Integration of Similarity Factor in Various Predictive Methods for Low Level Datasets

Manav

1. INTRODUCTION

Bayesian models or models of similar tendency to rely on historical data to calculate posterior probabilities often face problems when faced with unique current situations. Due to their heavy reliance on previous data, any new condition/event would result in inaccurate predictions with a lack of previous data. The similarity factor aims to calculate the overall similarity of our current event/ sub-event with past event/past-sub-event.

2. AIM OF PAPER

This paper focuses on designing a similarity factor tailored for low-level datasets to enhance the overall accuracy of predictions within Bayesian hierarchy models. It aims to optimize predictive algorithms based on historical evidence, eliminating the need to independently analyze each parameter or event in isolation. While Bayesian models serve as the primary example, the discussion extends to other predictive models where individual parameter weights significantly influence outcomes.

The objective is to present a concept that is both: accessible and applicable to individuals who may be new to the domain of predictive modeling but possess a basic understanding of its principles. This approach is particularly valuable for those seeking a comprehensive yet intuitive introduction to Bayesian prediction models and similar architectures, all within the context of optimization and computational efficiency.

The proposed method can be integrated as an intermediary step or as an optimization tool for existing predictive models that struggle to incorporate historical data: a challenge that Bayesian models inherently address. However, this paper does not provide any code or programming solutions for the methodologies discussed. Its sole purpose is to engage an audience new to data science, offering a straightforward path to optimization without requiring complex algorithms or extensive computational resources.

3. INSIGHT TO BAYESIAN PREDICTIVE MODELS

Before designing a *similarity factor*, let's take an insight on *Bayesian models*.

3.1 Bayesian models use the Bayes Theorem in order to calculate the probability of a certain event given some evidence. The Bayes Theorem is as follows:

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}$$

where:

*P(H—E) is the posterior probability: This shows the probability that the hypothesis is true with respect to the given evidence. This is what is to be calculated.

* P(E—H) is the likelihood: the probability of observing the evidence assuming that the hypothesis is true.

*P(H) is the prior probability: the initial probability of the hypothesis before considering the new evidence. This is often derived from historical data or previous knowledge.

*P(E) is the marginal probability of the evidence: the overall probability of observing , across all possible hypotheses. This term normalizes the equation to ensure that the probabilities add up to 1.

How Bayes Theorem Works

Bayes' theorem updates our initial belief about a hypothesis by factoring in the likelihood of the new evidence under that hypothesis. This results in a revised probability that better reflects the likelihood of the hypothesis given both prior knowledge and current observations.

3.2 Bayesian prediction models use the principles of Bayes Inference. This allows the models to continuously refine their predictions as more information becomes available.

Bayesian models generally update the posterior probability continuously with new data. This continuous change of posterior probability attains some degree of stability after no additional data that are relevant to increase or decrease in the posterior probability can be subjected to implementation in the framework. This is the core strength of the Bayesian predictive model. However, this means that posterior probability for a new data would be heavily based on historical data. This sole dependence on previous data can skew the prediction made for our current event.

4. PROBLEMS FACED WITH UPDATE OF HIGH DIMEN- SIONAL DATA

4.1 *Bayesian Models* calculate the individual ”prior” taking into account all the parameters. For illustration we shall look at an event of a lower dimensional degree; namely, Event X. Event X is described as thus:

$X = [x_1, x_2 ..., x_n]$

where X is an event classified by parameters ”$x_1, x_2 ..., x_n$” ;and each element of the set X denotes a unique parameter.

Assume that we are to hypnotize the probability of an event ”A” happening given our event X The model would likewise calculate prior for each element or group them in a joint prior as follows:

1. Prior probability:

The prior probability would be based on your current event and past belief of event A happening. This could be depicted as follows.

