# Relational Probability Memory

## Introduction

Machine learning has made large advances over the last decade, powered primarily by Neural networks (NN). However, the advancement is becoming more and more incremental and current AI is nowhere near intelligent. Prior to the rise of NNs much of AI was focused on the use of statistical and symbolic methods, such as expert systems and fuzzy logic. More than 50 years of work in these areas lead to the initial breakthroughs in AI, but with the rise of NNs little has been done to leverage this knowledge or find a means to mix it with the opaque nature of NNs.

Methods attempting to integrate NNs and symbolic AI, such as neural-symbolic AI, have recently received going interest. The hope that greater progress can be made toward true intelligence by leveraging neural, statistical, and symbolic approaches to AI seems rational given that all methods have been proven leading to the current state of AI. Unfortunately, these technologies do not pair well together. NNs and symbolic AI don’t interface well together primarily due to the opaque nature of NNs which operate as black boxes with only inputs and outputs used to manage their behavior. Additionally, NNs are very resource intensive, limiting the environments in which they can be used. Alternately symbolic AI operates primarily via rules and logic to make decisions, these rules often require a lot of human input and intervention to get correct.

Relational Probability Memory (RPM) is a novel form of statistical machine leaning that use vector addressed memory to store information relationships. RPMs have been shown to produce highly accurate results comparable to NNs with small models and low computational complexity. RPMs in the place of NNs allow for must existing methods in neural machine learning to be more easily mixed with symbolic AI, due the completely logical and open nature of RPMs and vector addressed memories. As an alternative to NNs, RPMs bring simplicity and ease of integration without compromising on results.

## Overview

RPMs work by framing information into relationship vectors then mapping them into a vector addressable memory. Recalling information from the memory is probabilistic based on the information in the request frame and its corresponding vector set. Recall results are the set of values with probabilities that matching the vectors in the recall frame, the results have the most probable value first. Results will also include all other matched values and their probabilities sorted by probability.

**Processing Information**

RPMs are trained using datasets that have been tagged or can be organized with some relationship between elements of the dataset for unsupervised training. Data from input datasets is mapped into fixed sized frames in sequence, rolling through the dataset frame by frame. Framing can be simple or complex dependent on the features or combination of features being trained. Information in frames can come from transforms or context or be the results of prior or future iterations. This process is used for training, testing and for predictions, framing for all A screenshot of a computer

Description automatically generated with low confidenceinteractions with the RPM are identical.

**Training Models**

For each frame the vector definitions in the model, referred to as NumberSets are generated. The node, called an Accumulator, that associated with each vector for this data is retrieved and updated with the value or the count for the value is incremented.

A screenshot of a computer

Description automatically generated with medium confidence

**Recall and Prediction**

Recall and predictions or inferred values are determined by the full or partial information available in the model for vectors in the request. Each vector generated from the request frame is looked up in the memory, if present the vectors values are modified by the ‘calculator’ function then added to the value sums. The value with the highest probability wins, though all values and probabilities will be returned in order of probability. Ties happen often and are resolved with the baseline probabilities of values for the model.

Diagram

Description automatically generated with low confidence

NOTE: probability is not a strict term here, ideally values should be 0 – 1, but the rule is the highest value wins.

## Training

As describe in the last section, the training process encodes data into frames that are mapped to vectors for addressing memory, the trained values are then counted in Accumulators. Each of the NumberSets records information independently from the others, allowing for training to be done in series or in parallel and achieve the same result. This allow a large dataset to be trained across a large number of machines and compiled together after the fact. Training records all information that is passed into the model, producing a model that is many times larger than the original data. This can take a lot of memory if large windows are trained in single passes, but processing is very minimal.

Once trained, a model should be tuned to reduce its size and remove all the information that is not useful in making accurate predictions. This is covered later.

### NumberSet to Vector

The term NumberSet relates to the frame elements of a vector definition and is derived from the initial powerset for a frame’s positions. The powerset represents all possible relationships and each subset represents a specific dependent relationship that is a NumberSet or vector definition.

