parameterizes the covariance of multivariate

normal distributions by lower triangular matrices with non-zero diagonal, corresponding to the Cholesky decomposition of the covariance.

This formulation avoids redundancy. There appears to be no specific contribution from the mask, but an additive contribution when m_r is equal to one can be absorbed into the output layer of the neural network operating on x_r, while a baseline value corresponding to the case $m_r = 0 \; \forall \; r$ (never observed in the data) is absorbed into the first layer of the final neural network. Thus, a linear function of the mask is implicitly added.

In our models, the neural networks that transform x_r each have one hidden layer, with sigmoid activation, and non-negativity kernel constraints on both the hidden and output layers. There are in fact two separate neural networks converting the output of the additive layer into $E(z|x,m)$ and a diagonal standard deviation $SD(z|x,m)$ respectively. Each has one hidden layer with sigmoid activation. The neural network for $E(z|x,m)$ uses non-negativity kernel constraints, ensuring the mean of z is monotonic in x. The neural network for $SD(z|x,m)$ is not constrained; due to the way we generate predictions, the conditional covariance of z is not subject to any constraints. A softplus transformation is applied to ensure the standard deviation is positive.

Decoder and scorer structure

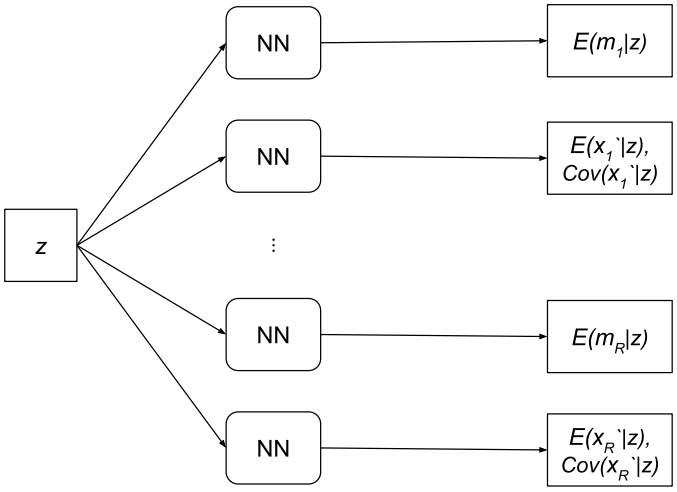

Figure 4: Decoder structure

Our decoder has a simple structure as shown in Figure 4: separate neural networks calculate the sufficient statistics of each data source vector x_r and each mask coordinate m_r given z. In our model, each network has one hidden layer with sigmoid activation. We compute a diagonal standard deviation for the latent x_r' given z. The decoder is not required to be monotonic. Since $x_r = m_r x_r' + (1 - m_r) D_r$ it is straightforward to calculate $p(x, m|z)$ from $p(x', m|z)$ to apply the correct loss function.

Our scorer is a single monotonically constrained neural network containing two hidden layers with sigmoid activation, which outputs $E(y|z)$ for the binary outcome variable y.

Monotonic prediction

Given a trained model, we will wish to generate predictions of the outcome variable y given inputs x and m. Strictly, we should calculate

$$E(y|x,m) = \int E(y|z)p(z|x,m)dz$$

or the variational approximation

$$E(y|x,m) = \int E(y|z)q(z|x,m)dz$$

A set of sufficient conditions for the second integral to be monotonic in x, for fixed m, is that $E(y|z)$ be monotonic in z, the conditional mean $E(z|x,m)$ be monotonic in x and the conditional covariance $Cov(z|x,m)$ be a function only of m. There are various strategies for approximating this integral:

- Evaluation of $E(y|z)$ only at the conditional mean $E(z|x,m)$;
- Simulation over $q(z|x,m)$, using fixed draws from a standard Gaussian distribution to ensure monotonicity is retained over multiple evaluations;
- Taylor series approximation around the conditional mean $E(z|x,m)$, though this is not guaranteed to be monotonic, at least in theory.

In experiments not detailed here, we have found negligible difference between the predictions obtained using these three approaches. We have therefore adopted the first approach, evaluation at the conditional mean $E(z|x,m)$. This has the benefit that the constraint on the covariance is not required: it is only necessary that $E(y|z)$ be monotonic in z and the conditional mean $E(z|x,m)$ be monotonic in x.

Experimental results

Real data and benchmark models

We tested our approach using a production modelling dataset, used for the development of the multi-data modelling product Equifax OneScore. The data consists of US consumer observations at the point of account origination, with attributes drawn from 4 data sources:

- Regular consumer credit data - A
- Closed user group telephone and utilities data - N
- Two alternative finance data sources – T and X

The data is split into train, validation and test samples with observation dates from May ’17 to Feb ’18; and an out of time (OOT) sample with observation dates from April ’19 to Dec ’19. The performance window for the OOT sample overlaps with the COVID-19 pandemic. There are a total of 13.6M observations in the in-time (train/val/test) samples. All 15 non-empty combinations of the 4 data sources are well represented in the data. Sampling weights are available and used throughout the model training and validation. The outcome variable is a 12-month performance flag defined over all non-mortgage trades.

Our results are compared with the existing production models developed using this data. There are 22 total production models:

- 7 GBM models for combinations A, N, X, NT, NX, TX and NTX.

