Intelligent Modeller

A System for Automated, Adaptive Predictive Modelling in Large Scale

by

Organon Analytics

January 23, 2011

Abstract

Intelligent Modeller is a high-level component library for doing large scale automated predictive modelling. Intelligent Modeller *models* multivariate statistical dependencies in real-life data that are presumed to be generated by an intrinsic data generating process. The creators of Intelligent Modeller designed the Intelligent Modeller based upon two emergent imperatives they have witnessed over their extensive academic and industrial expertise: 1- *Automation* imperative: Statistical model building can/should be automated in most scenarios, 2- *Scale* imperative: The demand for large scale predictive modelling has arrived, and it is here to stay. Intelligent Modeller aims to occupy this niche of statistical software where large scale, automated modelling is the rule. Intelligent Modeller is not a generic package intended to be a jack-of-all-trades software for statistical computation. It is rather a power-horse for building thousands of statistical models in an automated and intelligent fashion. This paper discusses the design considerations, modules, and algorithms of Intelligent Modeller.

1 Introduction

Predictive modelling is the practice of extracting multivariate statistical dependencies presumed to exist between a set of output variable(s), and a set of so called explanatory variables. *Predictive modelling* is in essence does not differ from all other endeavors of reductionist scientific practices: Its purpose is to simplify the complexity by using empirical data in order to explain, and possibly predict a set of important events. *Predictive modelling* is a novel term that refers to multivariate statistical analysis of empirical data, and a predictive model in the end is a mathematical structure that expresses the relationship between quantitites of interest in a specific field.

Intelligent Modeller is a system for doing predictive modelling in an automated and scalable way. The designers of Intelligent Modeller have decades of experience in building statistical models in various industries for a multitude of applications. It is a distillation and culmination of this experience, and reflects the best and the most useful practices gained over time. The contrast between **IM** and mainstream statistical packages might be phrased as follows: Metaphorically, large statistical packages are like swiss knives. They are good at accomplishing many diverse operations with acceptable performances whereas Intelligent Modeller is a laser knife that is the best thing when you want to cut steel.

As in almost every domain, the ever-present Pareto Rule is true with statistical packages as well: Most of the work is produced with a small amount of functionality, and the rest is redundant. The analysts may not even be aware of the existence of this redundant functionality. For instance, a lot of development effort goes into coding exotic black-box machine learning algorithms like neural networks, support vector machines, inductive logic programming etc. However, they can not be used in applications where the interpretability of the model is a strict requirement.

This paper discusess the design considerations, algorithms, technology, and business

model that went into the making of Intelligent Modeller.

2 Intelligent Modeller

The nature of predictive modelling as it is practiced in business is in a change.
The drivers of this change are following:

- **Scale**: The number of potential consumers are growing; the number of communication channels consumer is proliferating; the quantity of available consumer data is exponentially growing; the quality of data is increasing; the number of potential offers(products/services) is multiplying.

- **Complexity**: The definition of a *product* is changing: A news article, a video stream, a social media artifact, an advertisement, a solid durable product or service. Constraints enter the picture: capacity, budget, etc. Channel of communication needs to be customized. Pricing is another key paremeter that needs to be considered. Marketing, fraud, and risk predictables should be integrated.

- **Speed**: The response requirement of a predictive model is approaching real-time. *Scoring* consumer data in real-time is easy; however *building* models in real-time is becoming a necessity as business is becoming more dynamic and the shelf-life of a product(a news article, a tweet, a video) can be measured in hours.

Modern statistical software should respond to these changes. Intelligent Modeller is designed to meet requirements borne out of this change. The following section describes the considerations that went into this design:

2.1 Design Considerations

Automation : Automation is essential if the volume of work is high, and the amount of human resource (data analyst) to meet this demand is insufficient. Hopefully, modelling can be automated to handle large numbers of predictive models. If an algorithm can replicate the fine tuning modelling tricks a human analyst uses, then modelling can be automated. Intelligent Modeller(IM) have incorporated their years of modelling expertise into its algorithms. IM algorithms account for nonlinearities, are robust to outliers in data, handle missing values very well, account for the distribution instabilities in input variables, do auatomatic validation over population and time, etc. Though the quality of models produced by Intelligent Modeller is yet to match the quality attainable by a modelling guru, the margin is very close to justify automation. After all, that guru can not oversee hundreds of models anyway. Automation is provided in a limited scope by some software vendors, and it comes at additional price.

