Probabilistic Learning of Operator Interest in Surveillance Environments for Online Track Characterization

by

Eli Kravitz

B.S., University of Vermont, 2018

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Ann and H.J. Smead Aerospace Engineering Sciences Department

2023

Committee Members:

Nisar Ahmed, Chair

Zachary Sunberg

Danielle Szafir

Kravitz, Eli (M.S., Aerospace Engineering)

Probabilistic Learning of Operator Interest in Surveillance Environments for Online Track Characterization

Thesis directed by Prof. Nisar Ahmed

Understanding user interests and behaviors is an important aspect to consider when developing machine learning models to augment data-intensive human workflows. By understanding what a user desires, these models can help prioritize workflow and essential information for decision making the way a human would, and form the baseline for trusted autonomous systems. This work considers the problem of human interest classification in the context of missile defense surveillance. In this case, users are satellite operators who must prioritize simultaneous processing and characterization of multiple (e.g. hundreds of) target tracks across the globe for extended periods of time under strict time and accuracy constraints.

The given problem is particularly challenging because user interest is not generally static, tracks have a short lifespan, and the user pool is small. Here, the problem is formulated similarly to a binary Bayesian logistic regression problem to classify operator interest in a given candidate track, but with the added complexity of partially observable input track feature variables and dependencies between these variables. Prior feature weight distributions are developed based on human inputs, and labeled training data is generated from online user interactions with the system. Derivations for a novel probabilistic model lead to intractable conditional posterior distributions for regression feature weights. These distributions are approximated in real-time online. Given trained weight distributions, human interest for unlabeled candidate tracks is inferred.

The developed interest classification model is validated using a simulated truth model by examining average track interest, learning rate, and Discounted Cumulative Gain for

assessing relevancy of classified tracks. Validation is performed on several use cases, including an ideal case as well as several non-ideal cases. The results indicate that the model is robust to challenging cases, and is able to accurately learn features that interest a human operator. The introduction of a probabilistic model allowed for interest to be learned with limited prior information or labeled training data, and addressed the challenging problem of inferring user interest in a surveillance environment.

Dedication

To my Grandpa Kravitz.

Acknowledgements

I would like to thank the following people, all of whom helped make this thesis possible.

Professor Nisar Ahmed, for giving me the opportunity to work in the COHRINT lab, providing me with the foundation that much of this work is based on, and for giving guidance throughout this process.

Hunter Ray, for being there to bounce ideas off of, discuss math, and draw on whiteboards.

Nick Conlon and Ian Thomas for helping to convert ideas into software.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

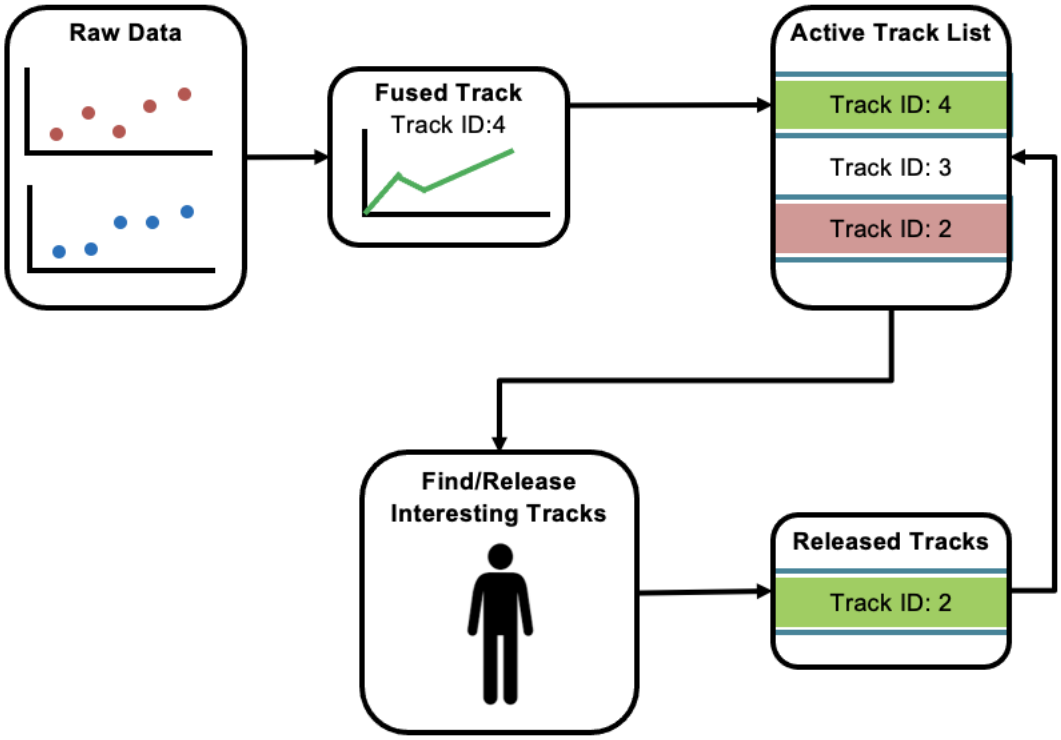
The US Military possesses a constellation of eight Overhead Persistent Infrared (OPIR) satellites whose mission is to maintain surveillance on targets of interest to the U.S. and its allies [28]. These satellites exist in geostationary orbit and are able to consistently feed battlespace information to operators in the OPIR Battlespace Awareness Center (OBAC) at Buckley Space Force Base in Colorado. In this context, surveillance generally entails detection, tracking, and object classification. This system is able to start and maintain tracks autonomously, but requires significant user input for object classification. As such, achieving the full capability of this system entails considerable effort from highly-trained operators. A block diagram of the problem space is shown in Figure 1.1.

At a high level, the system works as follows: Raw IR data is fused by a tracker to create a track[1] with an associated Track ID. All active tracks are then placed into an "Active Track List", which is sorted in chronological order (with newest tracks being presented first). As operators interact with the system, they have the ability to assign an object type to a track and release it downstream to analysts who perform more detailed and time intensive follow-up examinations of the data (i.e. for intelligence gathering purposes). After release, the Active Track List indicates that an operator has released a specific Track ID.

In terms of operator tasking, the general workflow is that personnel in the OBAC interrogate tracks before making an object type classification and releasing Track IDs down-

[1] A "track" is defined as a collection of persistently detected signal returns that are correlated to a possible object of interest with physical states evolving over time and space.

Figure 1.1: High-level problem space block diagram.



stream. Because tracks are presented in chronological order, the system requires users to manually scroll through the Active Track List to identify targets that are of potential interest to them. Given that many tracks are simultaneously active across the entire globe (e.g. with potentially many dozens being processed at a time over the course of many hours), this process is very time-consuming and labor-intensive. In an effort to expedite operator workflow, this work considers the task of augmenting current OBAC capabilities with machine learning (ML) in order to automate the "Find/Release Interesting Tracks" block in Figure 1.1. This task is made challenging by a variety of factors, including imperfect object type classification (such that an operator cannot directly tell the system to only show them certain object type tracks), limited user interaction with the system (such that there is limited labeled training data for an ML model), and rapidly changing CONOPS (such that the model may need to "start fresh" frequently). The concept of supplementing human workflow with machine intelligence is not unique to this problem, and many solutions have been considered across both industry and academia.

By and large, as human-machine interactions become more frequent, the need to augment human workflow with intelligent systems becomes paramount. As such, User Assistance Systems are employed to improve and expedite interactions between human operators and software. These systems seek to integrate into, and enhance the user's experience. There are many important factors in this process, such as how to present information to humans, how to augment their capabilities without limiting them, and how to learn what they find important.

Many systems have already been deployed to tackle this problem across numerous industries. Initial attempts included Microsoft's "Clippy", a paperclip assistant to Microsoft Office products that aimed to give users guidance within their applications. This product was widely disliked, with users claiming that it disrupted workflow rather than enhancing it [24]. Recently, there has been a surge of AI-based tools aimed at increasing productivity [9]. These tools have been met with mixed reviews and success, but despite their presence, some

research suggests a general downward trend in labor productivity in major economies since the turn of the century [7]. While this dip in productivity does not indicate that AI-based tools actively limit workers' capacity, it does call into question their utility.

Despite the questionable efficacy of many productivity tools, systems designed specifically for recommending interesting items to humans have produced desirable results across industry. Netflix uses personalized recommendations to increase their effective catalogue size, retain existing customers, and add new customers. As of 2015, they estimated that these algorithms saved them in excess of $1B annually [17]. Other major tech companies, such as Amazon and YouTube [32] [10], have leaned on sophisticated and evolving personalized recommendation systems to increase their market share and profitability for decades. Much of the existing literature on human interest learning is concerned with massive datasets across a vast user base.

This work seeks to create a recommendation system that sorts tracks not by time, but rather by potential operator interest. This context is different from those studied in existing literature for two main reasons. Firstly, in the OBAC, there is a very limited user pool, as opposed to Netflix or Amazon who each have millions of users. Secondly, while a video or product may exist for months or years on a platform, a track may only be active for several minutes. Despite the lack of similarity to existing literature, the end goal is very similar: to create a human interest classifier.

The main contribution of this work is the development of a framework that can be eventually applied to determine human interest in IR tracks for OBAC operations. These results are used to rank active tracks and sort them based on interest likelihood. This task is particularly difficult because, unlike in existing applications, the available data is very limited (in terms of tracks labeled for interest) and the tracks have a relatively short lifespan. These challenges lead to the development of a probabilistic model that is able to learn human preferences through user interactions with the system. While this thesis is concerned with the specific context informed by OBAC operations, the developed model can be generalized

to other applications with the goal of performing discrete classification given limited labeled training data.

In Chapter 2, the problem statement is introduced and related work is presented more formally. Problems with existing approaches as they pertain to this application are also discussed. Chapter 3 introduces the models that were developed to define and learn human interest parameters, as well as to infer interest in a specific track. Here, the theory and mathematical derivations necessary for implementation are introduced. Chapter 4 presents simulation results in several cases: an ideal scenario, challenging edge case scenarios, and a more realistic scenario. The intent of this section is to both show that the model performs well in a "perfect world", but is also robust enough to handle realistic and highly challenging cases. Finally, conclusions and future work are presented in Chapter 5.

Chapter 2

Background & Related Work

This chapter presents background and related work that pertains to the work introduced in subsequent chapters. Section 2.1 formally defines the problem statement. Section 2.2 provides an overview of important concepts and a basis for the following literature review. Next, Section 2.3 introduces existing approaches to solving the defined problem, along with related work. Section 2.4 illustrates problems with outlined approaches as they pertain to this specific context. Lastly, Section 2.5 introduces the standard Bayesian logistic regression framework, as it serves as an inspiration for this work.

2.1 Problem Statement

Given imperfect object type classification, changing operator CONOPS, and limited access to labeled training data (i.e. tracks labeled as interesting or not interesting), the goal of this work is to learn which tracks are likely to interest a particular user. By understanding what a human operator finds interesting, the manual process of flagging interesting tracks can be automated.

2.2 Overview

As stated above, the goal of this work is to develop a framework to learn human interest. With this task in mind, two important concepts are defined below:

- Classification: categorizing something into a specific group based on some vector of characteristics.
- Preference elicitation: development of a decision support system with the goal of generating recommendations to users.

This work exists on the boundary of the concepts of classification and preference elicitation. While learning interest is fundamentally a classification problem, in this context, classification is done in an effort to generate recommendations to users. As such, the remainder of this chapter provides background on both fields of study.

2.3 Literature Review

In this section, existing literature regarding both classification and preference elicitation is presented. These techniques are relatively simple to implement, well-documented, and have significant support in both research and industry. Note that this is just a small subset of methods, as both subjects have significant breadth.

2.3.1 Classification

2.3.1.1 Neural Networks

Neural Networks (NN) were first proposed by McCulloch and Pitts in 1943 [37], but have gained significant popularity in recent years [30] as computing power has increased drastically. NNs are motivated by the structure of the human brain [34], and have been employed in many use cases. At a high level, NNs receive data at an input layer, perform mathematical operations in hidden layers, and present a classification in the output layer.

NNs have been used for tasks such as speech recognition, computer vision, and medical diagnosis [1]. Furthermore, they have been applied to problems of understanding human psychology and interest. Almeida and Azkune [4] used a recurrent NN model to predict human behavior and identify atypical behaviors. Liu et. al [23] used convolutional NNs to

determine objects of human interest in RGB images. Given the significant research invested in NNs, a comprehensive literature review is not feasible in this setting, but it can be stated that they are applicable in many classification tasks.

2.3.1.2 Decision Trees

Decision trees are a rule-based method for determining discrete classifications from continuous or discrete input features. The architecture is such that the tree begins at a root node, where all labeled training data is contained by the node. Next, the data passes through a decision node, where some condition is placed on the input features such that the labeled data is split into two subsequent nodes, each containing some subset of the labeled data. This process continues until all leaf nodes (a node with no children) contain no impurity (all the leaf nodes contain only one class) or a maximum depth is reached. At each node, a condition is defined to maximize some entity. In general, Information Gain is maximized, such that:

$$IG = E(\text{parent}) - \sum_{i=1}^{\#\text{ children}} w_i \cdot E(\text{child}_i) \ . \tag{2.1}$$

In Equation 2.1, E represents entropy of a node and w_i represents the proportion of points from the parent that exist in child_i. Entropy is defined as:

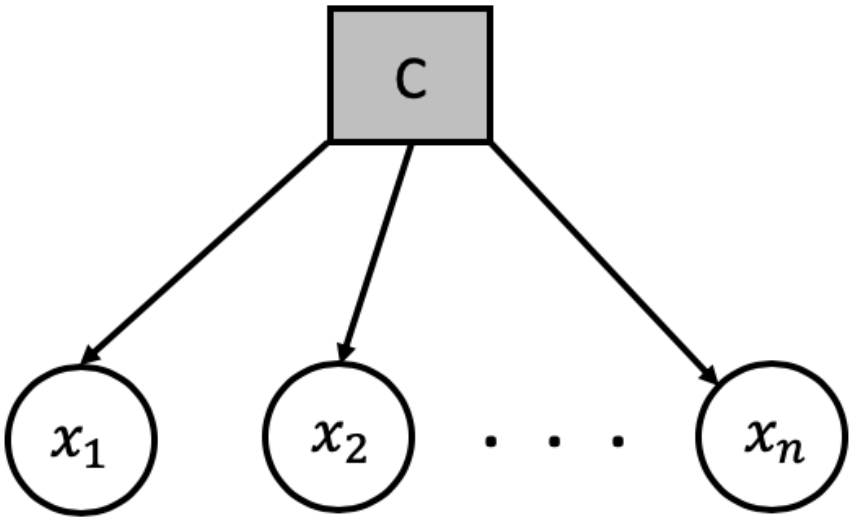
$$E = \sum_{i=1}^{\#\text{ classes}} -p_i \cdot \log_2(p_i) \ . \tag{2.2}$$

In Equation 2.2, p_i represents the prior probability of class i. Using this methodology, trees can be formed such that test data flows through the entire tree to a leaf node, where a classification is made. Kolo et. al [20] used decision trees to predict the academic performance of students with mixed results. Kim et. al used Decision Trees [18] to predict human interest in an effort to personalize advertisements on an internet storefront. Their results showed significant improvement over random advertisement sorting.

2.3.1.3 Probabilistic Generative Models

Many probabilistic generative models have been used for classification tasks. The simplest model is a Naive Bayes Classifier (NBC). It is generative in the sense that its goal is to maximize the joint probability rather than maximizing the conditional probability, and also, as the name suggests, allows synthetic data to be generated from the trained model. The graphical model for this classifier is shown in Figure 2.1.