- A logistic regression model for T. This segment suffered from poor OOT generalization when more complex models were used. Only a small number of inquiry-based attributes take non-default values for the majority of observations.
- 14 'fusion' models, 2 for each segment AN, AT, AX, ANT, ANX, ATX and ANTX. The fusion method combines the scores for segment A and the given combination of N, T and X (e.g. NT) to produce a score for the combined segment (e.g. ANT). There are 2 fusion models for each segment, because a thick/thin split is used based on the number of trades in source A.

Variable selection and treatment were carried out as part of the production modelling project, and we re-used the selected independent variables and treatments. Where there was any discrepancy between segments (e.g. attributes selected for the T model vs. the NT model) we used the union of all variables selected for the production models. The thick/thin flag is not used in our hybrid model.

In a from-scratch model development using our approach, we recommend that variable reduction be carried out for each data source separately, and then the union of all variables selected from all data sources be used for the model development. This way, highly correlated variable within a data source can be reduced, but correlated variables in different data sources will be retained. Since only a subset of the data sources may be available for prediction, it makes no sense to carry out cross-data source variable reduction.

Model details

We fit two versions of our model: one including the three non-traditional data sources N, T and X, and another including all four data sources A, N, T and X. Both models implement the PIN encoder, decoder and scorer structures described above. We tuned model hyperparameters to maximise fit, while avoiding *posterior collapse* (see e.g. (Lucas, Tucker, Grosse, & Norouzi, 2019)) which can occur in variational models where some dimensions of the conditional $q(z|x,m)$ collapse to be non-informative. We tuned the following hyperparameters:

- The dimension of the latent variable z, which is set to 20. Results indicated that a further increase could continue to improve test fit.
- The size of the hidden layers in the encoder, decoder and scorer modules. These are all set to 80. Again, a further increase could continue to improve test fit.
- The importance weight λ on the predictive loss for y. This is set to 10^3. Higher values led to posterior collapse.

The models were specified in Tensorflow using Tensorflow Probability and Keras. Doubly stochastic gradient descent (DSGD) was used to fit the models, using an Adam optimizer with learning rate 10^{-3} and a batch size of 256.

Results

Table 1 to Table 4 show hold out KS statistics for the 3 data source and 4 data source hybrid models compared to the production benchmarks.

Table 1: KS statistics for 3 data source models: Test

Segment	Production model KS	Hybrid model KS	Hybrid model +/-
Overall	51.0	52.0	+ 1.0
N	42.0	42.8	+ 0.8
NT	34.6	35.1	+ 0.5
T	16.0	19.8	+ 3.8
X	16.8	16.4	- 0.4

XN	26.9	27.8	+ 0.9
XNT	21.3	21.3	0.0
XT	16.5	16.8	+ 0.3

Table 2: KS statistics for 3 data source models: OOT

Segment	Production model KS	Hybrid model KS	Hybrid model +/-
Overall	48.0	48.4	+ 0.4
N	41.4	42.2	+ 0.8
NT	35.4	36.4	+ 1.0
T	16.4	18.6	+ 2.2
X	18.2	18.1	- 0.1
XN	30.0	30.8	+ 0.8
XNT	24.3	24.3	0.0
XT	19.6	20.1	+ 0.5

Table 3: KS statistics for 4 data source models: Test

Segment	Split	Production model KS	Hybrid model KS	Hybrid model +/-
Overall	-	60.0	60.3	+ 0.3
A	-	60.1	59.3	- 0.8
N	-	41.9	42.5	+ 0.6
NT	-	32.1	32.0	- 0.1
T	-	16.2	19.2	+ 3.0
X	-	17.9	17.4	- 0.5
XN	-	25.6	26.4	+ 0.8
XT	-	17.0	16.9	- 0.1
XNT	-	20.6	20.4	- 0.2
NA	**Thick**	64.0	64.4	+ 0.4
NA	**Thin**	49.6	49.4	- 0.2
NA	**Combined**	64.0	64.4	+ 0.4
NTA	**Thick**	48.3	48.7	+ 0.4
NTA	**Thin**	30.7	30.6	- 0.1
NTA	**Combined**	47.4	47.7	+ 0.3
TA	**Thick**	46.2	46.0	- 0.2
TA	**Thin**	29.5	28.7	- 0.8
TA	**Combined**	42.8	42.3	- 0.5
XA	**Thick**	37.8	37.7	- 0.1
XA	**Thin**	24.2	24.9	+ 0.7
XA	**Combined**	34.4	34.1	- 0.3
XNA	**Thick**	39.6	40.3	+ 0.7
XNA	**Thin**	25.0	25.2	+ 0.2
XNA	**Combined**	37.8	38.3	+ 0.5
XNTA	**Thick**	28.8	29.0	+ 0.2
XNTA	**Thin**	19.7	20.7	+ 1.0
XNTA	**Combined**	27.8	28.1	+ 0.3
XTA	**Thick**	26.4	27.2	+ 0.8
XTA	**Thin**	20.1	20.8	+ 0.7