Scalability : Intelligent Modeller is scalable, i.e. it can produce thousands of accurate statistical models in a matter of hours. Thus, a single analyst can build many models with minimal intervention. This is in contrast to the capabilities of modern statistical software: They provide many analyst to work on a few models at a time. Creating volume work with mainstream statistical software is hard if not impossible : It requires extra coding in a proprietary language to accomplish scalability and this demands expert knowledge. Organon Analytics has recognized the need for large scale predictive modelling years ago: For example, as of 2003, Organon analysts had coded software that predicted the amount of cash consumed at ATMs/branches of a bank, a job that required building and maintaining thousands of time-series forecasting models. These kind of problems exist everywhere : Personalization Systems is another example. Organon predicts that, the need for large scale modelling will increase exponentially in time.

Adaptiveness : The usual cycle of model building is as follows : An analyst builds a

model on a static file; Model is used until its performance degrades; The analyst rebuilds the model to account for changing time. Intelligent Moeller uses adaptive modelling techniques to build model to adjust themselves with fresh data. This capability never allows models to degrade; changes in model dynamics are immediately reflected in models. It also provides automation, and the data analyst can spend time on more creative work. Most of the statistical software packages currently do not have adaptive functionality.

Accuracy/Transparency : The accuracy characteristics of individual algorithms in predictive modelling is well known: Linear modelling is easy to understand but misses nonlinearities, not robust to outliers, etc.; Classification trees are simple to build, easy to understand, but not very accurate; Neural networks are accurate, but impossible to understand, hard to build, and hard to maintain; SVMs and boosting algorithms are very accurate, but impossible to understand, not scalable to large data, do not allow incremental learning, etc. Intelligent Modellers has selected the algorithms that are best of the both worlds: they are very accurate and their structural forms are interpretable.

Performance : Building models may take long if there are many inputs and/or if there are many models to build. Algorithms are not designed to optimize the computation with respect to specifics of data in a certain application. For example, most of the values in a typical file is either missing or zero (or any other default value). This reality creates sparse data structures for analysis, and **IM** algorithms were designed to exploit this sparsity to build models faster. However, mainstream statistical packages assume generic circumstances in designing their software; they are not optimized for real-world scenarios. Meanwhile, IM employs parallelism and high-performance numerical libraries to build models faster.

Data pre-processing: A significant portion of analyst time goes into massaging data before actual model building: Normalizations, binning, grouping, factor extraction, missing value handling, outlier handling, etc. Intelligent Modeller has standardized this

process by incorporating the best practice pre-processing methods into its algorithms. Pre-processing is not an option in Intelligent Modeller algorithms; it is part of the algorithm itself. This feature makes algorithms more accurate, and robust to idiosyncracies of future data.

2.2 Modules

Intelligent Modeller has currently two main modules differing with respect to the nature of the input data and types of the algorithms being used:

1 **Data Mining Module**: This module is used for routine data mining tasks such as classification and regression where the input data is dense, and is in standard cartesian format(where each row corresponds to an entity, and each column corresponds to a characteristics of an entity). The number of characteristics is not typically *large* (where "large" in this paper corresponds to more than 1000). Intelligent Modeller might be utilized to run thousands of data mining tasks in a parallel fashion, and persists the mining results to pre-determined locations.

2 **Personalization Module**: This module is used for personalization tasks where:

- Input data is (extremely)sparse
- Input data is not in cartesian format(there might be more than one record per entity)
- Static characteristics of the entities might stay in another data file
- Entities might have a hierarchy
- Data file looks long-and-narrow, rather than short-and-wide

The algorithms currently under development pipeline are as follows:

- A PageRank-type algorithm for measuring the network value of nodes in a large graph

- A Latent Semantic Indexing algorithm to classify tagged inputs(documents, tagged multi-media)

3 Algorithms

As of early 2011, **Intelligent Modeller** employs the following machine learning and statistical algorithms to produce high performance predictions.