Figure 2.1: Bayesian Network (BN) model for Naive Bayes Classifier. Note that the classification label is unobserved, and the features are observed.



This model considers a categorical classification variable and continuous/categorical feature variables. Classification probabilities can be defined as follows:

$$\begin{aligned} p(C = c \mid \vec{x} = \vec{x}) &= \frac{p(C = c, \vec{x} = \vec{x})}{p(\vec{x} = \vec{x})} \\ &= \frac{p(C = c)p(\vec{x} = \vec{x} \mid C = c)}{\sum_C p(C)p(\vec{x} = \vec{x} \mid C)} \,. \end{aligned} \tag{2.3}$$

The conditional independence properties of the model allow for the above to be simplified, because all features, x_i are independent given the classification, C:

$$p(C = c \mid \vec{x} = \vec{x}) = \frac{p(C = c) \prod_{i=1}^{n} p(x_i = x_i \mid C = c)}{\sum_C p(C) \prod_{i=1}^{n} p(x_i = x_i \mid C)} \ . \tag{2.4}$$

For this model to be used as a classifier, the conditional distributions $p(x_i \mid C = c)$ must be defined for all $i \in 1 : n$ and $c \in C$, as well as the distribution $p(C)$. Given these conditional probability distributions, the classification probability is straightforward to calculate using Equation 2.4. This model is also very useful in the sense that missing features, after being integrated out, are simply ignored:

$$p(C = c \mid \vec{x} = \vec{x}) = \frac{p(C = c) \int p(\vec{x} = \vec{x} \mid C = c) d\vec{x}_m}{\sum_C p(C) \int p(\vec{x} = \vec{x} \mid C) d\vec{x}_m} \ . \tag{2.5}$$

In the above, the term $\vec{x}_m$ represents the vector of missing features that are marginalized out of the calculation. We note that $\int p(\vec{x}_m \mid C) d\vec{x}_m = \sum_{\vec{x}_m} p(\vec{x}_m \mid C) = 1$, which simplifies Equation 2.5 to (noting that the set $\{j\}$ signifies the indices of all missing features):

$$P(C = c \mid \vec{x} = \vec{x}) = \frac{p(C = c) \prod_{i=1, i \neq \{j\}}^{n} p(x_i = x_i \mid C = c)}{\sum_C p(c) \prod_{i=1, i \neq \{j\}}^{n} p(x_i = x_i \mid C)} \ . \tag{2.6}$$

NBCs have been used in a wide variety of applications due to their simplicity to implement and often competitive results. Sahami et. al trained an NBC to detect spam emails (i.e. a spam filter) [29]. Using this methodology, they were able to achieve $> 97\%$ precision and $> 94\%$ recall in detecting spam emails.

While these classifiers often perform well, the conditional independence assumption is nearly always violated in real-world scenarios. The NBC model can be updated to include other dependencies between variables [21] [38], but these changes make the model more difficult to define and evaluate.

2.3.2 Preference Elicitation

2.3.2.1 A/B Testing

A/B testing is a framework where two or more versions of a variable are developed, and some percentage of the user base is presented with each version at random. After some interaction time, statistical analysis is done to determine which version performed the most favorably. This methodology can be used in a variety of settings in order to choose the most ideal developed version of a variable.

Kohavi and Longbotham [19] introduce the use of A/B testing for in the context of advertisements, and give motivating background for their efficacy. Bhat et al. [5] used an A/B testing framework to test different versions of a treatment in a medical setting. The authors extended the most basic version of A/B testing to near-optimally choose the correct treatment (as opposed to random assignment). Many other use cases and variations of the technique have been explored in literature.

2.3.2.2 Multi-Armed Bandit

Multi-armed bandits (MABs) are a widely used framework that are employed to choose optimal actions. They were first introduced in the 1930's [35] and have seen a recent surge in interest [31]. The basic concept behind a MAB is that each action represents an arm with some unknown reward distribution, and pulling an arm leads to a realization of the reward. The main goal of the MAB framework is to simultaneously learn the reward of pulling a specific arm, while also maximizing the long-term payoff (i.e. trading off between exploring the space and exploiting what is already known). This is equivalent to minimizing long-term *regret*, defined below:

$$R(T) = E\left[\sum_{t=1}^{T} r_{t,a_t^*}\right] - E\left[\sum_{t=1}^{T} r_{t,a_t}\right]. \tag{2.7}$$

In Equation 2.7, $R(T)$ is the regret over T trials (the horizon in question), r_{t,a_t^*} repre-

sents the reward for the optimal arm at time t, and r_{t,a_t} is the reward received for the arm that was pulled at time t. Note that regret is difficult to evaluate in a real-world setting because the true reward distribution of an arm is unknown.

Consider the simplified case of a movie recommendation system where there are many users, a fixed set of movies, and the goal is to show a single movie that is most likely to interest a user. Here, each movie is considered an arm that can be pulled. If the system recommends a movie (pulls the arm), a reward of 1 is obtained if the user clicks on it, and 0 otherwise. The MAB framework seeks to exploit the arm that gives the greatest reward, while also exploring other alternatives.

This process can be considered in a Bayesian framework, where there is a prior distribution on the reward for each arm, which is recursively updated when true rewards are realized:

$$p(\theta \mid a_t, r_t) = \frac{p(r_t \mid a_t, \theta)p(\theta)}{p(r_t \mid a_t)} \,. \tag{2.8}$$

In the above, a_t and r_t represent an action and reward at time t, respectively, and θ represents the vector of success probabilities for each arm. A Beta-Bernoulli model is often used [2], where, for each arm (or action) $p(r_t \mid \theta)$ is Bernoulli distributed and $p(\theta)$ is Beta distributed. This model is convenient due to its conjugacy, allowing for the posterior to be updated in closed form:

$$p(\theta \mid r_t) = \begin{cases} \text{Beta}(\alpha + 1, \beta) & \text{if } r_t = 1 \\ \text{Beta}(\alpha, \beta + 1) & \text{otherwise.} \end{cases} \tag{2.9}$$

The challenge arises in choosing the trade off between exploring and exploiting the arms. Many methods exist to determine which arm to pull next, including Epsilon Greedy, Thompson Sampling, and Upper Confidence Bound (UCB) [33]. Epsilon Greedy chooses the best arm with probability $1 - \epsilon$ (where $\epsilon \in (0, 1)$ is a tuning parameter representing the

probability of choosing an arm at random), Thompson Sampling takes a sample from each arm's reward distribution and chooses the arm with the highest sampled reward, and UCB chooses the arm with the highest confidence bound (often 95%). While other approaches exist, these are among the simplest and most straightforward.

Acuna et. al [2] showed that the MAB framework accurately modeled sequential human decision making in the context of playing slot machines (from which the bandit problem gets its name). They suggest that given the performance of the model, humans likely perform approximate Bayesian inference with minimal look-ahead in mind heads as a means of making decisions. MABs have also been shown to perform competitively in recommendation systems [13] [25] as a means of circumventing the "cold-start problem" (a problem that arises in recommendation systems when no knowledge of the user or arm is available a priori).

2.3.2.3 Contextual Multi-Armed Bandit

The example given in Section 2.3.2.2 is clearly limited in the sense that it ignores context. Neither user context (age, gender, etc.) or movie context (genre, run-time, etc.) are considered in the problem. The lack of user context assumes that all users are the same (unless broken into groups before running MAB algorithms), and the lack of movie context removes the ability to learn about other arms by pulling one arm. This problem was addressed with the introduction of the Contextual Multi-armed Bandit (CMAB), also known as a bandit with side information [36]. Li et. al [22] examine the CMAB in the context of user article recommendation and formulate the problem as being linear in the context of a feature vector belonging to an arm/user combination:

$$E[r_{t,a} \mid \vec{x}_{t,a}, \vec{z}_{t,a}] = \vec{z}_{t,a}^{T}\vec{\beta}^{*} + \vec{x}_{t,a}^{T}\vec{\theta}^{*}. \tag{2.10}$$

In Equation 2.10, $\vec{z}_{t,a}$ is a feature vector shared among arm/user combinations, $\vec{x}_{t,a}$ is a feature vector unique to a specific arm/user combination, and $\vec{\beta}^{*}$ and $\vec{\theta}^{*}$ are optimal

weights for each feature. They define an algorithm called LinUCB to choose an arm, which is an extension of the vanilla UCB algorithm specifically designed for the contextual bandit application. The authors applied this algorithm to a Yahoo! Front Page Today Module dataset, and observed a 12.5% increase in clicks over a context-free MAB, with greater improvements in data-sparse environments.

As with the context-free MAB, many variations of the CMAB have been proposed and researched. In general, the inclusion of context minimizes long-term regret relative to the context-free MAB.

2.4 Problems with Outlined Approaches

While useful in many contexts, for a variety of reasons, the approaches outlined above do not extend particularly well to the problem considered in this work. The shortcomings of the discussed methods are outlined below.

Both NNs and decision trees are useful for classification in settings where significant training data exists. However, in the context considered in this work, training data is limited. Incorporating prior domain knowledge may be a useful technique to circumvent the lack of labeled data, but neither of these methods naturally allow for the incorporation of domain knowledge (they are generally data-driven).

Probabilistic generative models like NBCs (or semi-Naive Bayes Classifiers) could be a potential solution to this problem due to their flexibility as classifiers. However, the conditional distributions $p(x_i \mid C)$ are difficult to define a priori due to both lack of information and their lack of intuitive form. The task of defining priors becomes increasingly difficult when individual features x_i become multidimensional. For example, in the OBAC setting, intelligence data might indicate that altitude and geographic location are correlated given a specific object type. In this case, altitude, latitude, and longitude might be considered as a single multidimensional feature $\vec{x}_i$. Conceptualizing a prior distribution over this three-dimensional feature that is representative of truth is generally difficult, and becomes in-

creasingly non-intuitive as dimensionality increases beyond human comprehension (however, without grouping correlated features, significant dependencies are ignored). These distributions are also difficult to learn online given that there is a limited amount of labeled data (i.e. cases where C is realized). This means that the classifier would be unlikely to perform well until the conditional distributions were adequately trained, or tight and accurate priors were defined based off of human intelligence. Due to these limitations and considering that the problem at hand is fundamentally a classification problem, a discriminative model may be a more ideal choice in this context. Discriminative models often produce better results than generative models in a classification context, and tend to converge more quickly in the training phase [6].

A/B testing, while plausible in many preference elicitation settings, is not a natural solution for this work. In general, their use requires a significant user base, which does not exist in the context of the OBAC. Also, some similarity is assumed between users, which is violated in the considered context, as operators may have drastically different CONOPS.

The MAB framework is useful for pulling a single arm (in this context, showing a single relevant track) that will receive positive or negative operator feedback. However, here we are interested in displaying many possibly interesting tracks (i.e. we do not want to limit the algorithm to only choose a single track to display) to a user without a guarantee that a reward will be realized. Also, an arm (or in this case a track) only exists for a very limited period of time. Conversely, in the movie selection example outlined above, the media exists from the day it is uploaded on the platform until it is removed, often a matter of months or years. Given the short temporal history of a track, feedback on any single arm would be very limited. Furthermore, context of both a user and track is available during OBAC operations, which are ignored by the standard MAB formulation.

CMABs alleviate some of the problems associated with MABs, but the issue of pulling a single arm and receiving feedback still remains a problem. Because we desire to display more than one track, which essentially represents pulling multiple arms simultaneously, and

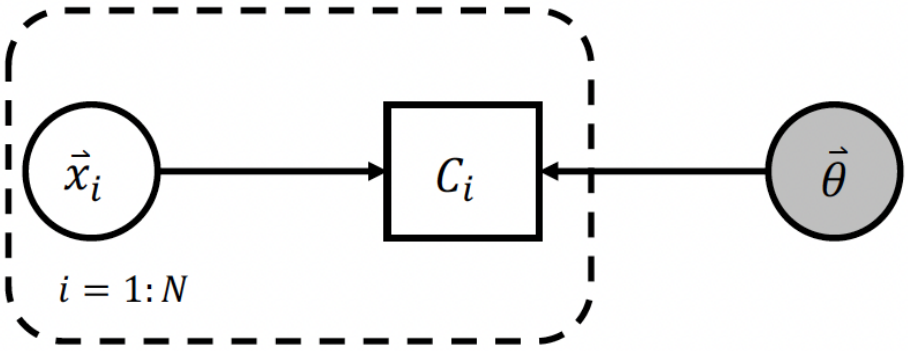
feedback is not a guarantee, the application does not naturally fall into the Multi-armed Bandit framework.

The approaches outlined above are among many efforts to understand human psychology using machine learning techniques. Despite significant effort on this front, much of the research on the topic is application-specific, and generally requires a substantial amount of training data. As opposed to massaging the problem to fit a stock solution, a model designed specifically for this problem is defined in Chapter 3.

2.5 Bayesian Logistic Regression

Here, the Bayesian logistic regression model and associated mathematics are introduced. A logistic regression is a discriminative model used to predict the probability of some categorical classification given all observed information. In particular, we consider the case of binary classification. The graphical model associated with parameter learning for Bayesian logistic regression is shown in Figure 2.2. The notation used here will be used in future sections. Firstly, circular nodes represent continuous random variables, while rectangular nodes represent discrete random variables. Next, shaded nodes are unobserved, while unshaded nodes are observed. Finally, the dashed box indicates a plate model representation.

Figure 2.2: BN for Bayesian logistic regression weight learning.



In the graphical model, the random variables are:

- $\vec{x}_i$: i^{th} observed feature set
- C_i: i^{th} observed classification (binary in this case)
- $\vec{\theta}$: Unobserved feature weighting vector.

The joint distribution is defined as:

$$p(\vec{\theta}, \vec{x}_{1:N}, C_{1:N}) = p(C_{1:N} \mid \vec{x}_{1:N}, \vec{\theta}) p(\vec{x}_{1:N}) p(\vec{\theta}). \tag{2.11}$$

Although the distributions $p(\vec{x})$ and $p(\vec{\theta})$ are important to denote in general, for the purpose of this brief introduction, they need not be explicitly defined. The conditional distribution $p(C \mid \vec{x}, \vec{\theta})$ is however important to define, as its denotation lends the logistic regression its name. For binary classification, this conditional posterior is parameterized as a logistic sigmoid:

$$p(C = 1 \mid \vec{x}, \vec{\theta}) = \frac{1}{1 + \exp(-\vec{\theta}^T \vec{x})} \,. \tag{2.12}$$

Given observed input features and labels, the posterior distribution over $\vec{\theta}$ can be defined as follows:

$$\begin{aligned} p(\vec{\theta} \mid \vec{x}_{1:N}, C_{1:N}) &= \frac{p(\vec{\theta}, \vec{x}_{1:N}, C_{1:N})}{p(\vec{x}_{1:N}, C_{1:N})} \\ &= \frac{p(C_{1:N} \mid \vec{x}_{1:N}, \vec{\theta}) p(\vec{x}_{1:N}) p(\vec{\theta})}{\int p(C_{1:N} \mid \vec{x}_{1:N}, \vec{\theta}) p(\vec{x}_{1:N}) p(\vec{\theta}) d\vec{\theta}} \\ &= \frac{p(C_{1:N} \mid \vec{x}_{1:N}, \vec{\theta}) p(\vec{\theta})}{\int p(C_{1:N} \mid \vec{x}_{1:N}, \vec{\theta}) p(\vec{\theta}) d\vec{\theta}} \\ &= \frac{\prod_{i=1}^{N} p(C_i \mid \vec{x}_i, \vec{\theta}) p(\vec{\theta})}{\int \prod_{i=1}^{N} p(C_i \mid \vec{x}_i, \vec{\theta}) p(\vec{\theta}) d\vec{\theta}} \,. \end{aligned} \tag{2.13}$$

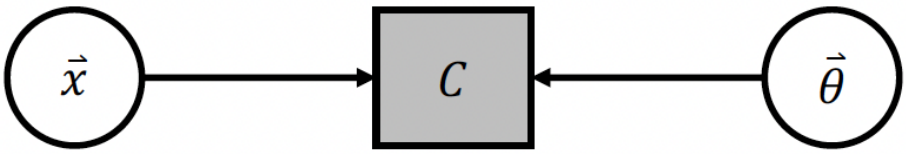
The posterior defined in Equation 2.13 generally does not lend an analytical solution. Furthermore, if $\vec{\theta}$ exists in high-dimensional space, it is also infeasible to numerically calculate

the integral in the denominator. As such, other approximation techniques must be employed, which will be discussed in later sections.