XTA	Combined	25.0	25.6	+ 0.6

Table 4: KS statistics for 4 data source models: OOT

Segment	Split	Production model KS	Hybrid model KS	Hybrid model +/-
Overall	-	58.2	58.3	+ 0.1
A	-	58.3	57.9	- 0.4
N	-	41.6	41.8	+ 0.2
NT	-	33.7	33.8	+ 0.1
T	-	16.3	17.3	+ 1.0
X	-	17.8	17.3	- 0.5
XN	-	27.5	27.6	+ 0.1
XT	-	18.9	19.3	+ 0.4
XNT	-	22.7	22.2	- 0.5
NA	Thick	62.7	63.0	+ 0.3
NA	Thin	51.4	51.4	0.0
NA	Combined	64.3	63.5	- 0.8
NTA	Thick	51.0	51.5	+ 0.5
NTA	Thin	36.9	37.0	+ 0.1
NTA	Combined	47.0	51.4	+ 4.4
TA	Thick	48.6	49.0	+ 0.4
TA	Thin	35.7	35.2	- 0.5
TA	Combined	42.5	46.8	+ 4.3
XA	Thick	41.2	41.1	- 0.1
XA	Thin	27.7	26.6	- 1.1
XA	Combined	33.8	37.2	+ 3.4
XNA	Thick	44.0	44.6	+ 0.6
XNA	Thin	29.5	29.0	- 0.5
XNA	Combined	38.2	43.0	+ 4.8
XNTA	Thick	33.9	33.8	- 0.1
XNTA	Thin	24.5	25.1	+ 0.6
XNTA	Combined	28.0	33.0	+ 5.0
XTA	Thick	32.6	32.8	+ 0.2
XTA	Thin	25.0	25.5	+ 0.5
XTA	Combined	25.5	31.2	+ 5.7

Our aim in building the hybrid model is to *match* the performance of the per-segment models using a single model; we do not necessarily expect to *exceed* the performance of the per-segment production models. With that in mind, we note that the hybrid model KS statistics often *do* exceed the per-segment model statistics, and are never more than one point behind. This validates the ranking power of the hybrid model as being competitive with the per-segment models. Some of the OOT uplift in the combined segments is remarkable. The single hybrid model is generalizing better than the combination of separate thick and thin models over the combined populations.

The following Figures are alignment plots for the hybrid models, by data source segment. These were obtained by binning predicted good rate into 200 equal width bins. Each point represents the combination of 20 consecutive bins, with the sets of 20 overlapping between points. This reduces noise in the plots while retaining detail. The axes show predicted and actual good rates for each

combined bin, converted to a log-odds scale. On the horizontal axis, log-odds has been further transformed to a score scale, with 40 points representing a doubling of odds. The dark dashed line is the diagonal, while the grey dashed lines show a band of +/- 10 points on the score scale.