A NumberSet contains the positions in the frame, the ‘-‘ and ‘x’ notation is commonly used in the software to see the relationship represented by a vector.

- - X - -

x x X x x

- x X x -

0

1

2

3

4

2

1

2

3

There are 2 specially named NumberSets that are referred to by name and used for specific tasks:

* **Identity NumberSet** – the number set with 1 member which is the token at the frame focus, when working with a model that has a focus. If not present, then the token is unknown.
* **Full NumberSet** – the single number set with window size members, it represents the most specific relationship and when present in a prediction all other NumberSets will have values as well.

Additionally, there are two to subsets:

* **Context NumberSets** – the set of NumberSets representing relationships without identity these do not include the token at the frame focus.
* **Relate NumberSets** – the set of NumberSets representing relationships with identity, the opposite of the context NumberSets

## Prediction and Visibility

Predictions are generated from input data one frame at a time, just as in training the values are mapped to vectors but in this case the Accumulators containing value probabilities are retrieved. The resulting set are summed to get a result, often with a weight or filter in the process.

### Visibility and Traceability

A benefit of RPMs is the information used to make predictions is available for decision traceability in the vector memory. Additionally, it makes debugging a model straightforward as the source of all information for each decision can be viewed and the reason for a match or mismatch can be determined. From this next steps can be decided be it changes in the model, additional training, design or tuning. The information available at decision is the list of Accumulators, 1 for each NumberSet. The set of value probabilities (value, probability, count) for each of them and the final decision set of value probabilities ordered from best to worst. If the vector set recording was enabled during training, each Accumulator can be identified by the exact trained input and linked via frequency to all options for the prior and later vectors (this can be used for content generation, or re-memory).

In addition to access to decision information, the model itself is open for inspection with all vectors, frequencies and probabilities clear for inspection or modification. Tunning processes utilize this to validate and modify the model in different ways to achieve better results or to reduce the size of a model.

### Prediction

Predictions generate a list or results ordered by probability, additionally a *Prediction Type* is provided indicating the amount and type of information used to make the prediction, these prediction types allow callers to route, tune, qualify, or trace the results of predictions to better achieve the desired results.

Prediction types:

* **Recall –** A match on the fullest or complete number set and having one and only one known outcome. This is the most common match when testing data that was used for training or learning.
* **Recall-Predict –** A match on the fullest or complete number set and having two or more possible outcomes. The outcome must be predicted from the set of options using the other information available. This is the case when the same sequence leads to different results.
* **Predict-Relate –** A match of at least one number set that includes the identity and another element.
* **Predict –** A match of at least the *identity number set*. The outcome must be predicted from the available information in the number sets and can be limited to the known outcomes for the token.
* **Predict-Unknown –** Same as predict, but with an unknown token (no *identity number set*). The outcome is determined by context number sets only, reducing the accuracy. This should be a very small or non-existent prediction type for a well-trained Model.
* **Default –** No information is available for decisioning for an input, the most probable outcome for the model is the best prediction available. This decision type should only be seen if the Model’s training is incomplete.

## Tuning Models

Tuning models produces significantly more accurate results model can be tuned for general accuracy or tuned for accuracy with respect to a specific goal. Generally, for RPMs tunning is about reducing noise, as the window size grows the information in the model grows exponentially, both relevant information and noise. There are two general methods of tuning a model 1) *probability weighting* and 2) *value reduction.* Reduction is the process of removing redundant or less valuable values or vectors to reduce memory usage and inaccuracy. For some weight calculations there will be improvement, for others degradation or no impact.

For both weighting and reduction there are two methodologies: *Logical Methods* and *Statistical Methods*. These are described in more detail below, additionally mixed methods or mixing methods serially often produce better results.

### Probability Weighting

Weighting of individual vector value probabilities or changing the NumberSet weights can be used to tune the results. There are a set of options for basic NumberSet weighting based on the size, distance or just all the same. Since a NumberSet represents a specific relationship set modifying the weight correctly for a dataset allows for better results. Individual value weights are generally modified relevant to any algorithm tuning the model as discussed later.