Generalized Additive Modelling with Regularization (R-GAM): One of the best algorithms that provide good balance between accuracy and computational performance. R-GAM is an extension of generalized linear modelling in two important directions:

1 Capturing the possible non-linear effects between inputs and output(s).

2 Regularizing the solution to prevent overfitting, thus producing robust models.

A GAM model between a statistical output Y, and a set of explanatory variables $\{X_1, X_2, \ldots, X_n\}$ can be expressed as follows:

$$\hat{f}(X) \equiv E(Y|X_1, X_2, \ldots, X_n) = f_0 + f_1(X_1) + \ldots + f_n(X_n) \equiv f_0 + \sum_i f_i(X_i) \qquad (1)$$

where each $f_i(X_i)$ is a non-linear smooth function of its argument. Regularization of the "solution" $\hat{f}(X)$ is obtained by minimizing the following "loss function":

$$L(Y, \hat{f}(X)) \equiv E(Y - \hat{f}(X))^2 + \lambda ||\hat{f}(X)|| \qquad (2)$$

where $||.||$ represents an appropriate norm on the function $\hat{f}(X)$, and the second term above corresponds to a penalty on the complexity of solution (λ adjust the smoothness

of final solution, and is determined by the data). Intelligent Modeller employs L_2 norm on the regularized solution $\hat{f}(X)$ due to its lower computational cost.
R-GAM modelling have following properties that make it one of the best competitive modelling techniques for building a recommendation engine:

1 R-GAM captures non-linear effects between a possible recommendation and its drivers

2 The number of inputs in a recommendation scenario might be large (most often, the number of parameters to be estimated is much higher than the number of available transactions). Hence, regularization of the solution is essential, and R-GAM handles this regularization with ease.

3 Incorporation of *any* type of input data does not present difficulty. Thus, modularity of recommendation engine is automatic.

4 Memory consumption of the computation is modest in model building process: Input data is converted into co-ocurrence tables, and due to sparsity of user-item dyadic matrix, the size of the co-occurrence table is very modest compared to the size of data. R-GAM directly works on co-occurrence tables -which easily fits into the memory- to build the models.

5 R-GAM computation allows *online learning*, hence can be made adaptive. Rebuilding the models do **not** necessitate a complete pass over the whole data; only the additional batch needs to be processed.

6 R-GAM modelling is very competitive in terms of model accuracy. Organon's experience in using R-GAM modelling to build recommendation engines, and statistical models in other domains (R-GAM experts have built thousands of statistical models that are used in various industries for different purposes) taught that R-GAM models are one of the best off-the-shelf statistical learning models.

7 Model formulas produced out of R-GAM models have low memory requirements which make them ideal for generating real-time recommendations.

8 R-GAM might be used for both classification tasks and regression tasks.

Intelligent Modeller has a history of extensive experience with R-GAM modelling, and will continue to use it as one of the work-horses in building predictive statistical models.

Sparse Generalized Additive Modelling with Regularization (Sparse R-GAM): Sparse R-GAM is a propritary extension of R-GAM algorithm developed by Organon Analytics. **Sparse R-GAM** is exceptionally good in scenarios where the percentage of missing values in the data is high. Traditional learning algorithms assume that data is dense, full, and values are reliable. The business reality is on the contrary: data has a high percentage of missing values, and values contain errors. **Sparse R-GAM** uses a local loss-function to find the best approximating solution rather than a global loss function. The result is not a single global equation but rather a family of equations changing with respect to missing value structure of the data sample. The trade-offs are **Sparse R-GAM** takes longer to train, and the model size is large compared to other models.

Classification and Regression Trees (CRT): CRT models are simple and intuitive models in understanding the statistical dependencies between a set of explanatory variables, and an output variable. Though being relatively inferior in accuracy, they are very good in variable filtering, and understanding the "model". Intelligent Modeller has a CRT implementation that can be used in an ensemble modelling framework with other models.

Boosting Algorithms: Boosting algorithms are meta algorithms that use other machine learning or statistical learning algorithms to build a better model than the individual models produced by each algorithm. Boosting has emerged as a powerful learning

methodology along with support vector machines (SVM) in the last decade. SVMs are very costly to be used as scalable recommendation algorithms, and thus currently is out of option. Boosting can be code computationally efficient if the underlying learner is kept simple. Boosting algorithms are very accurate, but their memory footprint is large. Depending on the problem dimension (item size is the critical factor) models produced by boosting algorithms can be used to generate recommendations. Intelligent Modeller has a boosting implementation based on AdaBoost algorithm.