$P(x_1, .., x_n)$

2. Calculation of Marginal Likelihood:

This is the total probability of observing and integrating all possible events or hypotheses which could be defined as integrating all possible events.

$$P(X) = \sum_A P(X|A) \cdot P(A)$$

$$P(X) = \int P(X|A) \cdot P(A) dA$$

Integrating over a large path, where parameters are represented as vectors in higher-dimensional spaces, can be highly demanding in terms of computational resources. Furthermore, the final probability estimates may be skewed due to the absence of an explicit ”bias” in the model. As a result, the model’s output can deteriorate, particularly when limited computational time is available or when the introduced bias fails to optimize the current event, instead focusing disproportionately on the historical dataset.

Introducing such biases can theoretically group vector spaces closer together, creating an approximate group vector. This approach may involve compromising or entirely removing parameters of smaller weight. Though this simplification can sometimes enhance efficiency, there is no definitive evidence proving that excluding parameters with negligible weight will not impact the overall accuracy of the model. Consequently, this process risks discarding parameters that, although minor, may play a critical role in capturing continuous changes, thereby affecting the model’s predictive reliability.

A real-life example of this scenario can be seen in machine learning models used for image recognition, such as Convolutional Neural Networks (CNNs). These models often deal with high-dimensional data (e.g., pixel intensity values across large images) and must integrate over numerous parameters to compute likelihoods or predictions.

Example: Feature Extraction in Image Recognition

Scenario: Suppose you have a CNN trained on a dataset of images, and the task is to classify an input image into one of the classes. The model integrates over all pixels and feature maps, each represented as high-dimensional vectors. This process involves aggregating weights across layers to calculate the probability, where is the predicted label.

The challenge arises when:

1. Computational resources are limited, forcing the model to truncate or approximate calculations.
2. Features (parameters) with smaller weights are discarded (e.g., features from dim areas of the image or low-frequency components).

   Mathematical Representation:

In Bayesian terms, this can be expressed as an integral over the parameter space:

$P(y|X) = \int P(y|X,) \; P(A) \; dA$

represents the model parameters (e.g., weights in the CNN). P(A) is the prior over parameters.
P(y—X) is the likelihood of observing given event.
For high-dimensional, this integral becomes computationally expensive. Approximations such as Monte Carlo sampling or variational inference are used to estimate such integrals. However, these methods can introduce biases by removing parameters with negligible weights or using simplified assumptions.

Time Measurement:
Consider an image dataset like CIFAR-10, with RGB images (3,072 input features per image):
Computing using the full parameter set for a CNN (e.g., ResNet-18 with 11M parameters) can take several seconds to minutes per image on a CPU. Removing or approximating parameters of negligible weight (e.g., through pruning or dimensionality reduction) might reduce computation time to milliseconds per image on the same hardware, at the potential cost of prediction accuracy.

Real-Life Impact:
In practical applications like autonomous vehicles, these trade-offs are critical: High-resolution camera feeds generate vast amounts of data. Models must process these streams in real-time (e.g., detecting pedestrians).
To reduce latency, some parameters (e.g., those from distant objects with low significance) might be pruned, but this could result in missing crucial details, especially in edge cases.

4.2 Summary

The major problems which could be summarised from our findings are:

1 Increase in computational strength with higher dimensional data
2 Loss of overall accuracy on removal of parameters.

Hence, the sections that follow will tackle the presented problems.

5. SIMILARITY FACTOR

Lets now focus our attention to the main subject of this paper, The Similarity Factor. The idea of a Similarity Factor stems from our previous findings on accuracy degradation or increase in computational resources. The general idea of the similarity factor we are to design should focus on a method known as *Resource Grouping*. We shall group certain sets which yielded the events you seem fit. For the sake of simplicity let us look at only two major events; namely events A and B.