The default weight calculation is:

*Weight = Accumulator Value Probability \* NumberSet Weight*

### Value Reduction

Removing value probabilities from accumulators or removing full NumberSets is an alternate to weighting the values and NumberSets. Value reduction has an added benefit of making the model smaller, a 90% - 95% reduction is common. In practice value reduction has been shown to produce more accurate models than weighting with the same algorithms, and it is much easier to identify when to stop as the reductions at some point no longer add value.

### Logical Methods

Logical methods utilize relationships between values, vectors, data or the NumberSets themselves. Most of these reductions remove far less data from the model than the statistical version and are more able to address information dependencies by leveraging known relationships.

#### Redundant sets

Redundant set reduction was created (in conjunction with entanglement) to make exported rules match more closely to abstract rules expected by humans. Each NumberSet is reviewed against its direct subset NumberSets, when a probability of a ‘parent’ and ‘child’ match the parent is removed, thus retaining the shortest vector (sequence) that produces the probability. When redundant vectors are removed in some models the inheriting vector (Accumulator) will need to have its probability increased to maintain model parity.

#### Data Definition Reduction

Data definitions when provided allow subsets to be known as closed or open sets, this provides with certainty several rules that allow vectors that cannot add value to be removed. For instance, some values are static and may only have a single token or small set. Others may be a closed set, so any vectors that would match on an unknown token to that value would never be relevant. This evaluation allows for a small increase in accuracy, but potentially a very large decrease in the size of the model and resources it needs.

When a data definition is not available, an evaluation of the data allows for a definition to be created from the model. This definition can produce almost the same result as a provided definition.

#### Dependency based Reduction

Assessing each decision with the tuning and or training set allows dependencies to be mapped. These maps can then be used to identify vector/value pairs that pass decisions are dependent on as well as pairs that prevent pass decisions, and those pairs that simply never matter in decisioning. By reducing the vector/value pairs that increase error rates the accuracy of the model is increased significantly and very quickly. Longer dependency chains can also be mapped and used to reduce vector/value pair chains to improve results.

This method can be used after other tuning methods to improve the results further, using it before other methods may reduce the effectiveness of the other methods. This method like many others can tune too closely to the tuning set, providing very accurate results on that set but limited or negative results on other sets.

### Statistical Methods

Most Statistical methods are based on regression to improve the accuracy or an accuracy score, but they may also be based on simple frequency such as the Naïve reduction. Tuning with these methods produces good results and does not require any logical understanding of information dependencies but are also fundamentally limited in there results by that fact.

#### Naïve Reduction

Naïve reduction is simply removing the most naïve values or vectors from a model, it is fast, significantly reduces a model’s size and generally slightly improves result accuracy. For Naïve reduction all Accumulators and values are reviewed, any that fall below a set threshold for count are removed. Limiting the NumberSets this is applied to can improve results for some models, such as when there are context and non-context NumberSets.

#### Iterative Carving

This method iteratively ranks value per NumberSet fscores, then for each in sequence tests a set of reductions limited by the count (frequency). The largest reduction that results in a positive or neutral movement toward the goal is performed, then it continues. This takes many iterations but produces very consistent results in increased accuracy. As the reductions are each dependent on others, the order of the reductions is very significant.

This process could be significantly improved with dependency information such as would be derived in a logical method, at this time know reasonable method has been identified. In addition, this process can be very slow.

#### NumberSet Reduction

The most significant reduction is the removal of a complete NumberSet and all its vectors. In larger windows many of the NumberSets contain mostly noise or redundant information and the model can only achieve greatest accuracy by removing full NumberSets. Most of the tuning methods will result in some NumberSets being removed. NumberSet only reduction operates in 2 different ways:

* *Dependency independent* method checks for any NumberSet that on removal improves the model’s accuracy, and removes it, then iterates. This is relatively fast and requires few passes through the NumberSets.
* *Dependency-based* method groups all NumberSets together and checks the accuracy of the model with the set removed. This process is very slow as worst case is exponential for the number of NumberSets. In practice the total number of sets to check is much smaller though still very large.