When inferring the probability of a certain binary classification for unlabeled data, one could assume that $\vec{\theta}$ becomes observable as the maximum a posteriori (MAP) estimate from the posterior defined in Equation 2.13. The graphical model for inference with observed weights is shown in Figure 2.3. Inference can then be computed as:

$$\begin{aligned} p(C \mid \vec{x}, \vec{\theta}) &= \frac{p(C, \vec{x}, \vec{\theta})}{p(\vec{x}, \vec{\theta})} \\ &= \frac{p(C \mid \vec{x}, \vec{\theta}) p(\vec{x}) p(\vec{\theta})}{\sum_C p(C \mid \vec{x}, \vec{\theta}) p(\vec{x}) p(\vec{\theta})} \\ &= \frac{p(C \mid \vec{x}, \vec{\theta})}{\sum_C p(C \mid \vec{x}, \vec{\theta})} \, . \end{aligned} \tag{2.14}$$

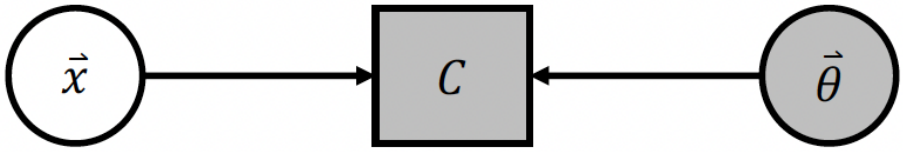
Figure 2.3: BN for Bayesian logistic regression inference with observable weights.



Without assuming a MAP estimate of $\vec{\theta}$, the class probability requires integration over all possible $\vec{\theta}$. The graphical model for inference with unobserved weights is shown in Figure 2.4. This is less straightforward, but will give more accurate results if the distribution is non-symmetric, and in particular if it is not unimodal. In this case, class probability is computed as:

$$
\begin{aligned}
p(C \mid \vec{x}) &= \frac{p(C, \vec{x})}{p(\vec{x})} \\
&= \frac{\int p(C \mid \vec{x}, \vec{\theta}) p(\vec{x}) p(\vec{\theta}) d\vec{\theta}}{\sum_C \int p(C \mid \vec{x}, \vec{\theta}) p(\vec{x}) p(\vec{\theta}) d\vec{\theta}} \\
&= \frac{\int p(C \mid \vec{x}, \vec{\theta}) p(\vec{\theta}) d\vec{\theta}}{\sum_C \int p(C \mid \vec{x}, \vec{\theta}) p(\vec{\theta}) d\vec{\theta}} .
\end{aligned}
\tag{2.15}
$$

Figure 2.4: BN for Bayesian logistic regression inference with unobservable weights.



Bayesian logistic regression models are flexible and can be used for many classification problems. Variations of the simple model outlined above have been used for applications from text categorization [14] to distinguishing between blasts and seismic events in mines [11].

Chapter 3

Modeling & Technical Approach

This chapter introduces the modeling choices made in an effort to learn human interest. An introduction to the solution approach is outlined in Section 3.1. Here, a high-level solution is outlined, and background is given regarding track data and human interaction with the system. The specific modeling choice is outlined in Section 3.2.

In Section 3.2.1, a model is introduced that learns human interest weights (similarly to Bayesian logistic regression), along with all necessary derivations. Interest prediction is discussed in Section 3.2.2. Here, an updated model is presented for inferring human interest along with the mathematical derivation for interest prediction. The full system-level workflow is considered in Section 3.3, and model limitations are addressed in Section 3.4.

3.1 Solution Approach Introduction

This section describes the solution approach as inspired by OBAC operations. This includes a high-level block diagram, background on available track data, and expected human interaction with the system.

3.1.1 High-level Proposed Solution Approach

A block diagram of the solution approach is shown in Figure 3.1. Existing blocks include:

- Track data: physical data (intensity, altitude, position, etc.) for all live tracks

- Object type classifier: a probabilistic object type classifier outputting probability of each object type given some subset of track features
- Time-sorted track list: active tracks sorted by time (with most recent appearing first), along with associated object type classifications
- Human operator: OBAC personnel who interact with the system.

The proposed solution updates the existing block diagram to include all blocks highlighted in red. New blocks include:

- Interest preferences: operators are able to include their object type and geographic preferences upon login
- Labeled tracks: operators are able to rate their interest in active tracks online
- Machine learning: a model that infers human track interest by considering their preferences, labeled tracks, track data, and object type classifications
- Interest-sorted tracks: a new track list sorted by predicted operator interest rather than time.

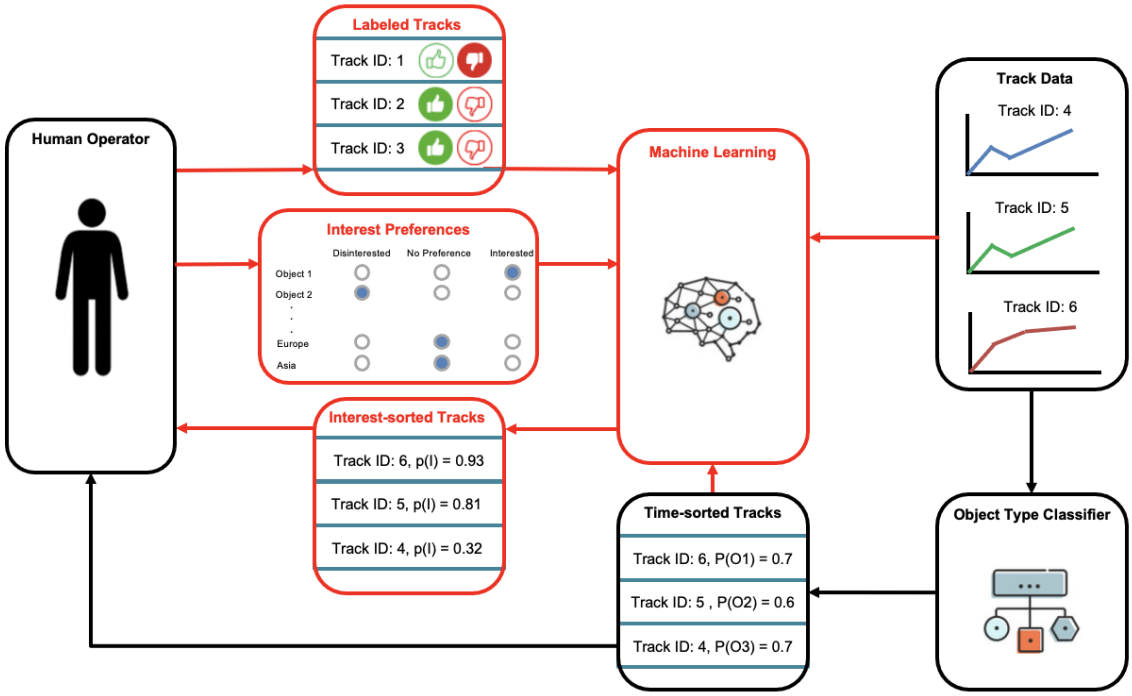
It is noted that for the purpose of this work, the "Object Type Classifier" block is considered to be a black box (to be discussed more thoroughly in Section 4). However, any probabilistic classifier is sufficient to insert in its place.

3.1.2 Track Data Details

Details regarding track data are discussed in this section. Active tracks have several properties that may be considered when attempting to learn whether they are interesting to a human operator. These are outlined below:

- Track features: measured or derived track characteristics such as altitude, intensity, and speed;

Figure 3.1: Block diagram model for the solution approach. All red blocks are new additions in an effort to learn human interest. The "Machine Learning" block is the primary focus of the remainder of this thesis.



- Object type: each track has some true classification;
- Geographic location: each track has some geographic information associated with it (this could be launch point, impact point, current location, etc.).

For the purpose of this work, we assume that track features and geographic location are fully observable. This is an approximation because these properties are actually derived from the statistical fusion of noisy dynamics and measurements (i.e. from a Kalman Filter), but assuming statistically consistent and accurate fusion, this is a minor assumption. True object type, however, is not observable. Instead, a probabilistic object type classifier gives some probability of the target being a particular object type given all known information. It should also be noted that track features, object type, and geographic location may not be independent of one another (for example, given a specific object type, it may be more likely that the track exists in one geographic location than another). We consider discrete geographic locations and object types:

$$\begin{aligned}\text{Geographic location} \in \{&\text{Africa, Asia, Caribbean, Central America,}\\ &\text{Europe, North America, Oceania, South America}\},\end{aligned} \tag{3.1}$$

$$\text{Object type} \in \{\text{Object 1, Object 2, Object 3, Object 4, Object 5, Object 6}\}. \tag{3.2}$$

The considered geographic locations are the eight regions defined by the Department of Homeland Security [26]. For the purposes of this work, synthetic, unclassified target types are considered and simply denoted as Objects 1-6. Representative altitude, intensity, and speed data for each object type is presented in Figures 3.2-3.4.

From these plots, it is seen that each object type has a relatively constant altitude profile, with each being at a different altitude. Objects 3 and 4 maintain a constant intensity throughout their history, while the remaining object types oscillate in a square waveform

Figure 3.2: Representative altitude profiles for each object type.

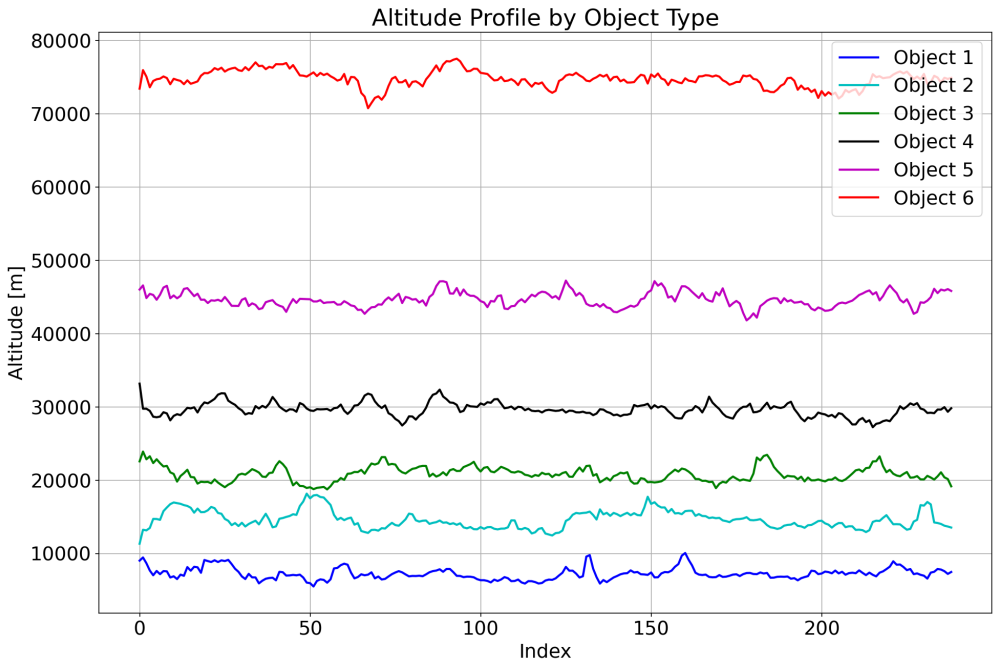
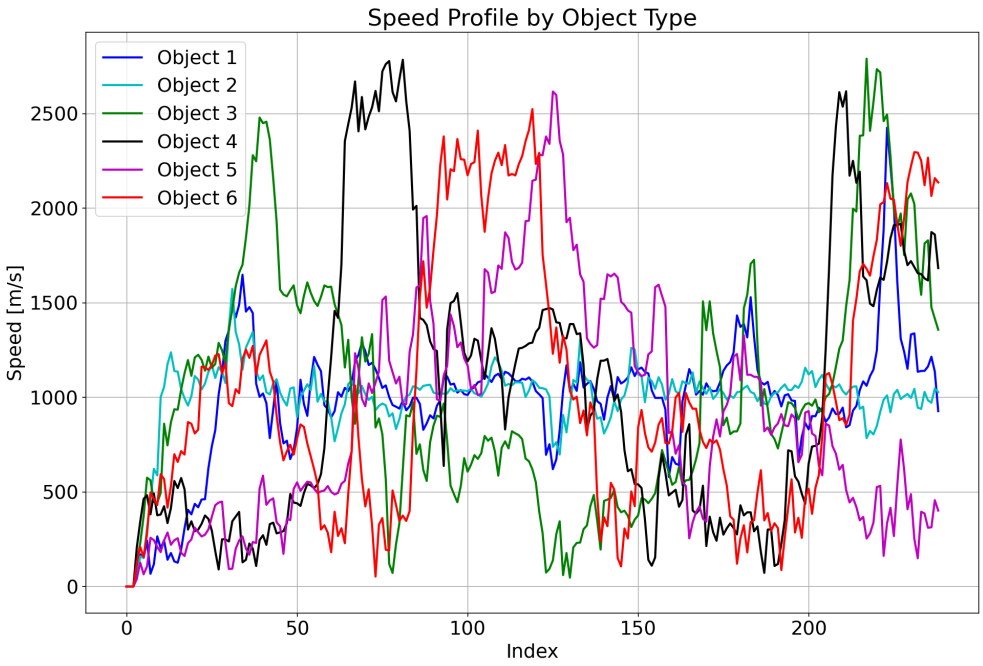


Figure 3.3: Representative intensity profiles for each object type.

Figure 3.4: Representative speed profiles for each object type.



pattern. Finally, Objects 1 and 2 maintain a near-constant speed (after an initial ramp-up), while the remaining object types oscillate in a sinusoidal manner.

3.1.3 Human Interaction Details

Human interaction with the system is introduced in this section. In this context, human operators are able to provide preferences for geographic locations and object types of interest upon login by ranking them as "Disinterested", "No Preference", or "Interested". They are also able to rank tracks as either "Interesting" or "Not Interesting" during live operations. Due to changing CONOPS and a limited pool of users, we use a worst case assumption that there is no training data available a priori (i.e. no a priori data available in the form of tracks labeled for interest). It is also noted that limited labeled data is expected to be received online from each user.

3.2 Modeling

Because of the lack of training data and limited ability to receive online feedback on tracks (it is not expected that a user will label a significant number of tracks during a session due to time constraints), it is necessary to incorporate prior information into the machine learning model. For this reason, probabilistic models like those introduced in Chapter 2 are a natural choice. Furthermore, discriminative models a more naturally implemented in this classification context than generative models. As such, the problem defined in Section 2.1 will be solved with a modified version of Bayesian logistic regression. The specific model is defined in more detail in following sections.