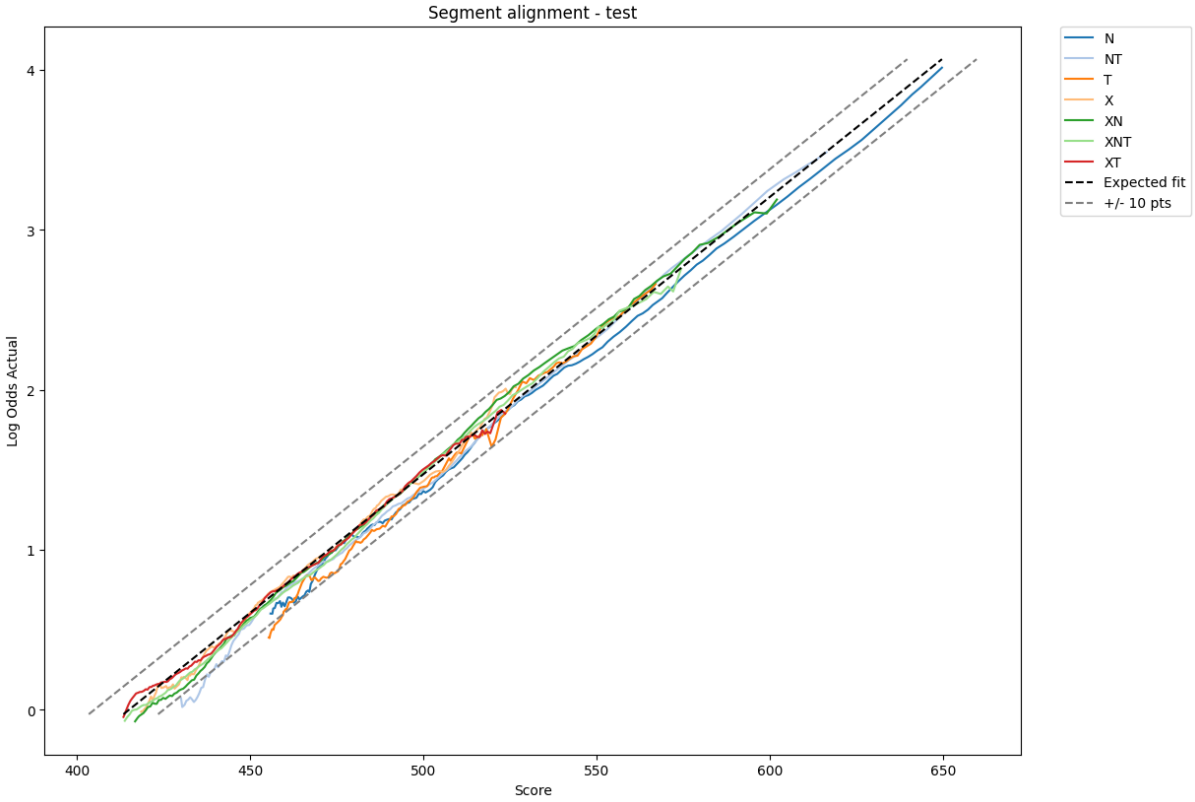

Figure 5: Alignment for 3 data source models: Test

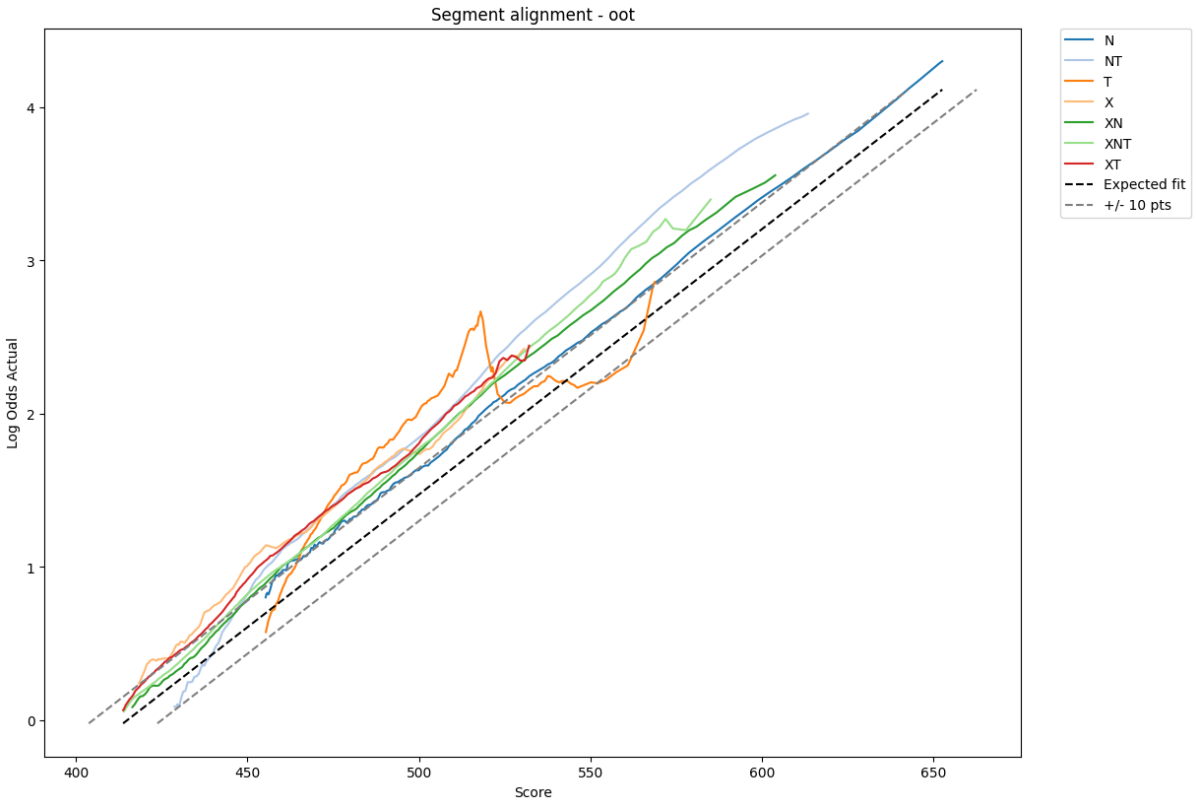

Figure 6: Alignment for 3 data source models: OOT

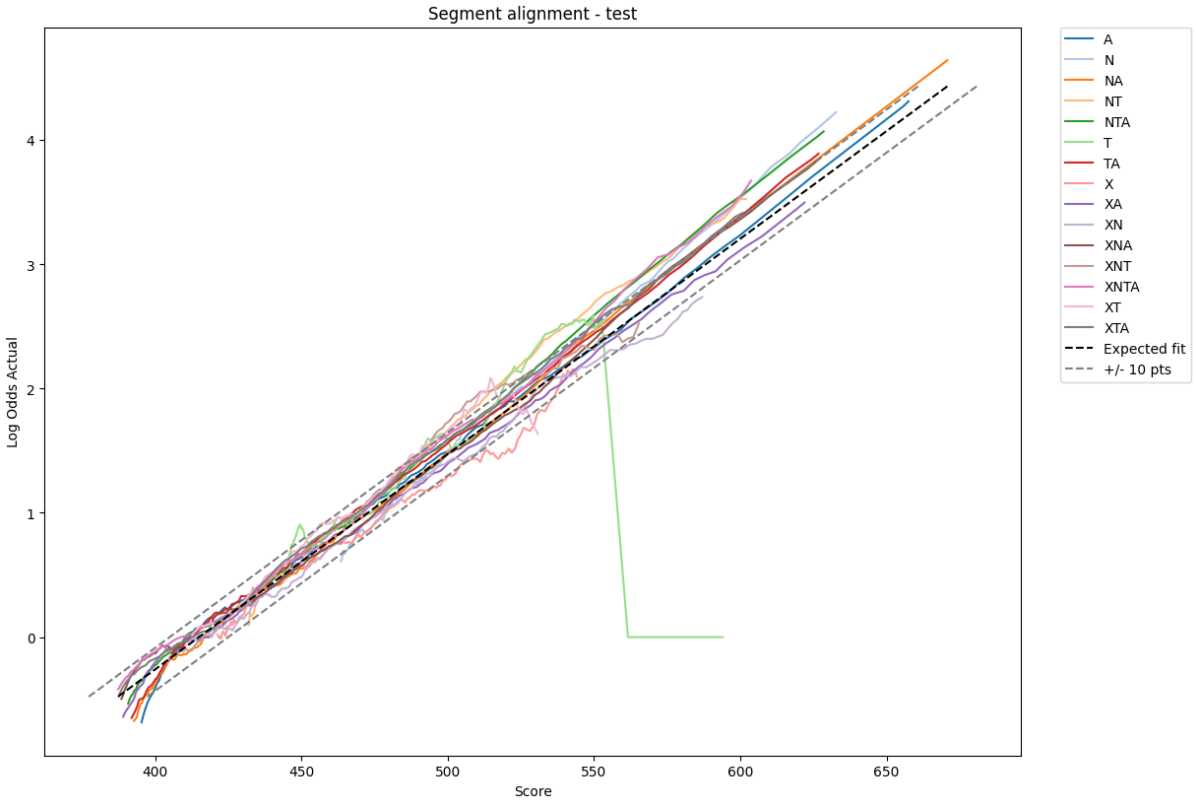

Figure 7: Alignment for 4 data source models: Test

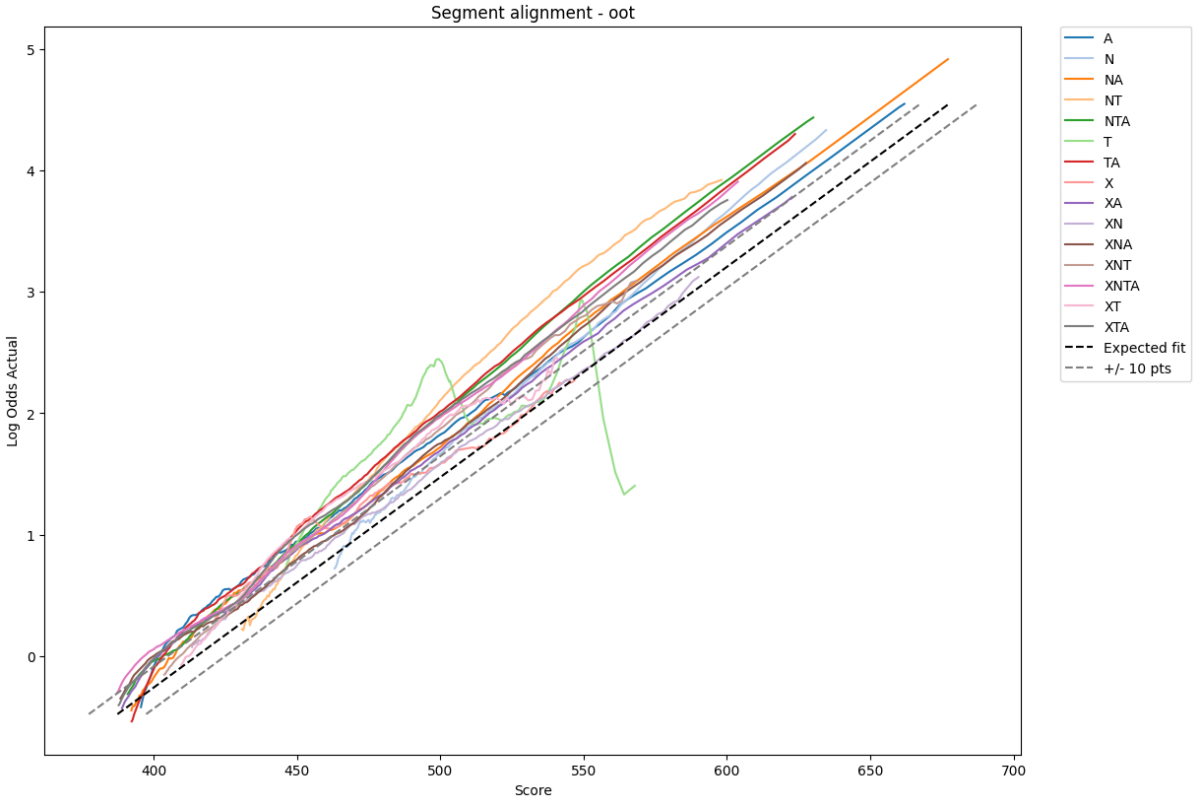

Figure 8: Alignment for 4 data source models: OOT

The plots show a strong linear relationship between score and log-odds for each combination of data sources, with the exception of the T only segment. As noted above, the production modelling project also encountered overfit for this segment. We could address this is a production model build using the hybrid model by using an out-of-time exploratory data analysis to further refine attribute selection, and/or restricting the complexity of the encoder component for data source T.

The alignment plots have the correct gradient and, in the test (in-time) sample, fall mainly within +/- 10 points of the ideal fit. This shows that the model is aligned to accurately predict good/bad rates for each combination of data sources. At most, a single intercept correction term could be used to realign each segment.

In the OOT sample there is a systematic under-prediction of good rate. This is normal, and explained by changing economic and market conditions. Nevertheless, the plots show consistent prediction of relative good/bad rates for each combination of data sources.

Segment cross-validation

We claim that our model should generalize better to unseen combinations of data sources, compared to a 'black box' model with mask. To test this claim we carried out 'segment cross-validation' for the hybrid model and a monotonically constrained neural network model with mask, using the three data sources N, T and X:

- Exclude all records for one data segment (e.g. XN) from the train sample
    - Fit the model using all train data except the excluded segment
    - Use the resulting model to make predictions for the excluded data segment, on the test sample
    - Compare predicted to actual overall good/bad rate for the excluded segment

- Repeat for each data segment
- Compute RMS error in predicted vs actual segment good rate

Table 5 shows results for three models. Figure 5 shows predicted and actual good rates per segment, for each model.