### Tunning Models to a Goal

Model can be tuned in different ways to achieve different results, tunning to optimize for a specific Prediction Type for instance can allow a model to have higher accuracy on unknown inputs or on known inputs, allowing the models to be used in conjunction for an overall improvement. Goals can be extended to anything that is countable on evaluation of a model such as on specific outcome or a give value. In model mixing we will explain how goal tuned models are used to create more accurate mixed models.

### Result Amplification

Result amplification allows for a model to reduce the set of expected outputs and amplify them based on the resulting reduction in probability each vector. It produces the same result as would be produced by a model that is only trained on the subset of results, probabilities for that subset are increased. Results from amplification are not insignificant and should be used when possible, to produce the best outcome. Some models do not work well with amplification specifically; poor or even negative results will come from the dataset used to tune the model.

## Mixing Models

To achieve desired results, it is often necessary to mix multiple models together that may contain different information, have different tunning, or a different view of the same information. Since RPMs retain probabilities for all information and decision it is a straightforward process to mix the information as well as optimize the mixes. With RPMs the result sets just need to be merged to get a combined outcome, though the merges must be balanced.

### Merging Model Results

Models are mixed by merging their respective result sets, the simplest form is to just mix directly by summing and averaging each value between the models. This often is not going to produce effective results as each model has different probability ranges, weighting, etc. Thus, the models need to be brought to parity in the merge to get effective results.

* **Flat** – Single value applied to adjust the probability in the merge
* **Prediction Type** – Value per Prediction Type to adjust probability in the merge, Prediction Type of the base model is used
* **Prediction Type x2** – Value per Prediction Type of base model to each Prediction Type of the merged in model to adjust probabilities
* **Value Weighted** – Individual weights for each value being merged in these are added after any of the other balancing has been applied

In practice, the more detailed balancing produces better results on trained and tuned data, but often will go askew on a given test data set. More work needs to be done to affectively model the disparity between models in individual value prediction strength. It is clear from a logical view that some models are more accurate at predating a subset of values and less accurate at others.

### Tuning Model Merges

Merge value tuning successfully produces good results in both identifying if two models should be merged to improve accuracy and establishing the best values for merges of each type (aside from value weighted, which is a work in progress). The values are simply set and iterated over a range then bisected individually until local maxims are located. This process can be done very quickly.

## Learning Curve

RPMs will provide a prediction once any information is added to them, as more information is added the results prediction will improve. Reasonable results can be achieved with very little training data, quality and generality of the relationships is far more important that quantity. Thus, it is possible to train a model to accurately make a relatively simple prediction with just a few datapoints.

### Model Completeness

Models can be tested to determine their completeness by common evaluation that is independent of the model or its purpose.

* Check the number of default tokens with the tuning this should be 0.
* Check the percentage of unknown tokens with the tuning set if greater a few percent the model could use additional training.
* Run the data definition evaluation to determine if there are values that are not represented in the tuning set then the set is missing required information
* Check to see how many values are associated via naïve data (data with a single instance), if more than a few percent of decisions are based on naïve data the model could use additional sources if the model is complex.
* Datasets may also be marked as complete, this is the case for limited sets such as those that grain AND, OR, XOR gates

## RPM Complexity

The linear structure of PRM models require only additive arithmetic for training and prediction, this significantly reduce computational load. The complexity of a Model is directly associated with the window size. The size of the model goes up exponentially with the window size thus large windows require significantly more memory resources, however in all cases training and prediction are O(n). Parallel training and logical reduction are two methods that have been found to reduce resource usage, allowing larger models to be trained and used with less memory consumption.

### Computational Complexity

RPMs are linear models that use only simple arithmetics for all operations. This aspect of RPMs makes them very simple computationally allowing them to operate in environments that are generally not available to machine learning or AI.

### Model Size

After training models contain all the information that was available, the tuning process allows for most of this information to be discarded leaving a much smaller and more accurate model. In many cases though the model can be reduced significantly more by goal-oriented tuning, allowing the minimal set of desired outcomes to be the only focus of the model. This allows for a larger model ~500mb to be reduced to ~250kb while gaining accuracy and becoming faster.