Using a modified Bayesian logistic regression requires a prior distribution to be placed on feature weights to initialize the preference inference algorithm. While uninformative priors can be used, the object type and geographic location preferences input by the user can serve prior knowledge to define more appropriate prior distributions. This approach also requires tracks to be labeled in order to generate conditional posteriors on the feature

weights. Given user labeling during live operations, this requirement is also met. For these reasons, the proposed model naturally fits within the problem space, and allows for interest prediction with limited training data.

The remainder of the modeling section will be split into two separate parts, each with its own slight variation of the same underlying probabilistic model. The first part will use the model to learn about human interest, while the second part will use the model to infer human interest given trained parameters. The two phases make use of the same underlying model, but are separated because random variable observability changes between the learning and inference phases, which will be discussed in more detail below.

3.2.1 Parameter Learning

This section discusses how priors for the weights within the modified Bayesian logistic regression model, $\vec{\theta}$ are defined, as well as how they are updated online for a specific user.

3.2.1.1 Graphical Model

The graphical model for parameter learning is shown in Figure 3.5. Note that while the network is presented with random variables $\vec{x}$, O, G, and I in a plate model, future derivations only consider a single index, i. This is done to simplify notation, and because tracks labeled for interest are given at discrete, separate times. This process is discussed in more detail in Section 3.3. Note that the point nodes represent non-random parameters. The nodes defined in this graphical model are:

- $\vec{x}_i$: Vector of track features at each time step (altitude, speed, and intensity) for track i;
- O_i: Object type for track i (as defined in 3.2);
- G_i: Current geographic location for track i (as defined in 3.1);
- I_i: Human interest for track i (0 for "Not Interested", 1 for "Interested");

- $\vec{\theta}$: Vector of feature weights;
- u_j^O: Human object type ranking for each object type j (-1 for "Disinterested", 0 for "No Preference", 1 for "Interested");
- u_k^G: Human geographic location ranking for each geographic location k (-1 for "Disinterested", 0 for "No Preference", 1 for "Interested").

The goal of this model is to calculate the conditional posterior $p(\vec{\theta} \,|\, I, \vec{x}, G)$. To achieve this goal, it is first necessary to define the joint distribution. This model, given its encoded conditional independencies, provides the following joint distribution for all random variables:

$$p(\vec{x}, O, G, I, \vec{\theta}) = p(\vec{x})p(O \mid \vec{x})p(G \mid O)p(I \mid \vec{x}, O, G, \vec{\theta})p(\vec{\theta}). \tag{3.3}$$

This distribution is important because it will be used in derivations in the following sections of this text. Individual distributions described in the joint are defined as follows:

- $p(\vec{x})$: undefined, drops out of derivation;
- $p(O \mid \vec{x})$: output of probabilistic object type classifier;
- $p(G \mid O)$: multinomial distribution;
  - Defined as uniform given lack of prior knowledge:

$$p(G = g \mid O = o) = \frac{1}{\#\text{ geographic locations}} \forall\ g,\ o \tag{3.4}$$

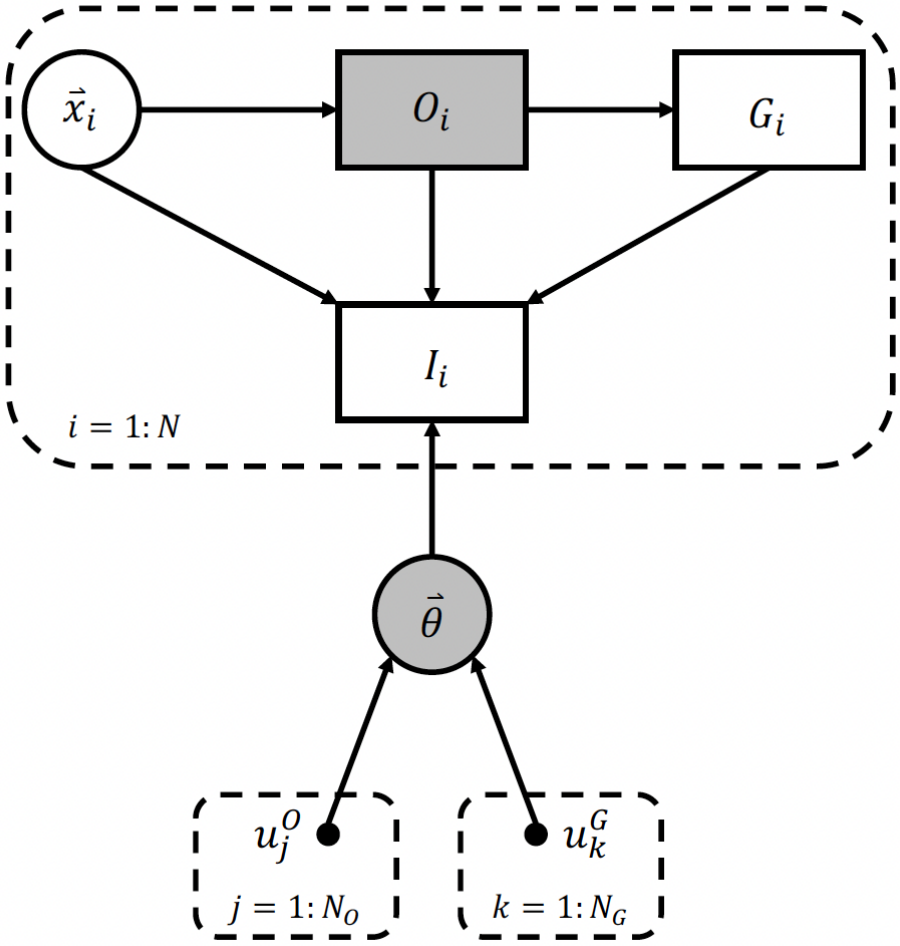
- $p(I \mid \vec{x}, O, G, \vec{\theta})$: logistic sigmoid function;
  - Defined as follows:

$$p(I = 1 \mid \vec{x}, O, G, \vec{\theta}) = \frac{1}{1 + \exp(-\vec{\theta}^T \vec{\mathcal{X}})}, \tag{3.5}$$

$$\vec{\mathcal{X}} = [\text{Intercept, One-hot } O, \text{ One-hot } G,\ f_1(\vec{x}),\ f_2(\vec{x}),\ f_3(\vec{x})], \tag{3.6}$$

$$f(\vec{x}) = \left[\frac{\text{max altitude}}{\alpha_1},\ \frac{\text{max intensity}}{\alpha_2},\ \frac{\text{max speed}}{\alpha_3}\right] \tag{3.7}$$

Figure 3.5: BN for learning feature weights. Priors for weights are determined by user preference inputs, and the object type is a partially observable input variable.



- $p(\vec{\theta})$: multivariate Gaussian
  - Defined as follows:

$$p(\vec{\theta}) = \mathcal{N}(\vec{\mu}_\theta, \Sigma_\theta), \tag{3.8}$$

$$\vec{\mu}_\theta = [w^I, w_1^O, ..., w_{N_O}^O, w_1^G, ..., w_{N_G}^G, w_1^{f(\vec{x})}, ..., w_n^{f(\vec{x})}]^T, \tag{3.9}$$

$$\Sigma_\theta = \text{diag}[\sigma^I, \sigma_1^O, ..., \sigma_{N_O}^O, \sigma_1^G, ..., \sigma_{N_G}^G, \sigma_1^{f(\vec{x})}, ..., \sigma_n^{f(\vec{x})}]. \tag{3.10}$$

In Equation 3.7, α_i represents the maximum value of feature i across all tracks. The scaling factors are included such that the continuous features exist on the same order of magnitude as the discrete features. This indicates that:

$$1 \geq f_i(\vec{x}) \; \forall i \in \{1, 2, 3\}. \tag{3.11}$$

The scaling factors were derived from all available data, and for this application are set as $\alpha_1 = 80764$, $\alpha_2 = 20800$, and $\alpha_3 = 4152$ (note that these factors are subject to change given new data).

The scaled track features are shown in Figures 3.6-3.8. These plots indicate that there is visual separation between object types based on these features. Note that features $f(\vec{x})$ are not exhaustive, and could include other features (i.e. maximum intensity rate, average altitude, etc.) to improve classification ability. However, in this case, only the three outlined features are considered.

3.2.1.2 Posterior Derivation

Given a track labeled with interest ($I = 1$) or lack thereof ($I = 0$), the goal of Bayesian parameter learning is to find the posterior distribution $p(\vec{\theta} \mid I, \vec{x}, G)$. By solving for this conditional posterior, the distribution of the weights can be updated in order to better predict interest in future tracks. The posterior is:

Figure 3.6: Scaled maximum altitude of all tracks, organized by object type.

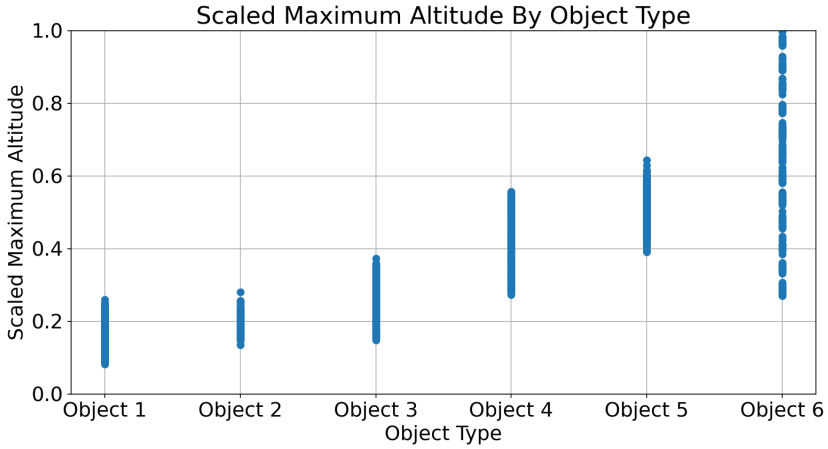


Figure 3.7: Scaled maximum intensity of all tracks, organized by object type.

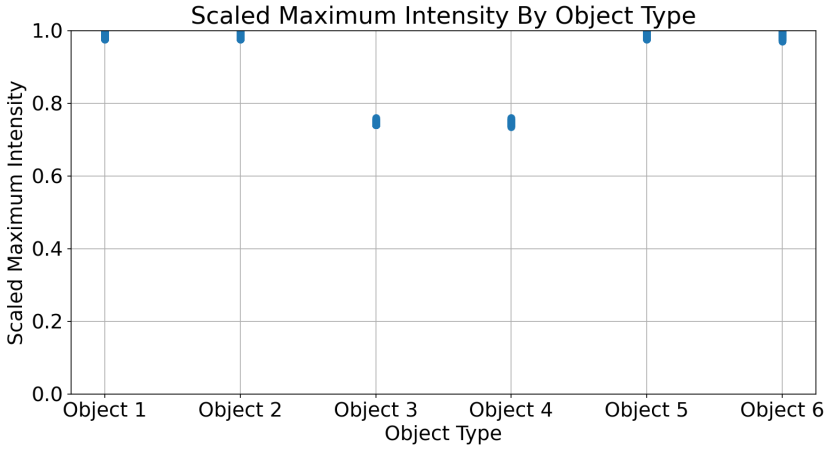
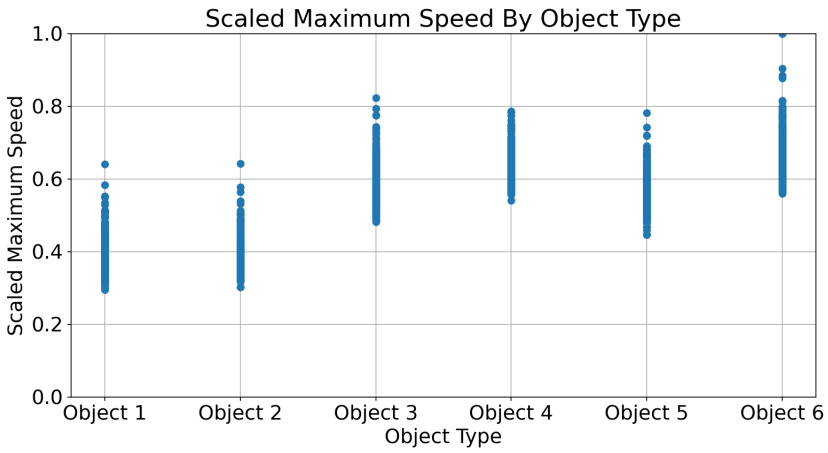


Figure 3.8: Scaled maximum speed of all tracks, organized by object type.



$$
\begin{aligned}
p(\vec{\theta} \mid I, \vec{x}, G) &= \frac{p(\vec{\theta}, \vec{x}, G, I)}{p(\vec{x}, G, I)} \\
&= \frac{\sum_O p(\vec{x}) p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta}) p(\vec{\theta})}{\sum_O \int p(\vec{x}) p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta}) p(\vec{\theta}) d\vec{\theta}} \\
&= \frac{p(\vec{x}) p(\vec{\theta}) \sum_O p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta})}{p(\vec{x}) \sum_O \int p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta}) p(\vec{\theta}) d\vec{\theta}} \\
&= \frac{p(\vec{\theta}) \sum_O p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta})}{\sum_O \int p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta}) p(\vec{\theta}) d\vec{\theta}} .
\end{aligned}
\tag{3.12}
$$

Conveniently, $p(\vec{x})$ drops out of the conditional posterior. This massively simplifies the problem because, in general, it would be very difficult to define the prior distribution over all track features (at least without significant offline training data).

The denominator in Equation 3.12 cannot be computed analytically because it involves taking the integral of the product of the logistic function with a Gaussian, and because $\vec{\theta}$ exists in high dimensional space, it is also intractable to compute with numerical integration. As such, approximation techniques are required to find the posterior. For this application, the Laplace approximation is employed as a means of defining the intractable posterior. This Laplace approximation and its application in this context are outlined below.

3.2.1.3 Laplace Approximation

The Laplace approximation is a technique used to fit a Gaussian posterior to an unnormalized continuous probability density function [6]. While many other techniques exist to approximate intractable posteriors, the Laplace approximation is a natural choice given its relative simplicity and computational speed. Its general theory and application to this specific problem are discussed here.