Table 5: Segment cross-validation results

Model	RMS error in segment good rate - % points
Monotonic NN	3.8
Hybrid (large z)	3.4
Hybrid (small z)	3.1

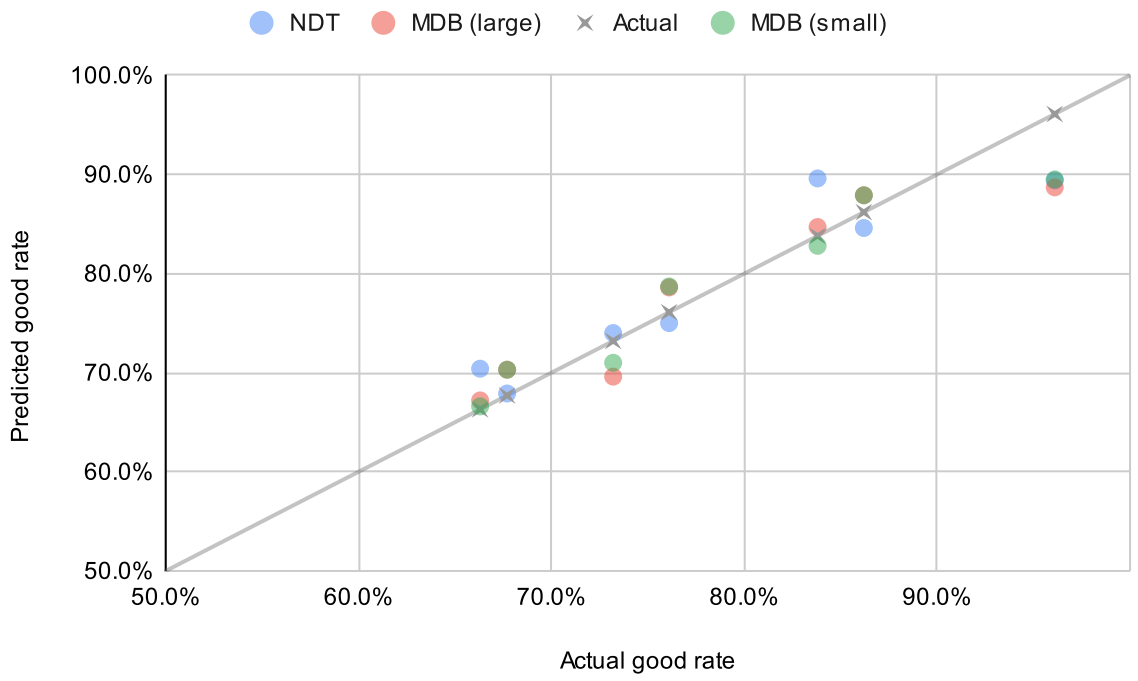

Figure 9: Segment cross-validation good rate results

In terms of segment good/bad rate, a hybrid model with latent space of dimension 20 shows ~10% smaller error on unseen segments compared to a neural network model. A hybrid model with a smaller latent space (K=10) has ~20% smaller error than the neural network model. Our interpretation is that shrinking the size of the latent space forces the model to represent information about the mask in the same dimensions that are used to predict the outcome. Too large a latent space encourages separation of these sources of information.

Adding a new data source

Having developed a model to predict an outcome variable using attributes from multiple data sources, we might hope to integrate additional data sources into the existing model without needing to rebuild the model from scratch. It would also be beneficial to re-use the estimates of the latent variable z based on the existing data sources in some form, to avoid ever-widening development datasets as we incorporate more data sources into the model.

Without expanding z

In principle, all the information contained in the existing data sources that is relevant to prediction of the outcome is reflected in the multi-dimensional latent variable z, for which the encoder calculates a conditional mean and covariance. If new data sources contain information that overlaps with the existing sources, we should be able to learn the relationship between the new independent variables and the existing coordinates of z by fitting new encoder and decoder modules as illustrated in Figure 6, while fixing the relationships between z, the existing data sources and the outcome variable y. This also involves fitting a new additive constant to represent the case that all mask coordinates are zero.