These micro models combined with the simplicity of evaluation allow them to run on many devices and environments they typically couldn’t be used, such as on a webpage in JavaScript or on an embedded device.

## Interfacing and Integrating

RPMs can be interfaced with via APIs, rules, or data. This provides many options for integration with other methods or systems.

### APIs Interfacing

Integration via API allows direct access to the model and its functionality. Direct data access allows checking existence, probabilities, and comparisons as well as the ability to modify the model. APIs for Java allow RPMs to be directly integrated into any java application or workflow. REST APIs and command line allow for integration with applications in any language.

### Symbolic Rules Interfacing

Symbolic AI has been around for almost a century, generally it represents logic as people do allowing for evaluation and understanding. RPMs represent information such that it easily interacts with in abstract symbolic logic or rules. This provides a path for integration between Symbolic AI and neural AI, although RPMs are technically not neural, they serve the same role.

#### Exporting Symbolic Rules

RPMs can export the model in the form of symbolic rules, much like other rule based and expert systems use. The format can be modified but often the ruleset is very large, so reductions are recommended to get the set of exported rules to include only elements that are relevant. Each rule is a set of elements to match or not match and a probability for a value. Combining all matched rules results in a proper prediction +/-.

#### Training via Symbolic Rules

Symbolic rules in the probabilistic form or in other forms can be used to train an RPM, the resulting model may have less information on frequency, but will have the ability to make predictions. Additionally, a model can be trained with a combination of data and rules, creating a method for integrating what it learns with what another system knows in order to get the greatest accuracy.

### Data Interfacing

Interfacing with data via the vector addressed memory allows a direct means to get or modify values, probabilities and more. This access is used for many tuning algorithms and provides the fastest method to modify RPM models.

## Long-term Memory & Recollection

RPMs are a relational memory, as such they provide very accurate recall of information trained into them. The boundaries of information or windows are adjoined via probability and/or frequency allowing long streams of data to be wholly reconstructed, albeit with sometimes forking paths based on probability. This feature of RPMs allows for content generation based on the information that it has learned producing goal-oriented content based on seeding (nothing is a seed as well) and probability.

## Additional Algorithms

### Solid Models

Solid models have data compressed to the minimal needed to make predictions. Making a solid model significantly reduces the memory it uses and optimizes it for much faster predictions. Solid models cannot have accumulators or values modified; thus, they can’t do additional cycles of training and tuning. It is best to convert a model to solid when performance is need, or memory becomes an issue. The original model can always be saved or retained by a different name in the same container.

### Entanglement

Entanglement is a special case, as it does not reduce out any accumulators, it instead folds like accumulators together to creating an entangled accumulator that is the same, but different depending on how you view it. Generally, 85% – 99% of accumulators will be entangled creating a large memory and performance benefit. Entanglement produces much better symbolic rules as like slot elements with the same result are folded into the same rule possibly creating an ‘\*’ rule and allowing simplification to negative rules.

### Data Generation or Information Recall

Data can be generated or ‘recalled’ with a level of probability in the mix once it has been trained. For larger window sizes the results are very close to exact recall but must start with a seed to identify a start. Outcome specific content generation or recall provide much better results. This generation is simply finding the most probable next, last, or missing token oriented to a specific outcome, set of outcomes, or any outcome. There is no additional work for this to be present if save VectSets is left on during the model during training.

## Adaptive Training

Adaptive training is a method by which a model is trained with a single new information relationship at a time combining training, tuning and multi-model merging into a single process that reduces memory use and attempts to always maintain the most optimal model as new information is integrated into the model.

### Iterative process

The new information is determined externally or by extending the current frame, all relevant relationships are then determined, trained, and evaluated in multiple steps. Tuning and evaluation methods can be mixed and are pluggable, due to the incremental nature it is good to determine the most effective method early. Most tuning and configuration options can be left to the algorithm to determine and set in the model early in the evaluation process. The result is an easy to use, low resource training method that delivers optimal results.