Theory Firstly, we denote some given information, D. Note that this may be a single value, or a vector of values. We can then say:

$$\begin{aligned} p(\theta \mid D) &= \frac{p(\theta, D)}{\int p(D, \theta) d\theta} \\ &= \frac{\exp(\log\left[p(D, \theta)\right])}{\int \exp(\log\left[p(D, \theta)\right]) d\theta} \ . \end{aligned} \tag{3.13}$$

The term $\log\left[p(D, \theta)\right]$ is then defined as the energy function $E(\theta)$, and is approximated using a second-order Taylor series expansion:

$$E(\theta) \approx E(\theta_0) + (\theta - \theta_0)^T \nabla E(\theta_0) + \frac{1}{2}(\theta - \theta_0)^T \nabla^2 E(\theta_0)(\theta - \theta_0). \tag{3.14}$$

The term θ_0 is chosen as the MAP estimate of θ (called θ^*) where $\nabla E(\theta^*) = 0$, such that:

$$E(\theta) \approx E(\theta^*) + \frac{1}{2}(\theta - \theta^*)^T \nabla^2 E(\theta^*)(\theta - \theta^*). \tag{3.15}$$

Using the above definition, Equation 3.13 can be approximated by the second order Taylor series expansion evaluated at θ^*:

$$\begin{aligned} \frac{\exp[\log p(D, \theta)]}{\int \exp[\log p(D, \theta)] d\theta} &\approx \frac{\exp[E(\theta^*) + \frac{1}{2}(\theta - \theta^*)^T \nabla^2 E(\theta^*)(\theta - \theta^*)]}{\int \exp[E(\theta^*) + \frac{1}{2}(\theta - \theta^*)^T \nabla^2 E(\theta^*)(\theta - \theta^*)] d\theta} \\ &\approx \frac{\exp[-\frac{1}{2}(\theta - \theta^*)^T \{-\nabla^2 E(\theta^*)\}(\theta - \theta^*)]}{\int \exp[-\frac{1}{2}(\theta - \theta^*)^T \{-\nabla^2 E(\theta^*)\}(\theta - \theta^*)] d\theta} \ . \end{aligned} \tag{3.16}$$

Due to the form presented in Equation 3.16, the Laplace approximation for the posterior distribution of θ is given by:

$$p(\theta \mid D) \approx \mathcal{N}(\theta^*, H^{-1}). \tag{3.17}$$

Note that in Equation 3.17, θ^* is approximated by maximizing the energy function $E(\theta)$ over all possible θ. This is generally accomplished numerically by using gradient ascent (or gradient descent, if we take $-E(\theta)$ to be the energy function). Also, H is the Hessian matrix that is defined as:

$$H = -\nabla^2 \log E(\theta^*)$$

$$\text{or, element-wise:} \tag{3.18}$$

$$H_{ij} = \left.\frac{\partial^2 E(\theta)}{\partial\theta_i \partial\theta_j}\right|_{\theta=\theta^*}.$$

The Laplace approximation essentially fits a Gaussian at the mode of the posterior distribution whose curvature matches that of the posterior. While this derivation focused on the form shown in Equation 3.13, the mathematics are valid when the numerator can be expressed as the energy function $E(\theta)$.

Application Given the intractable denominator in Equation 3.12, and because the denominator has no dependence on $\vec{\theta}$, we can say:

$$p(\vec{\theta} \mid I, \vec{x}, G) \propto p(\vec{\theta}) \sum_O p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta}). \tag{3.19}$$

Given the above definition, energy is defined as:

$$\begin{aligned} E(\vec{\theta}) &= -\log p(\vec{\theta}) \sum_O p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta}) \\ &= -\log p(\vec{\theta}) - \log \sum_O p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta}). \end{aligned} \tag{3.20}$$

With this equation, and using the definitions of the priors for each individual distribution, $\vec{\theta}^*$ can be calculated numerically using gradient descent. Note that if energy is not strictly log-concave, gradient descent may converge to a *local* minima rather than a *global* minimum. As such, it may be best practice to minimize the function multiple times with different initial conditions, and choose $\vec{\theta}^*$ as the best result.

The Hessian can be defined numerically using four-point central differences [27] to calculate second derivatives at $\vec{\theta}^*$. The elements are:

$$H_{ii} = \frac{E(\vec{\theta}^* + \Delta\vec{\theta}_1) - 2E(\vec{\theta}^*) + E(\vec{\theta}^* + \Delta\vec{\theta}_2)}{h^2}, \tag{3.21}$$

$$H_{ij} = \frac{E(\vec{\theta}^* + \Delta\vec{\theta}_3) - E(\vec{\theta}^* + \Delta\vec{\theta}_4) - E(\vec{\theta}^* + \Delta\vec{\theta}_5) + E(\vec{\theta}^* + \Delta\vec{\theta}_6)}{4h^2}\ . \tag{3.22}$$

In Equations 3.21 and 3.22, the $\Delta\vec{\theta}$ terms represent small perturbations to the optimal weight estimate, as outlined below:

- $\Delta\vec{\theta}_1$: zero vector with i^{th} element equal to $+h$;
- $\Delta\vec{\theta}_2$: zero vector with i^{th} element equal to $-h$;
- $\Delta\vec{\theta}_3$: zero vector with i^{th} and j^{th} elements equal to $+h$;
- $\Delta\vec{\theta}_4$: zero vector with i^{th} element equal to $+h$ and j^{th} element equal to $-h$;
- $\Delta\vec{\theta}_5$: zero vector with i^{th} element equal to $-h$ and j^{th} element equal to $+h$;
- $\Delta\vec{\theta}_6$: zero vector with i^{th} and j^{th} elements equal to $-h$.

Note that in the above, h is a small perturbation term whose value should be set small enough to approximate the derivatives appropriately. In the case of this application, the perturbation term was set heuristically as $h = 1 \times 10^{-5}$. Given the approximate optimal weighting and Hessian, the posterior distribution of $\vec{\theta}$ is defined as:

$$p(\vec{\theta} \mid I, \vec{x}, G) \sim \mathcal{N}(\vec{\theta}^*, H^{-1}). \tag{3.23}$$

3.2.2 Interest Inference

Interest inference for a specific track/user combination is discussed in this section. Note that this is examined separately from Parameter Learning because the goal is to predict interest rather than to learn the weights (i.e. no learning takes place based on the derivations in this section).

3.2.2.1 Graphical Model

The graphical model for inference is shown in Figure 3.9. The notation here is the same as the above model for parameter estimation. The two changes between the models are 1) the weights are observed, and 2) interest is no longer observed. The weights, $\vec{\theta}$ are assumed as the maximum a posteriori (MAP) estimate of the distribution $p(\vec{\theta} \mid I_{1:N}, \vec{x}_{1:N}, G_{1:N})$ such that $\vec{\theta} = E[\vec{\theta}]$ (after N labeled tracks). This assumption is made to simplify inference and allows for an analytical solution.

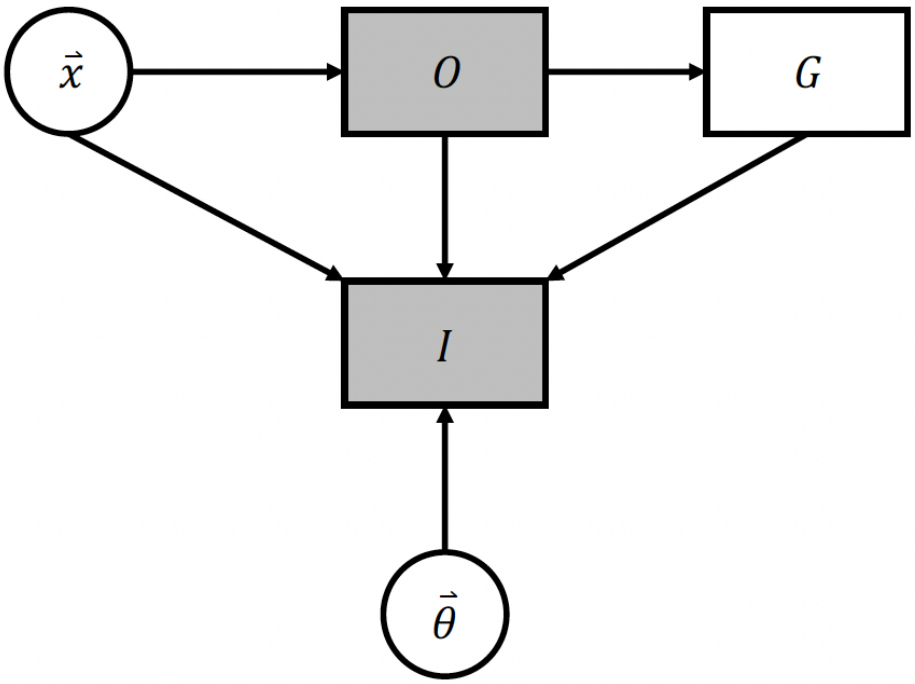
3.2.2.2 Probability of Interest Derivation

The goal of this model is to infer interest via the conditional distribution $p(I \mid \vec{x}, G, \vec{\theta})$. The joint distribution is the same as was derived for parameter estimation (Equation 3.3). Given the joint, probability of interest for a single, unlabeled, track can be determined as follows:

$$\begin{aligned} p(I \mid \vec{x}, G, \vec{\theta}) &= \frac{p(I, \vec{x}, G, \vec{\theta})}{p(\vec{x}, G, \vec{\theta})} \\ &= \frac{\sum_O p(\vec{x}) p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta}) p(\vec{\theta})}{\sum_O \sum_I p(\vec{x}) p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta}) p(\vec{\theta})} \\ &= \frac{p(\vec{x}) p(\vec{\theta}) \sum_O p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta})}{p(\vec{x}) p(\vec{\theta}) \sum_O \sum_I p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta})} \\ &= \frac{\sum_O p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta})}{\sum_O \sum_I p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta})} \,. \end{aligned} \tag{3.24}$$

Because the denominator is not a function of I, and due to the relationship $p(I = 1 \mid \vec{x}, G, \vec{\theta}) + p(I = 0 \mid \vec{x}, G, \vec{\theta}) = 1$, Equation 3.24 can be expressed up to a normalizing constant, and normalized after calculating both $p(I = 1 \mid \vec{x}, G, \vec{\theta})$ and $p(I = 0 \mid \vec{x}, G, \vec{\theta})$ for a single track:

Figure 3.9: BN for predicting interest. Note that weights are assumed to be observable in this case, and are taken as $E[\vec{\theta}]$ with respect to Equation 3.23, as generated by observed interest labels up to a certain point.



$$p(I \mid \vec{x}, G, \vec{\theta}) \propto \sum_{O} p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta}), \tag{3.25}$$

$$p(I = 1 \mid \vec{x}, G, \vec{\theta}) = \frac{\sum_{O} p(O \mid \vec{x}) p(G \mid O) p(I = 1 \mid \vec{x}, O, G, \vec{\theta})}{\sum_{O} \sum_{I} p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta})}, \tag{3.26}$$

$$p(I = 0 \mid \vec{x}, G, \vec{\theta}) = \frac{\sum_{O} p(O \mid \vec{x}) p(G \mid O) p(I = 0 \mid \vec{x}, O, G, \vec{\theta})}{\sum_{O} \sum_{I} p(O \mid \vec{x}) p(G \mid O) p(I \mid \vec{x}, O, G, \vec{\theta})}. \tag{3.27}$$

Equations 3.26 and 3.27 allow for interest prediction for a single track/operator combination. These conditional distributions take into account all information that is given or was learned based on the procedures outlined in Section 3.2.1.

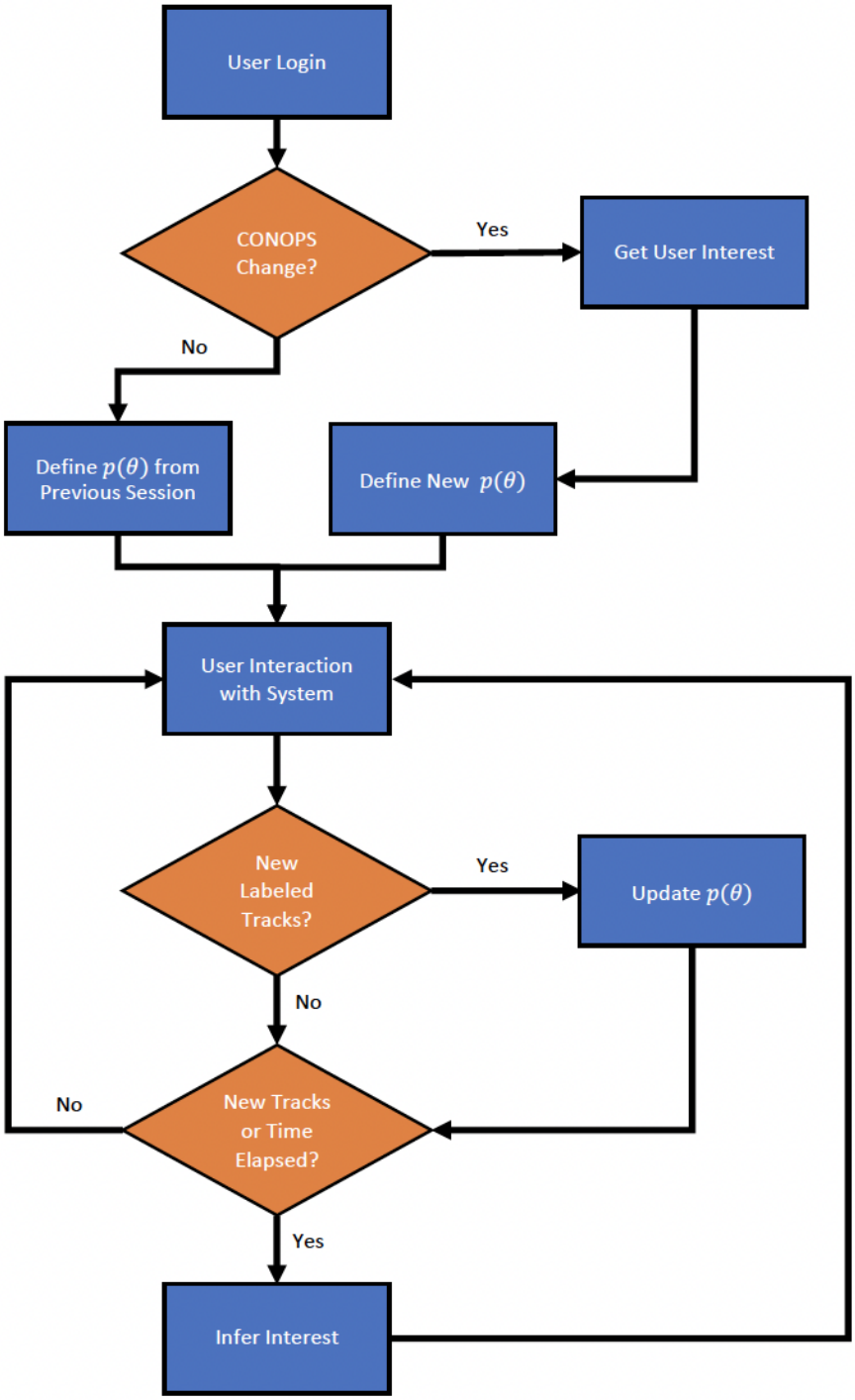
3.3 System-level Workflow

The entire system-level flowchart is presented in Figure 3.10. Upon login, the distribution $p(\vec{\theta})$ is either defined based on user inputs (if the CONOPS changed) or initialized with the posterior from the previous session (if the CONOPS remained constant). The user then interacts with the system normally. If there are new labeled tracks, $p(\vec{\theta})$ is updated using Equation 3.23. If there are new unlabeled tracks in the active track list, or a specified amount of time has elapsed, probability of interest is recalculated for each active track. This loop is evaluated in perpetuity until the operator logs off or changes CONOPS. Refer to Appendix A for an algorithmic view of the workflow and how user inputs are used to initialize and train the model.

3.4 Model Limitations

Three significant limitations of this model are that the activation function for predicting human interest is defined as a logistic sigmoid, the Laplace approximation is used to determine posterior weight distributions, and a MAP estimate of $\vec{\theta}$ is assumed when inferring interest. These limitations are discussed below.