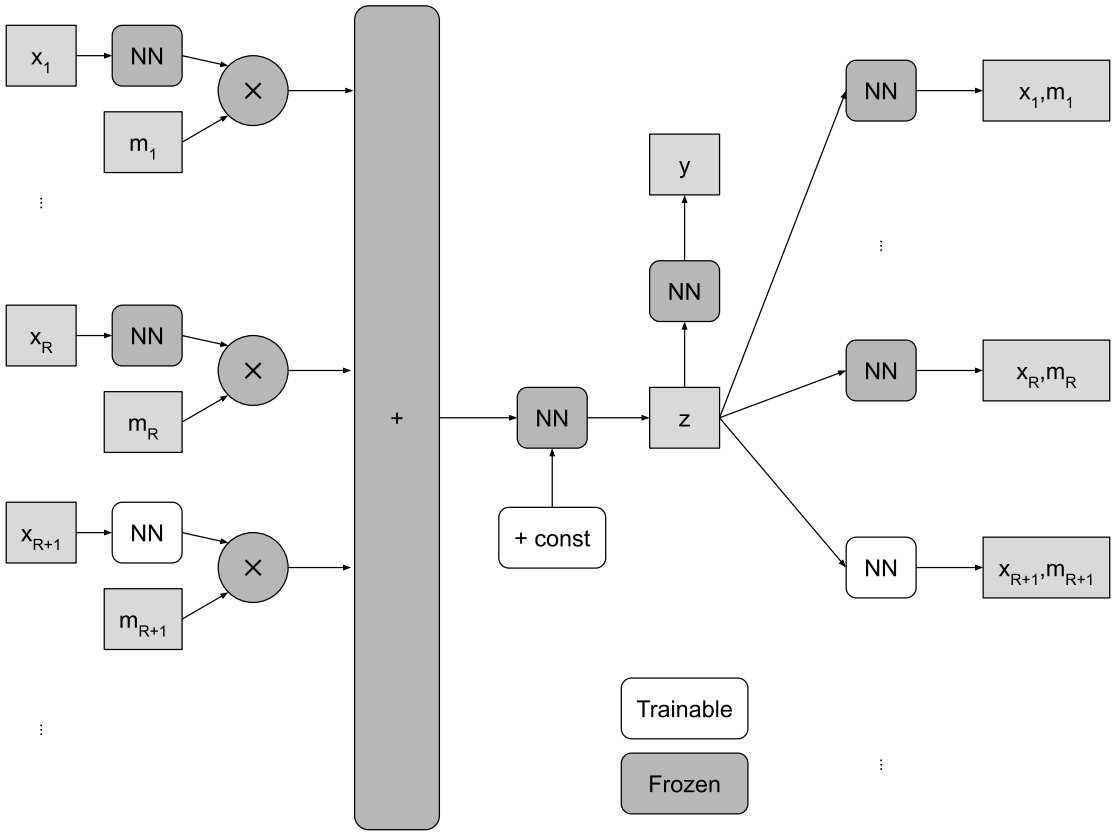

Figure 10: Adding a new data source, version 1

A benefit of freezing the encoder modules associated with existing data sources is that we may pre-compute their outputs when building a new model development dataset. That is, we may replace the vectors of independent variables $x_1, \dots, x_R$ for the existing data sources $1, \dots, R$ with the K-dimensional additive term $\sum_{r=1}^{R} m_r h_r(x_r)$ where h_r represents the frozen encoder component specific to data source r. Since we are not refitting the decoder components for existing data sources, we do not need x_r in the data. Thus, by reusing the output of the additive layer (as a proxy for estimates of z), we can indeed avoid ever-widening development datasets as we incorporate more data sources into the model. This is illustrated in Figure 7.

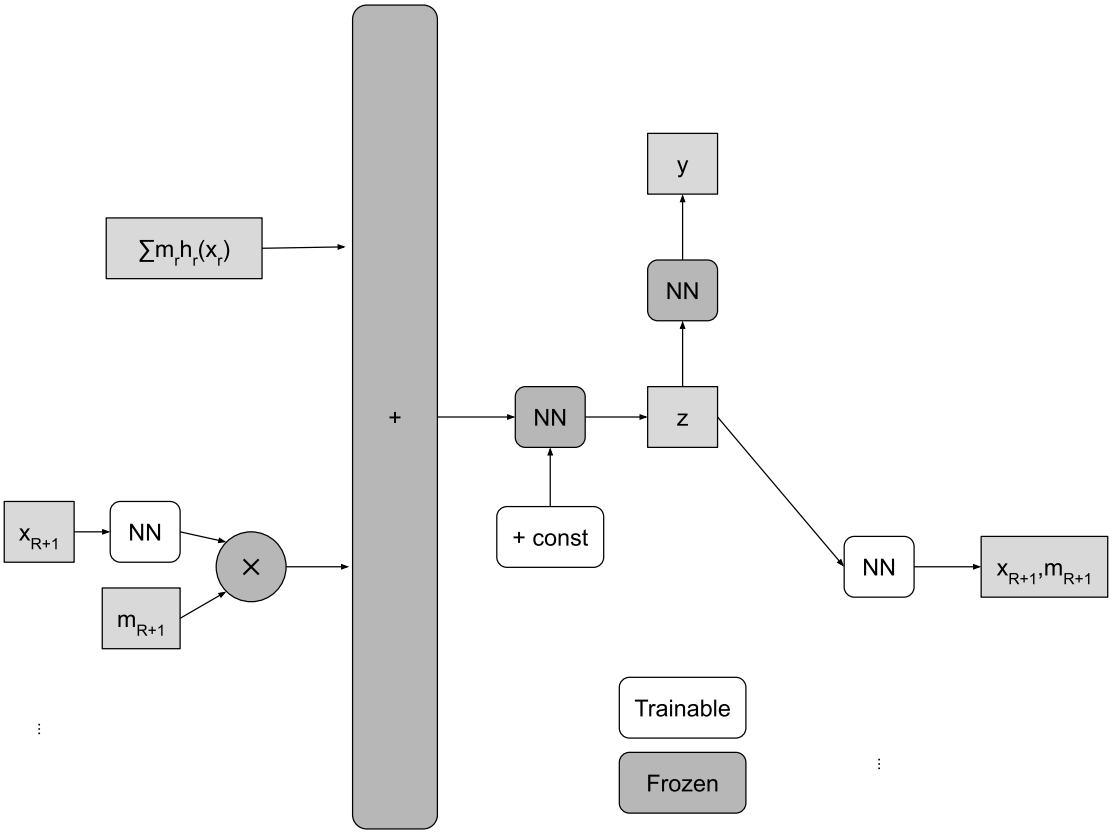

Figure 11: Adding a new data source, reusing encoder outputs

Expanding *z*

Because the relationship between *z* and the existing data sources is fixed, the approach just described may not be able to incorporate net new information from the new data sources. This is borne out by our experiments. In order to incorporate net new information, we may add new coordinates to *z* while imposing the assumption that the existing data sources contribute no information about these new coordinates. This approach is illustrated in Figure 8.