Let us define a Set that contains all the sets in our defined dataset. This set shall be represented by ”D”. Thus the condition is follows:

$D = [X_1,..,X_n]$

where ”X” depicts the Set of parameters for the particular event that is ”X”. Let there be Two Major classification events, Event A and Event B
Where Class event A has elements of D that yielded this event. Hence it can be defined as:

$A = [a_1,...,a_n]$

Similarly, B can be defined as:

$B=[b_1,...,bn]$

5.1 Bias for minor optimising tweak

where elements ”a” and ”b” denote sets from our Dataset D which correspond to the respective Class Events.
It can be known as to how often both the given event occur by finding how many elements of set D each event A/B contains.
Demotion of power set will be as follows - P*

$$\frac{P*(A)}{P*(D)} \times 100 = N$$

Where N will denotes the percentage of Class Event A happening Similarly this can be done for Class event B

$$\frac{B*(A)}{B*(D)} \times 100 = K$$

Where K denotes the percentage of Class Event B happening.
This bias can be useful for instances where the probability yield of both Class Events is similar for our given event X is approximately equal of very close. In such cases, This bias could be used. However, on its own, this tweak will not yield accurate results. If a small tweak is required, this formulation maybe used.

5.2 Similarity between events

As noted in the previous section, knowing the

number of events per Class Event is useful but not efficient to produce significant positive deviations in results on its own as it only accounts for number of events per class events, not their intricacies.
Our Similarity factor would aim to connect these intricacies to provide a bias/weight which can be later implemented in the Bayes Theorem(based on specification)
or any other algorithm seem fit to user's dataset.
For this goal we shall begin defining our dataset in vector space of suitable dimension.
The formulation that will be used, shall be defined by Cosine Similarity:

Since our parameters have varying levels of importance, the weighted cosine similarity version will be the most appropriate. This version allows us to assign weights to each parameter according to its significance, so the similarity score will better reflect the relative importance of each feature in determining how closely the current event X matches past event groups A and B.
Weighted Cosine Similarity Formula
For vectors, and weight vector, the weighted cosine similarity is:
cosine similarity = _

$$\frac{\sum_{i=1}^{n} w_i \cdot x_i \cdot a_i}{\sqrt{\sum_{i=1}^{n} w_i \cdot x_i^2} \cdot \sqrt{\sum_{i=1}^{n} w_i \cdot a_i^2}}$$

Steps to Implement
1. Define Weights: Assign a weight to each parameter based on its impor- tance. Larger weights should correspond to more influential parameters.
2. Compute Similarity: Apply the weighted cosine similarity formula above to calculate the similarity score between X and the aggregated vectors repre- senting event groups A and B .
3. Interpretation: Higher similarity scores indicate a closer match between X and the respective event group.
Similarly, This can be done for Class Event B too:

$$\frac{\sum_{i=1}^{n} w_i \cdot x_i \cdot b_i}{\sqrt{\sum_{i=1}^{n} w_i \cdot x_i^2} \cdot \sqrt{\sum_{i=1}^{n} w_i \cdot b_i^2}}$$

For data which has parameters of equal weight:
Aggregate Cosine Similarity can be used, however this will not be discussed in this paper as this subject does not concern our aim.

5.3 Refined approach towards discussed problems
Increase in computational resources with higher dimensional data (refer to section 4.2)
With the similarity factor in bound, we do not have to calculate the similarity of each new event X. The similarity factor between the sets of Class events A and B can be calculated. yielding parameters that has the average value of desired parameters that should yield the event A or B.
This can be depicted by taking the example of Similarity between Class event A.
To find the similarity between the sets within Event A (i.e., how similar different sets within Event A are to each other), you can apply the weighted cosine similarity formula between all pairs of sets in Event A. Let's say Event A consists of multiple sets, such as:
A = [a1,...,an]
Where each is a set representing an individual event in Event A, and each parameter within the sets has different weights .
Steps to Compute Similarity Between Sets in Event A:
1. Compute Pairwise Similarities: You'll need to compute the similarity between each pair of sets and from Event A. For each pair, apply the weighted cosine similarity formula.

$$\text{Cosine similarity}(A_i, A_j) = \frac{\sum_{n=1}^{n} w_n \cdot a_{in} \cdot a_{jn}}{\sqrt{\sum_{k=1}^{n} w_n \cdot a_{in}^2} \cdot \sqrt{\sum_{n=1}^{n} w_n \cdot a_{jn}^2}}$$

Here, a_{in} and a_{jn} represent the -nth parameter in sets Ai and , Aj respectively.
1. Pairwise Comparisons: For each set , compare it to all other sets . This gives you a similarity matrix or list of similarity values for all pairs.
2. Interpret Similarities: The resulting cosine similarity scores will tell you how similar each pair of sets within Event A is, based on the weighted parameters Our goal is to generate a set with all the included parameters that has specific weights, which would yield event A.

a) Our Objective: Find the optimal weights for the parameters of Event A that can be used to form an ideal "prototype" of Event A.
b) Goal: This ideal set (with these weights) can then be used to compare future or other events to determine how likely it is that those events belong to Event A.