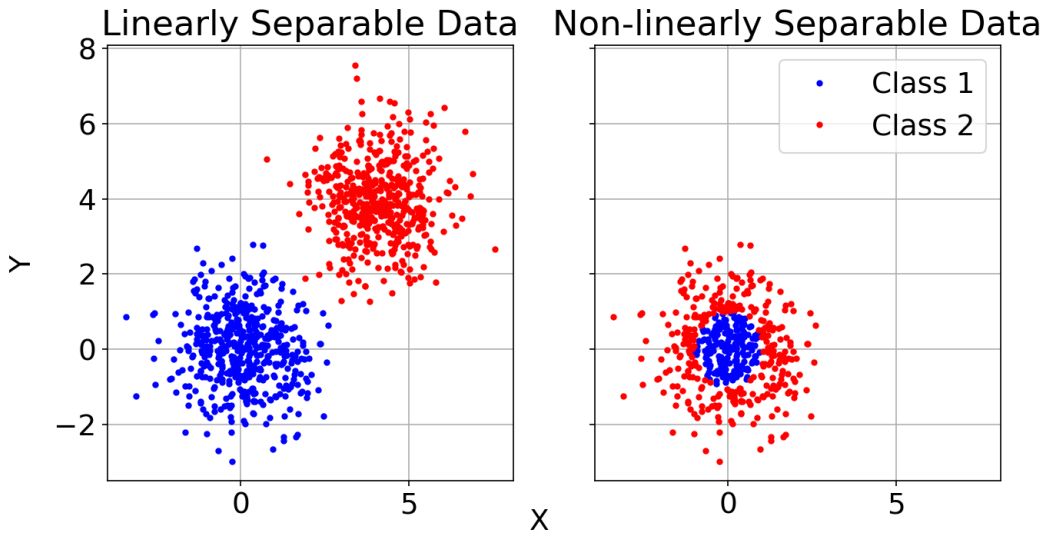
Figure 3.10: Full system-level flowchart for interest learning and inference.



3.4.1 Logistic Sigmoid

While the probability response defined by a logistic sigmoid is nonlinear, it is nominally linear relative to the input features. This is apparent given the exponential term in the denominator of the logistic sigmoid (Equation 3.5), whose argument is a linear combination of feature weights and features themselves. As such, this activation function is only useful as a classifier if the labels are linearly separable in feature space. The distinction between linearly-separable and non-linearly separable data is shown in Figure 3.11. In essence, linearly separable data means classes can be separated by a straight line, whereas non-linearly separable data cannot.

Figure 3.11: Linearly separable versus non-linearly separable data. In the left plot, a line could be drawn to separate the two classes, while this is not possible in the right plot.



Figures 3.12 and 3.13 consider the case of binary logistic regression for a single input variable (for ease of visualization). At a high level, non-Bayesian logistic regression aims to find the Maximum Likelihood Estimate (MLE) of the sigmoid parameters (as opposed to the MAP estimate for the Bayesian case) using a gradient-based technique. However, the specifics are not particularly important for this analysis.

Figure 3.12: Logistic regression results for a single input variable. Note that the data points are linearly separable (i.e. a single straight line could be drawn that separates the two categories).

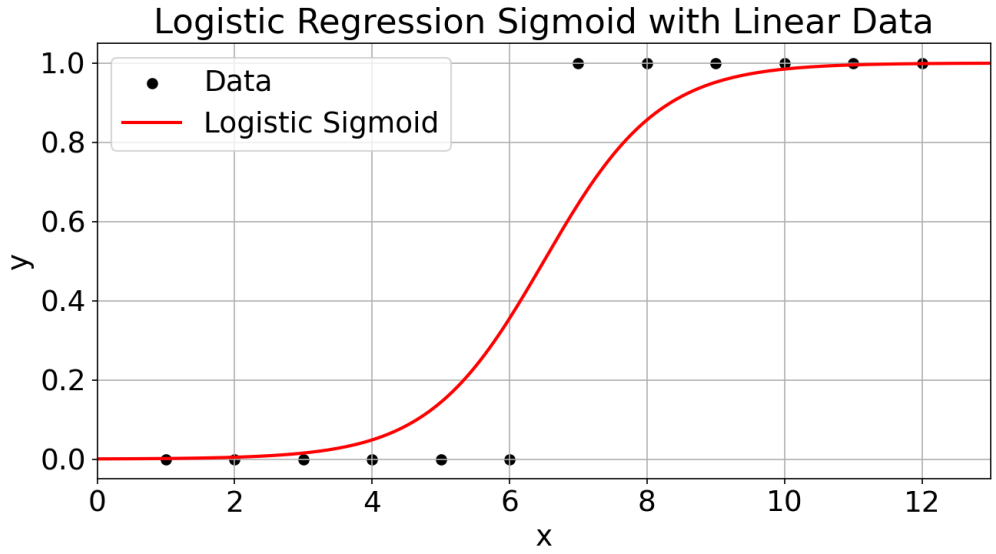


Figure 3.13: Logistic regression results for a single input variable. Note that the data points are not linearly separable(i.e. a single straight line could not be drawn that separates the two categories).

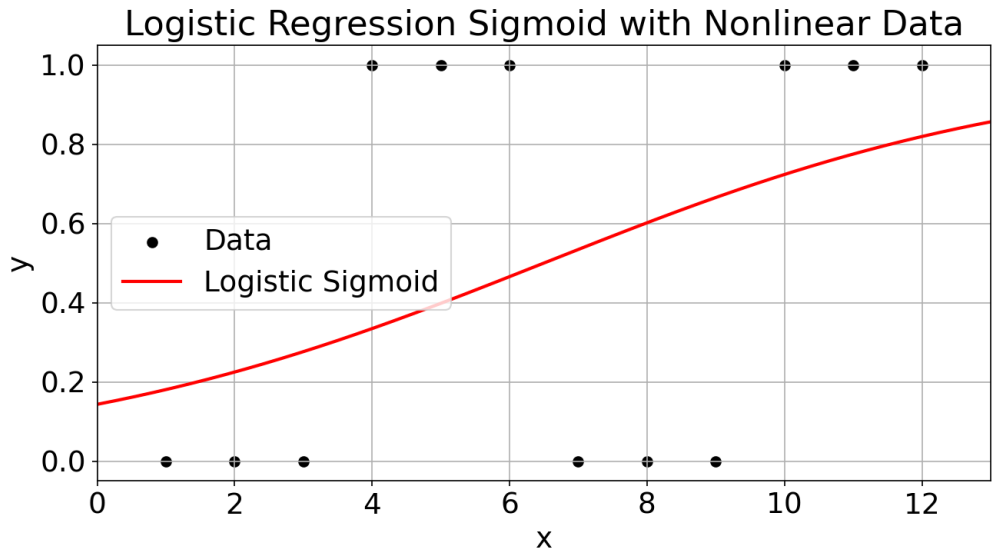


Figure 3.12 shows the case of a logistic regression being performed on linearly separable data. Visually, the classifier behaves desirably. Setting a classification probability threshold of 0.5 (where all points greater than probability 0.5 are classified as $y = 1$ and all point below are classified as $y = 0$), all data points are correctly classified by the model. Conversely, Figure 3.13 shows a logistic regression being performed on non-linearly separable data. This fit is clearly poor from a visual standpoint. From the labeled data, we can see that a multimodal function would act as a more natural classifier in this case. Again setting a classification threshold of 0.5, only half of the data is correctly classified by the model.

In the context of this work, this activation function will tend to struggle with logical operators defining a user's interest. For example, the logistic sigmoid function may not be appropriate for a user who is interested in objects whose maximum altitude is greater than 10 km *or* less than 1 km and disinterested otherwise. Despite this limitation, it is assumed that the object type and geographic location weights will tend to compensate for non-linearly separable features, and the logistic sigmoid is used because it neatly maps probabilities between 0 and 1.

3.4.2 Laplace Approximation

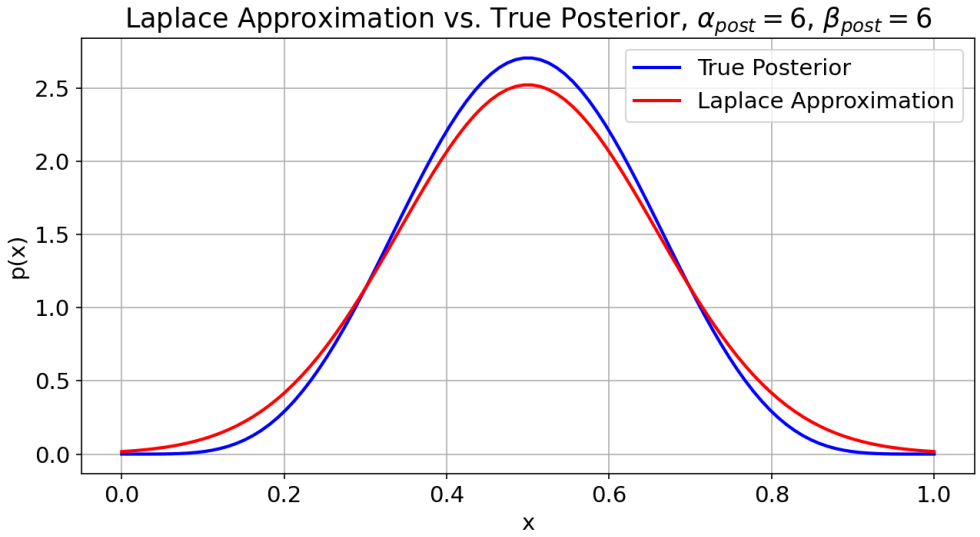
The Laplace approximation is fundamentally limited in the fact that it approximates the posterior as a Gaussian. This assumption is violated if the distribution is highly non-Gaussian, and in particular, if the true target distribution is multimodal. Assuming the posterior is not perfectly Gaussian, the approximation tends to become less accurate further from its mode.

Below, we consider the accuracy of the Laplace approximation in estimating the posterior of a Beta distribution. A conjugate prior was used (as in Section 2.3.1.3), such that the exact posterior could be calculated in closed form. Figure 3.14 shows the case where 5 positive and 5 negative observations were given, resulting in a posterior distribution defined by $\text{Beta}(6, 6)$. Here, because the true posterior is nearly Gaussian, the Laplace approximation

performs favorably.

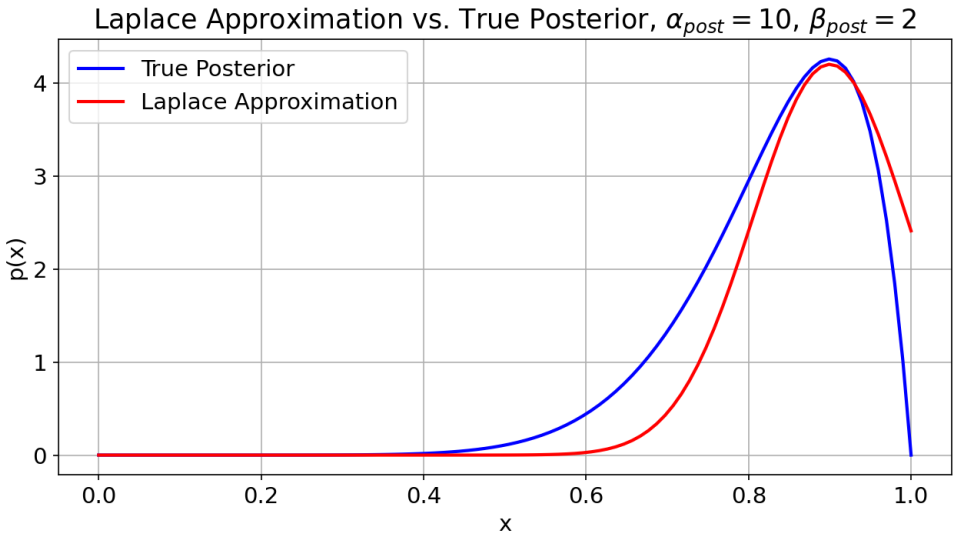
Figure 3.15 shows the case where 9 positive and 1 negative observations were given, resulting in a posterior distribution defined by Beta$(10, 2)$. The true posterior in this case is highly non-Gaussian. Because the Laplace approximation assumes a Gaussian posterior, the estimated posterior diverges greatly from truth as the value of x moves further from the mode of the true posterior. This phenomena is exasperated in the case of multimodal posterior distributions.

Figure 3.14: Laplace approximation compared to exact posterior for a Beta distribution. This fit is strong because the true posterior is nearly Gaussian.



There are many approximate inference techniques that could be employed to more accurately estimate posterior weight distributions. Adaptive Rejection Sampling [16] has been used to approximate univariate log-concave density functions with good results, while Adaptive Rejection Metropolis Sampling [15] does the same while removing the log-concavity assumption. Markov Chain Monte Carlo methods such as Gibbs sampling [8] have been used to approximate multimodal posteriors using empirical samples [3]. Despite the favorable properties of these methods, they generally take longer to converge than the Laplace

Figure 3.15: Laplace approximation compared to exact posterior for a Beta distribution. This fit is weak because the true posterior diverges significantly from Gaussian.



approximation, particularly in high-dimensional space. This context requires the model to operate in near real-time in order to not interrupt the existing workflow. In the interest of allowing for real-time posterior updates, the Gaussian estimate produced by the Laplace approximation is a logical technique choice.

3.4.3 MAP Estimate of Feature Weights

As stated in Section 3.2.2, a MAP estimate of the track feature weights $\vec{\theta}$ is assumed when inferring user interest in unlabeled tracks. This allows for the conditional distributions $p(I = 1 \mid \vec{x}, G, \vec{\theta})$ and $p(I = 0 \mid \vec{x}, G, \vec{\theta})$ (as defined in Equations 3.26 and 3.27) to be computed analytically without the need to integrate over all possible $\vec{\theta}$. We note that many techniques could be employed to approximate these conditional distributions without assuming a MAP estimate, including the Laplace approximation, or other methods discussed in Section 3.4.2. However, because $p(\vec{\theta})$ is already approximated as a Gaussian (meaning it is assumed symmetric and unimodal), this approximation is reasonable. Because user interest in unlabeled tracks is computed far more often than the posterior distribution of the weights (where the Laplace approximation is used), even relatively fast approximation techniques are infeasible given computation time constraints. Therefore taking the MAP estimate of $\vec{\theta}$ is vital for meeting the real-time requirements of the system.

Chapter 4

Results

This chapter validates the functionality of the model proposed in Chapter 3 in several possible settings:

(1) An ideal scenario where the operator provides accurate prior information and the probabilistic classifier perfectly classifies the correct object type (4.1);

(2) A scenario with a poor object type classifier that does not reliably and confidently predict the correct object type (Section 4.2);

(3) A scenario where the simulated human operator provides poor initial inputs where their actual interests do not match the given inputs (Section 4.3);

(4) A realistic scenario where ideal assumptions about operator responsiveness are relaxed (Section 4.5).

Due to lack of access to true operator inputs, all results were obtained via truth model simulation. In all test cases outlined below, the true user interest remains constant. When labeling tracks for interest, the user is always labels consistently based on their true interest. Truth vectors are outlined below:

$$u_T^O = [1, -1, -1, -1, -1, -1], \tag{4.1}$$

$$u_T^G = [0, 0, 0, 0, 0, 0, 0, 0]. \tag{4.2}$$

These true interest vectors indicate that the user is interested in Object 1, and disinterested in all other object types. They have no preference for geographic location. As such, this means that in all test cases, the user will always label true Object 1 tracks as interesting regardless of geographic location, and all other tracks as not interesting.

In this chapter, we will adopt the term "training epoch" to represent the labeling of a single track of each object type (i.e. one epoch means the user labels one of each Object 1-6 as interesting or disinteresting). This terminology is introduced as a means of directly and consistently comparing different test cases.

Sections 4.1-4.4 compare the performance in challenging edge cases to the performance in an ideal setting. Section 4.5 introduces a deeper dive into a more realistic test case. Finally, closing remarks are offered in Section 4.6.

4.1 Ideal Case

This section presents an ideal case, where the following ideal assumptions are made: the user inputs are correct relative to their true interest model; a perfect probabilistic classifier exists (that predicts the true object type with probability 1); and the user's interest labeling is consistent. During simulation, their track labeling is always consistent with their true interest (as outlined in Equations 4.1 and 4.2), which in this case matches the given inputs.