Time Dependent Regularized Matrix Factorization: The second work-horse of the family of Intelligent Modeller personalization algorithms. In a nut-shell, the personalization score $s_{ui}(t)$ for the item i by the user u at time t can be formulated with matrix factorization as follows:

$$s_{ui}(t) \equiv f(t) + f_u(t) + f_i(t) + p_u(t) \cdot q_i(t) \tag{3}$$

$f(t)$ models the global temporal behavior of the score, and is independent of users and items. $f_u(t)$ corresponds to temporal evolution of user preferences, and also captures the effects of user profiles. $f_i(t)$ corresponds to temporal evolution of item behaviors, and also captures the effects of item profiles, seasonalities, etc. The last term $p_u(t) \cdot q_i(t)$ is where the matrix factorization happens: $p_u(t)$ vector captures the **latent** user **factors** that codes the user proclivities towards items, e.g. an entry in this vector might show the high tendency of user to buy "luxury" items. $q_i(t)$ vector captures the latent items factors that codes the hidden item clusters, e.g. an entry in this vector might indicate that item *is* a luxury item.

Time Dependent Regularized Matrix Factorization modelling has following properties

1 They are very competitive models in terms of accuracy.

2 Temporal dependency can be captured by allowing user factors and item factors

to be dependent on time. Low frequency (entire behavior) and high frequency (session-based) dependencies can be captured by incorporating factors at various time-scales.

3 Allows efficient compression of user-item-time triadic tensor which is extremely sparse.

4 Memory consumption at model building phase is high if temporal dependency is strong.

5 Training of the model is more demanding than other models used by Intelligent Modeller (in terms of learning algorithms, and CPU cost).

6 Number of parameters might exceed the number of transactions easily. **IM** employs regularization for all parameters in above equation to prevent overfitting.

7 Factors can be updated adaptively. However, cost is higher than R-GAM modelling.

Matrix Factorization models when used in conjunction with R-GAM models in an ensemble-modelling framework produces excellent personalization models in terms of accuracy. They extract the information from input data with different specifications, hence combining them produces a superior ensemble model.

Item-to-item collaborative filtering: Probably the simplest and the most widely used personalization algorithm. Item-to-item CF algorithms build similarities between recommendables, and weights these similarities according to past user preferences to generate new recommendations. Though being widely used due to its low computational cost, it suffers from several weaknesses: 1) User-based features (demographics, derived transactional variables), and item-based features can not be incorporated, 2) Cold-start recommendations for new users can not be generated, 3) Temporal dependency can not be modelled, 4) Models lack dynamism: Real-time adjustment of recommendations is

hard, 5) Variable selection is not possible, 6) Suffers from multicollinearity which results in favoring items with high-support. Despite its disadvantages, this is a viable algorithm when data is plenty, items have long shelf-time, cold-start is not an issue, and performance has the highest priority. bf IM has an extremely fast item-to-item CF algorithm.

Latent Semantic Indexing (LSI): If the shelf-life of an item is low (news, articles, etc.) or the item is new, personalization engine might not properly *personalize* these items. If a hierarchical taxonomy of items is available, this hierarchy can be used to personalize these items. For example, if once a new multi-media content about a soccer team becomes available, it might be personalized to interested customers by using the higher hierarchical concept of a "soccer team". However, there are situations when an efficient taxonomy of items is not available, and an automatic approach to classification of items is necessary. Latent Semantic Indexing uses tokens (tags) attached to an unstructured item to produce a taxonomy. For example, a news item can be parsed to classify it as a "news item about soccer team X". Latent Semantix Indexing is a matrix factorization technique that is very powerful in classifying unstructured data into homogeneous classes. LSI should be made adaptive and parallel to meet high computational demands. A parallel LSI algorithm is currently being developed to be integrated to **IM** algorithm family.

Ensemble Learning: Ensemble learning combines many models to produce a best model whose accuracy surpasses the accuracy of its individual constituents. Each learning algorithm puts forward a different specification (a "formula") to learn the regularities buried in data. Ensemble learning, in a way, optimally combines the "opinions" of each model about a case (e.g., best personalization set to a customer) into an "overall opinion" that is collectively better than any individual opinion.