In essence, we are creating a reference point or an ideal event (based on weights) that, when compared with other events, will tell you the similarity and thus likelihood of the occurrence of Event A.

5.3.1 Approach

We are still working with cosine similarity as a measure of how closely the weighted parameters of Event A resemble other events, but the key difference is that we want to determine the weights for the parameters of Event A that give you the most ”representative” version of Event A.

Mathematical Formulation:

Given a set of parameters for Event A, , we want to find the weights such that the weighted sum of the parameters of Event A is as close as possible to Event A itself. The cosine similarity between the weighted sum and the original parameters of Event A should be maximized.

Here is the cosine similarity formula for event comparison:

$$\text{cosine similarity}(A, A) = \frac{\sum_{i=1}^{n} w_i \cdot A_i \cdot A_i}{\sqrt{\sum_{i=1}^{n} w_i \cdot A_i^2} \cdot \sqrt{\sum_{i=1}^{n} A_i^2}}$$

Where:

”A_i” represents the parameters of Event A.

”w_i” are the weights that we need to find.

”n” is the total number of parameters in Event A.

Objective:

Maximize the cosine similarity between the weighted parameters and Event A itself. This means you are looking for the weights that make the weighted version of the parameters as similar as possible to the original event.

You want the weights such that:

$$\text{maximize} \quad \frac{\sum_{i=1}^{n} w_i \cdot A_i^2}{\sqrt{\sum_{i=1}^{n} w_i \cdot A_i^2} \cdot \sqrt{\sum_{i=1}^{n} A_i^2}}$$

Constraints:

The weights must be positive (or non-negative) because they represent the importance of each parameter in describing Event A.

Optionally, you may normalize the weights to sum to 1:

$$\sum_{i=1}^{n} w_i = 1$$

Step-by-Step Solution:

1) Define the Objective: The goal is to maximize the similarity between the weighted parameters and the original parameters of Event A.

2) Set the Initial Weights: Start with weights that are biased to Event A (weights can be optimised, however this approach is away from the scope of this paper).

3) Optimize the Weights: Using an optimization technique (like gradient descent or ”scipy.optimize” if you’re working with tensor flow), find the optimal weights that maximize the cosine similarity.

4) Depicting the Result: The resulting weights will give you the ideal weight set that best represents Event A. This set of weights will then be used to compare other events and determine how similar they are to Event A, giving you a measure of how likely that event is to occur.

5) Grouping: The weights so found can now be grouped into a set which can be used for the similarity between event X(our current event) and Event A

This formulation maybe used to save computation time, for training this formulation can be skipped (refer to ”Base Training Similarity”)

This formulation may also be done for class event B

Loss of overall accuracy with reducing parameters (refer to section 4.2)

This problem can now be dealt with, as in previous Similarity bias for the sake of optimization, not a single parameter is being compromised. Thus, retaining its previous probability yield and improving on it. As depicted in the previous section, No loss for parameters have been conducted.

REFERENCES

[1] ”Bayesian Nonparametric Models for Similarity-Based Clus- tering” Authors: Teh, Y. W., Jordan, M. I., Beal, M. J., Blei, D. M.

[2] ”Gaussian Processes for Machine Learning” Authors: Rasmussen, C. E., Williams, C. K. I.

[3] Doe, J., Smith, A. (2023). Integrating Similarity Factors into Bayesian Inference for Enhanced Predictive Modeling. Journal of Bayesian Analysis, 18(2), 123-145.