In general, results will be presented after 20 training epochs. This number was chosen because it represents a middle-ground of allowing the model to learn user interest, while also taking into account that it is unrealistic to expect a busy human operator to label a large number of tracks. The general methodology for obtaining results is:

(1) Complete 1 full training epoch;

(2) Score 20 tracks of each object type for interest probability (where the same 20 test tracks are used after each epoch);

(3) Repeat steps 1 and 2 for 20 training epochs.

The probability of interest for 20 test tracks of each object type is shown in Figure 4.1 (after $p(\vec{\theta})$ was updated given 20 training epochs). Across all tested tracks, Object 1 tracks had a higher calculated interest than tracks of any other object type, and $< 10\%$ for all other object types. This indicates that the classifier is functioning as intended in an ideal case.

In an ideal scenario, the model would be able to rank Object 1 tracks as more interesting than other object types immediately (i.e. without any labeled tracks for training). As such, it is important to understand the model's predictive ability over time. This behavior is shown in Figure 4.2. Here, the probability of interest averaged over 20 tracks of each object type is shown at each training epoch. "Epoch 0" corresponds to the model inferring interest with only user inputs u^O and u^G and no training data.

This plot indicates that, given appropriate user inputs u^O and u^G, the model is immediately able to rank interest in Object 1 over all other object types. Object 1 suffers an initial downward slope in the first epoch before increasing in interest probability across all subsequent epochs. All other object types steadily decrease in interest probability.

4.2 Poor Object Type Classifier

Section 4.1 addresses the behavior of the model in an ideal case. In practice, an ideal case will never exist. This section aims to characterize the model's ability to determine interesting tracks when a probabilistic classifier does not predict the correct object type with probability 1. The same user inputs (Equations 4.1 and 4.2) and interest labeling methodology are used here as were used before. Results were obtained using using the test methodology outlined in 4.1.

4.2.1 Varying True Object Type Classification Probability

This section investigates the impact of varying confidence in true object typing, with the remainder of the probability spread evenly across all other objects. In this section, we will define the probability of an object type given track features (i.e. the direct output from

Figure 4.1: Probability of human interest over 20 tracks of each object type, after 20 training epochs in the ideal scenario. The model accurately predicts high probability of interest for Object 1 and low probability of interest for the rest.

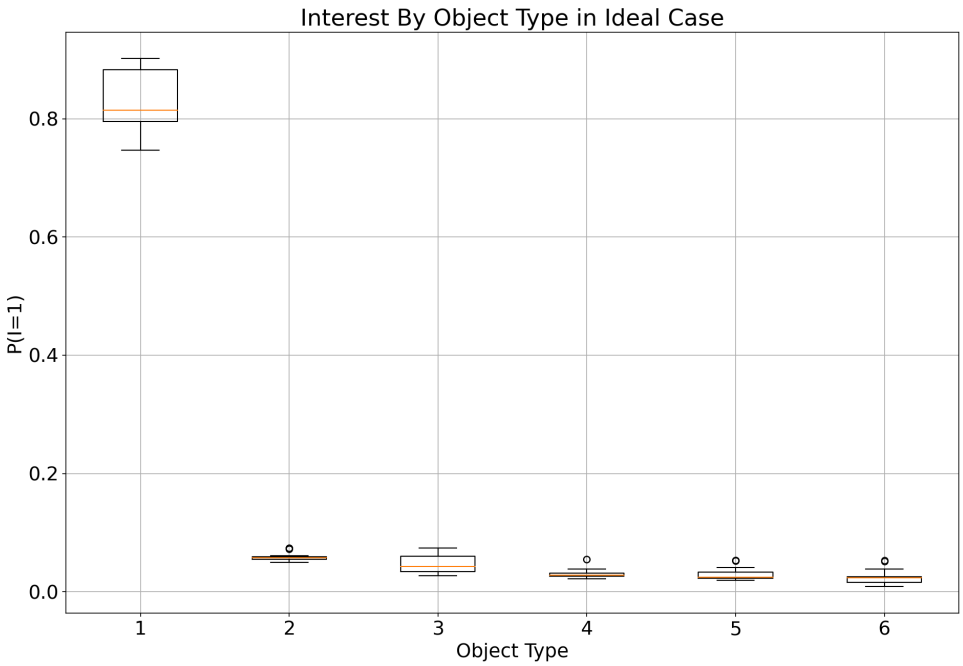
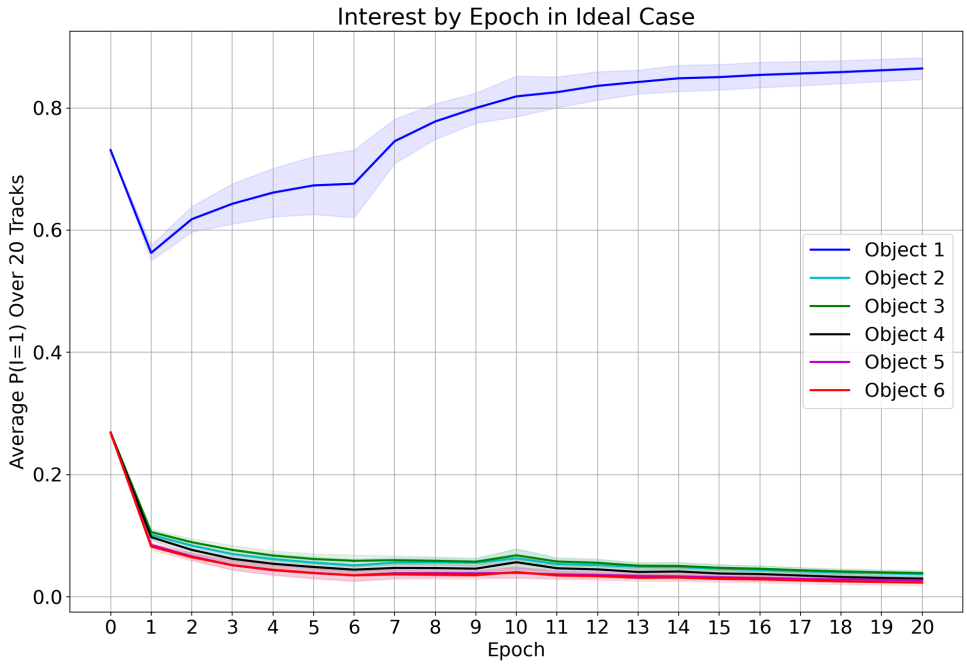


Figure 4.2: Average interest probability and 95% confidence bounds by epoch for each object type in ideal case.



a probabilistic classifier) as:

$$p(O = o \mid \vec{x}) = \begin{cases} \gamma & \text{if } o = \text{true object type} \\ \frac{1-\gamma}{\#\text{ objects}-1} & \text{otherwise} \end{cases} \tag{4.3}$$

Firstly, it should be noted that $\gamma = 0.167$ is a critical point because at this value, all object types are predicted with the same probability. Therefore, at this point, the classifier is comparable to a uniform classifier. The probability of interest for 20 test tracks of each object type for $\gamma = 0.167$ is shown in Figure 4.3 (after $p(\vec{\theta})$ was updated given 20 training epochs).

This plot, when compared to Figure 4.1, shows the interest classification ability in the presence of a perfect vs. uniform object type classifier in otherwise constant conditions. Despite the degraded performance in predicting Object 1 interest, its average interest remains significantly higher than the remaining object types. However, unlike in the ideal case, there is overlap between the Object 1 box plot and box plots for other object types. This indicates that the model is robust to the case of random classification in general, but may not accurately sort every track.

Epoch learning is shown for $\gamma = 0.167$ in Figure 4.4. The learning shown here is in contrast to that exhibited in the ideal case (Figure 4.2). Firstly, at epoch 0, interest probability is nearly identical for all object types. This is because weight priors only take into account object type information, for which a uniform object type classifier provides none. Initially, learning decreases average predicted interest in all object types before all reach a maximum at epoch 10. From epochs 10 to 20, Objects 1 and 2 level off while the remaining object types decrease in predicted interest. Object 2 does not continue to decrease because, as shown in Figures 3.6-3.8, it has relatively similar derived features in comparison to Object 1. Despite the degraded performance relative to the ideal case, at all epochs (other than epoch 0), Object 1 has the highest average predicted probability of interest.

Interest classification as a function of γ is presented in Table 4.1. This table shows the

Figure 4.3: Probability of human interest over 20 tracks of each object type, after 20 training epochs with $\gamma = 0.167$ (uniform object classification). The model, in general, predicts higher probability of interest for Object 1 than the rest.

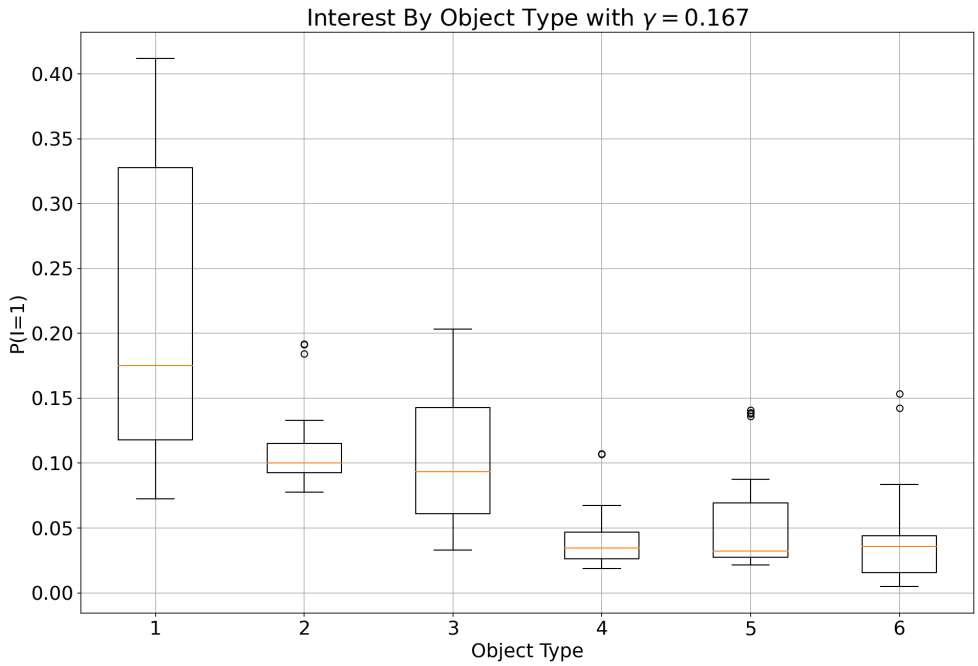
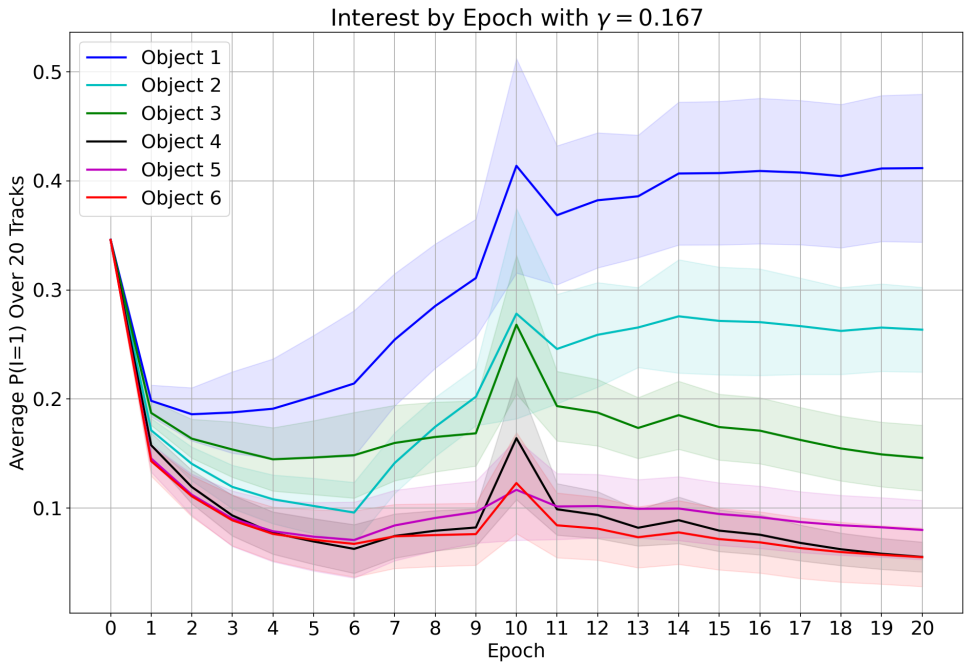


Figure 4.4: Average interest probability and 95% confidence bounds by epoch for each object type with $\gamma = 0.167$ (uniform object classification).



average (over 20 tracks) $p(I = 1 \mid \vec{x}, G, \vec{\theta})$ for each object type. These results are after 20 training epochs for each case.

This table indicates that the interest classifier's ability to mark Object 1 as interesting decreases as probabilistic object type classifier performance decreases. This behavior is expected, particularly because the separation in features $f(\vec{x})$ between objects is the only means of predicting interest in this case. However, because the intention of this work is to rank tracks in order of relative interest, the model performs well regardless of classifier accuracy.

Between $\gamma = 0.2$ and $\gamma = 0.1$ is a critical point where the true object type is considered less likely than all other object types by the probabilistic object type classifier. Thus, the probabilities associated with $\gamma = 0.1$ marks the first point where the object type weights would potentially lower the probability of interests in Object 1. However, even with very poor object typing, the model is able to distinguish Object 1 as being most interesting.

4.2.2 High Confidence in Incorrect Object Type

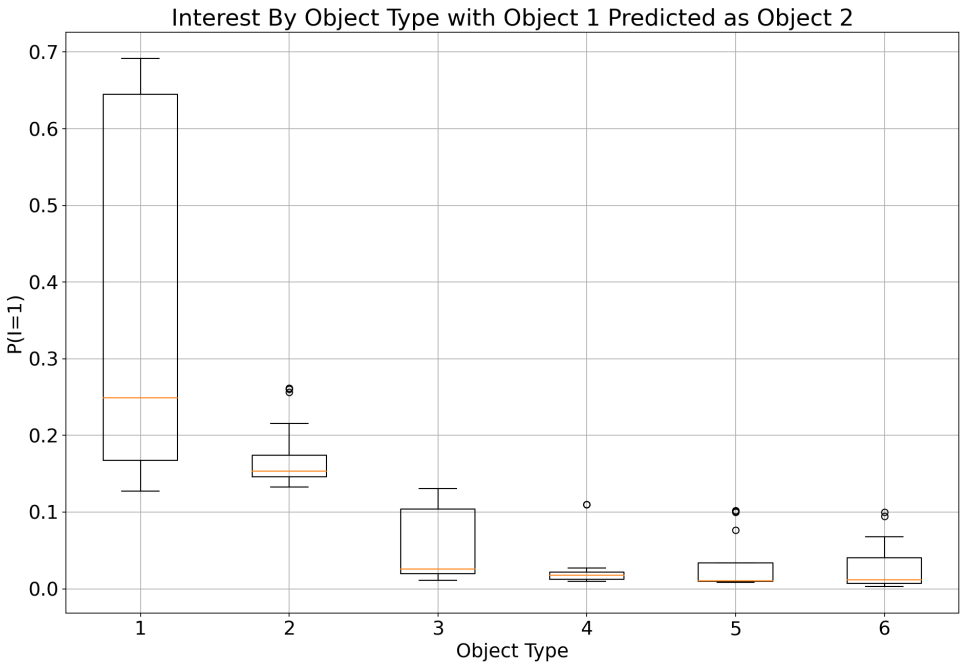
A scenario that is also important to understand is the case where the object type classifier confidently types an object incorrectly (e.g. it predicts Object 2 with 90% probability when the true type is Object 1). Here, we consider a user with the same preferences as before, and an object type classifier that predicts actual Object 2-6 tracks correctly with probability 1. However, the classifier predicts actual Object 1 tracks as another object type with probability 1. The probability of interest for 20 test tracks of each object type is shown in Figure 4.5. This prediction is after $p(\vec{\theta})$ was updated given 20 training epochs and with Object 1 being classified as Object 2 with probability 1 (this is generally the most challenging case, as discussed previously).