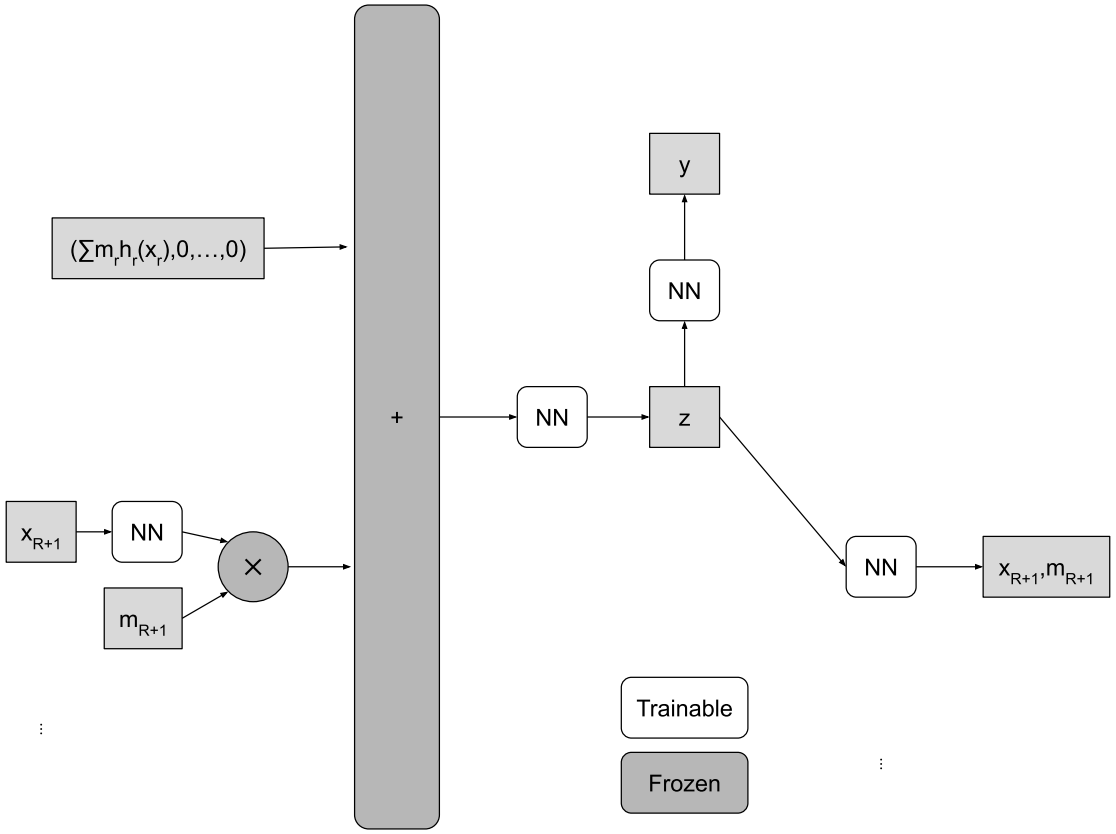

Figure 12: Adding a new data source, expanding z

Increasing the dimension of z also requires that we learn the relationship between the new coordinates of z and the outcome variable y. This requires learning a new scorer module. Rather than learn the new scorer from scratch, we may initialize it so that it replicates the behaviour of the existing scorer, with weights related to the new coordinates of z and any new nodes initialized to zero, and weights in existing nodes related to existing input coordinates initialized to their previous values. The same approach may be applied to the internal neural network that transforms the output of the additive layer to compute the conditional mean and covariance of z.

Conclusions

We have shown that a deep hybrid model using an additive encoder structure (based on a partial inference net) can be trained to predict an outcome from multiple data sources with varying availability. The predictive power of the hybrid model is competitive with existing production models segment-by-segment, with strong KS statistics and good alignment.

Compared to a monotonic neural network model without the principles additive structure, we have shown improved generalization to unseen combinations of data sources. We have also shown the potential for new data sources to be incorporated in to an existing model using this structure.

We believe the Bayesian/latent variable framework is a natural fit for multi-data modelling and data integration.

References

Collier, M., Nazabal, A., & Williams, C. (2020). VAEs in the presence of missing data. In *ICML Workshop on the Art of Learning with Missing Values (Artemiss).*

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint*.

Kuleshov, V., & Ermon, S. (2017). Deep hybrid models: Bridging discriminative and generative approaches. *Proceedings of the Conference on Uncertainty in AI (UAI)*.

Lucas, J., Tucker, G., Grosse, R., & Norouzi, M. (2019). Understanding Posterior Collapse in Generative Latent Variable Models. *ICLR Workshop DeepGenStruct*.

Ma, C., Tschiatschek, S., Palla, K., Hernandez-Lobato, J., Nowozin, S., & Zhang, C. (2019). EDDI: Efficient dynamic discovery of high-value information with partial VAE. *Proceedings of Machine Learning Research, 97*.