As with Figure 4.3, the model has more difficulty separating interest between objects than in the ideal case. This is expected because object type cannot be used as a predictive feature given the poor classifier. However, even in this challenging case, Object 1 is correctly

Table 4.1: Predicted interest probability for each object type averaged over 20 tracks. The posterior weight distributions are after 20 training epochs, given a classifier parameterized by γ, and a human that always labels true Object 1 as interesting and everything else as not interesting. Note that cells are color-coded such that green represents high values and red represents low values.

$p(I = 1 \mid \vec{x}, G, \vec{\theta})$						
γ	True Object Type					
	Object 1	Object 2	Object 3	Object 4	Object 5	Object 6
0.9	0.695	0.083	0.07	0.045	0.045	0.036
0.8	0.538	0.098	0.082	0.047	0.05	0.04
0.7	0.402	0.105	0.087	0.045	0.051	0.04
0.6	0.308	0.108	0.09	0.043	0.052	0.04
0.5	0.25	0.11	0.093	0.043	0.052	0.041
0.4	0.217	0.111	0.097	0.043	0.053	0.042
0.3	0.204	0.111	0.1	0.043	0.054	0.042
0.2	0.207	0.113	0.103	0.043	0.056	0.043
0.1	0.217	0.115	0.106	0.043	0.057	0.044

Figure 4.5: Probability of human interest over 20 tracks of each object type, after 20 training epochs with Object 1 predicted as Object 2 with probability 1. The model, in general, accurately predicts higher probability of interest for Object 1 than the rest.



predicted to have higher average interest than all other object types. Again, there is some overlap between the box plot for Object 1 and the box plots for other objects, specifically with Object 2.

Epoch learning is shown for the case where Object 1 is classified as Object 2 in Figure 4.6. While each object type has a similar average interest probability at epoch 0, after a single training epoch, and for all subsequent epochs, Object 1 has the highest average predicted interest. The asymptotic learning behavior is also encouraging because it indicates that given more labeled tracks, the model would continue to predict true Object 1 tracks as most interesting.

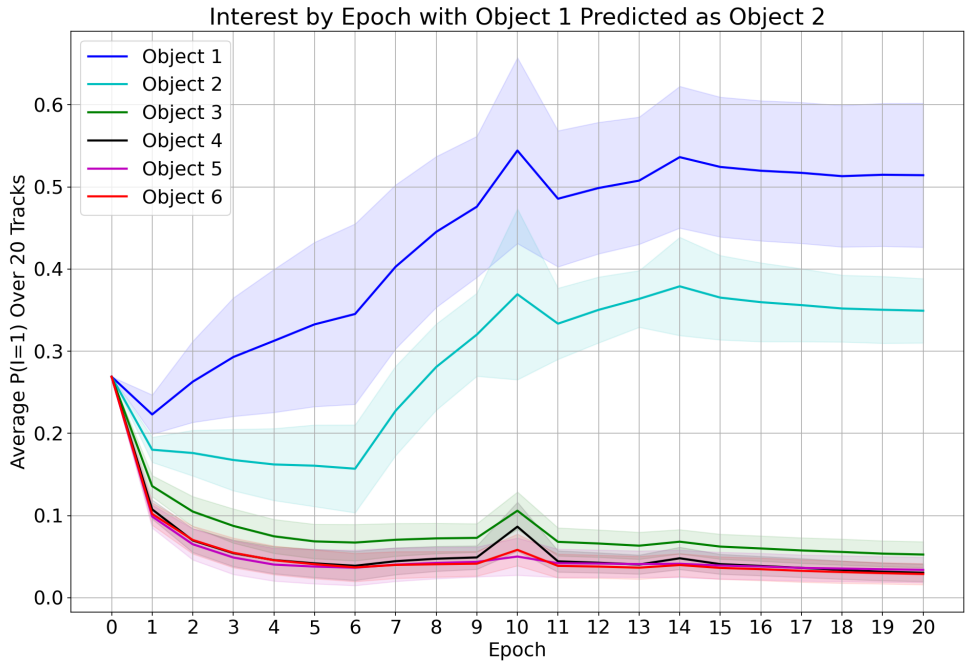
The simulation results for average probability of interest by object type, averaged over 20 test tracks (after 20 training epochs), for each Object 1 typing is shown in Table 4.2.

Table 4.2: Predicted interest probability for each object type averaged over 20 tracks. The posterior weight distributions are after 20 training epochs, given a classifier that predicts Objects 2-6 correctly with probability 1 and Object 1 as the object in the "Predicted Object 1 Type" column with probability 1. The human always labels true Object 1 as interesting and everything else as not interesting. The cells are color-coded such that green cells represent high values and red cells represent low values.

$p(I = 1 \mid \vec{x}, G, \vec{\theta})$						
Predicted Object 1 Type	True Object Type					
	Object 1	Object 2	Object 3	Object 4	Object 5	Object 6
Object 1	0.828	0.059	0.047	0.031	0.03	0.024
Object 2	0.342	0.172	0.057	0.026	0.032	0.028
Object 3	0.439	0.077	0.199	0.024	0.031	0.02
Object 4	0.553	0.077	0.06	0.166	0.033	0.023
Object 5	0.512	0.077	0.062	0.038	0.172	0.02
Object 6	0.502	0.078	0.06	0.036	0.034	0.127

This table indicates that the case where Object 1 is classified as Object 2 with probability 1 is the most challenging for the model. This result is expected because their derived features are most similar, which makes interest separation most difficult. Also, the object

Figure 4.6: Average interest probability and 95% confidence bounds by epoch for each object type with Object 1 predicted as Object 2 with probability 1.



type that Object 1 was classified as always has a relatively high predicted interest value (indicated by the green-shaded cells along the main diagonal). Again, this is expected behavior because the model is not aware of the accuracy of the object type classifier. In all cases, the model is able to rank Object 1 tracks, on average, as more interesting than the rest.

4.3 Poor Human Inputs

Previous sections have assumed that the initial user input defining their interest in different object types and geographic locations were consistent with their actual interest. Here, this assumption will be broken. The user's true interest will still be Object 1 in any geographic location, but their initial inputs are as follows:

$$u^O = [\text{-}1,\ 1,\ 1,\ 1,\ 1,\ 1] \tag{4.4}$$

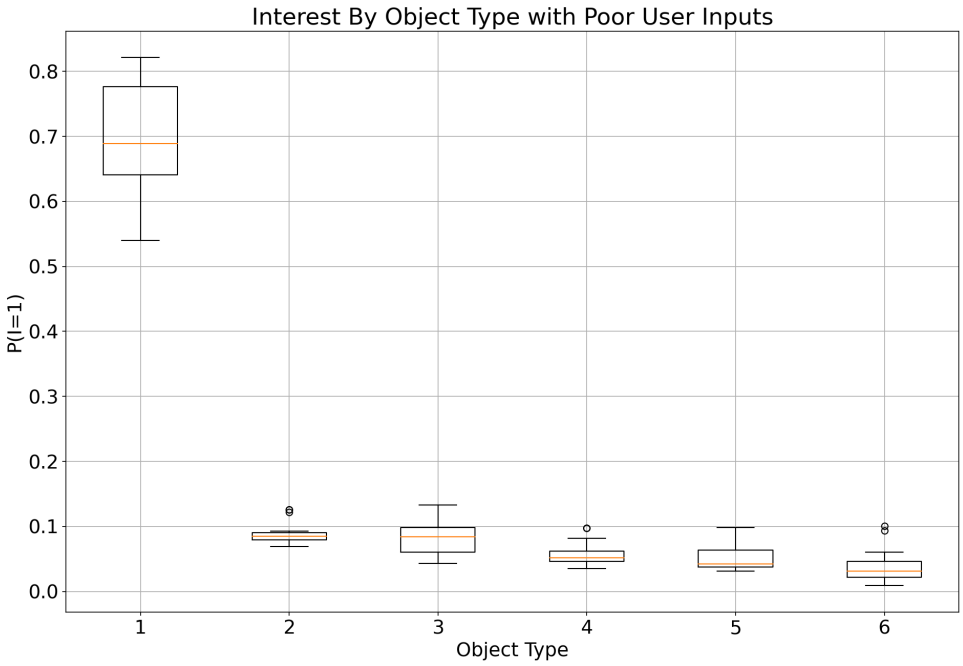
$$u^G = [\text{-}1,\ \text{-}1,\ \text{-}1,\ \text{-}1,\ \text{-}1,\ \text{-}1,\ \text{-}1,\ \text{-}1] \tag{4.5}$$

These inputs indicate that the user is disinterested in Object 1 and interested in all other object types, while being disinterested in all geographic locations (which is in contrast to the true interests defined in Equations 4.1 and 4.2). During simulation, we again assume perfect object typing such that the classifier predicts the true object type with probability 1 and obtain results using using the test methodology outlined in 4.1.

Figure 4.7 shows interest probability for 20 test tracks of each type after 20 training epochs. This plot compares favorably with Figure 4.1, where the initial user inputs were consistent with true interest. As in the ideal case, there is no overlap between the box plot for Object 1 and the box plots for any other object type. However, it is noted that the median interest probability in Object 1 for this case is lower than in the ideal case.

Given the model's ability to eventually recover from bad inputs, it is important to understand how long this process takes. The average predicted human interest for each

Figure 4.7: Probability of human interest over 20 tracks of each object type, after 20 training epochs. The model accurately predicts high probability of interest for Object 1 and low probability of interest for the rest. Despite poor initial inputs, the model performs nearly as well as in the ideal case.



object type is plotted by epoch in Figure 4.8.

Given the inconsistent inputs, at epoch 0 (before any training), the classifier predicts an average interest in Object 1 of about 12%. This is compared to about 50% average probability of interest for all other object types. Given training data, the curve for Object 1 steadily increases, while the other curves steadily decrease. After only 3 training epochs, probability of interest in Object 1 overtakes all other object types. Learning true interest takes longer in this case than previously tested non-ideal scenarios because the prior weights are fundamentally incorrect, but given a limited amount of labeled tracks, the model is able to recover. The learning rate could potentially be expedited by allowing for more variance on the initial weights, but this would come at the expense of trusting labeled tracks significantly more than initial inputs. This analysis suggests that the model is robust to poor user inputs assuming favorable object typing and consistent user track labeling.

4.4 Model Posteriors

One major advantage of using a Bayesian classifier as opposed to a black box classifier (i.e. a neural network) is that the posterior distributions give insight as to how the model learns. Below, marginal prior and posterior weight distributions (after 20 training epochs) are shown for all test cases outlined above. In the presented results, the posteriors are shown for:

- The ideal case (Section 4.1);
- $\gamma = 0.167$ for the case of the varying confidence (Section 4.2.1);
- Object 1 classified as Object 2 with probability 1 for the incorrect, confident classifier case (Section 4.2.2);
- The case of poor user inputs (Section 4.3).

Firstly, Figure 4.9 shows the intercept weight distribution. While important for classification, the distribution does not give significant insight into how the model learned.

Figure 4.8: Average interest probability and 95% confidence bounds by epoch for each object type, given poor initial inputs.

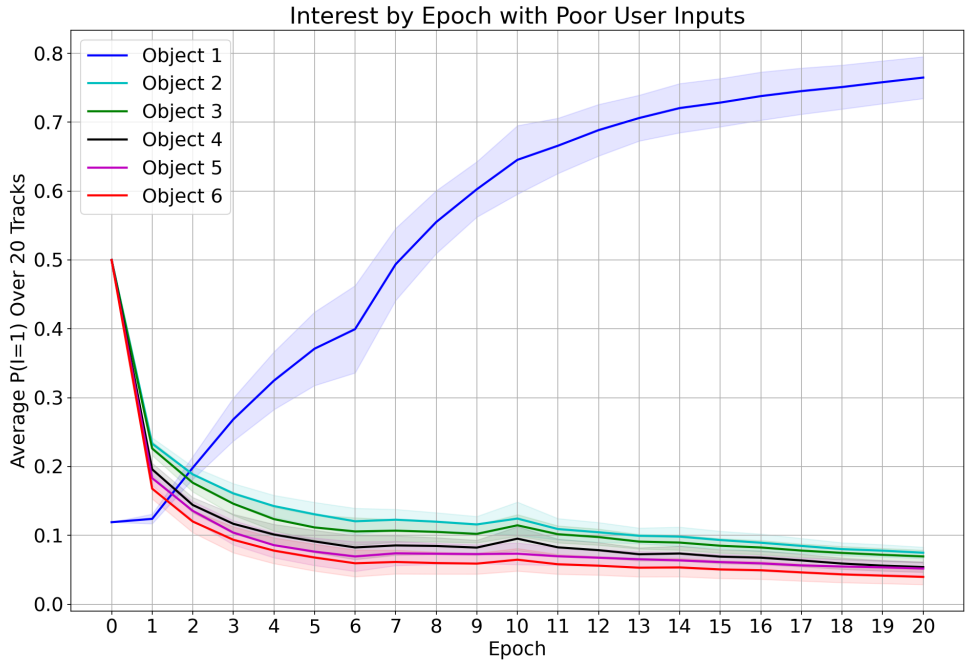
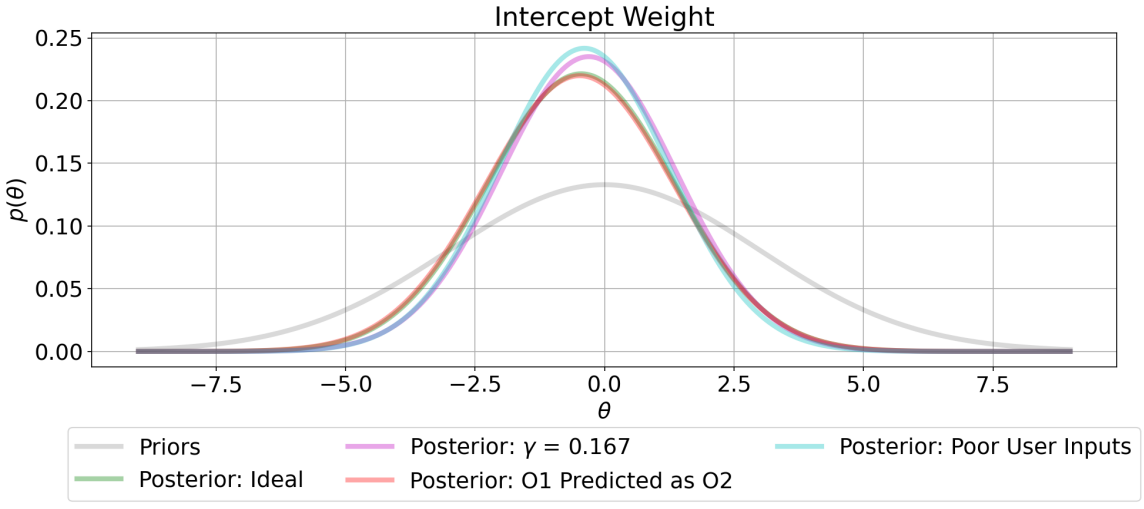
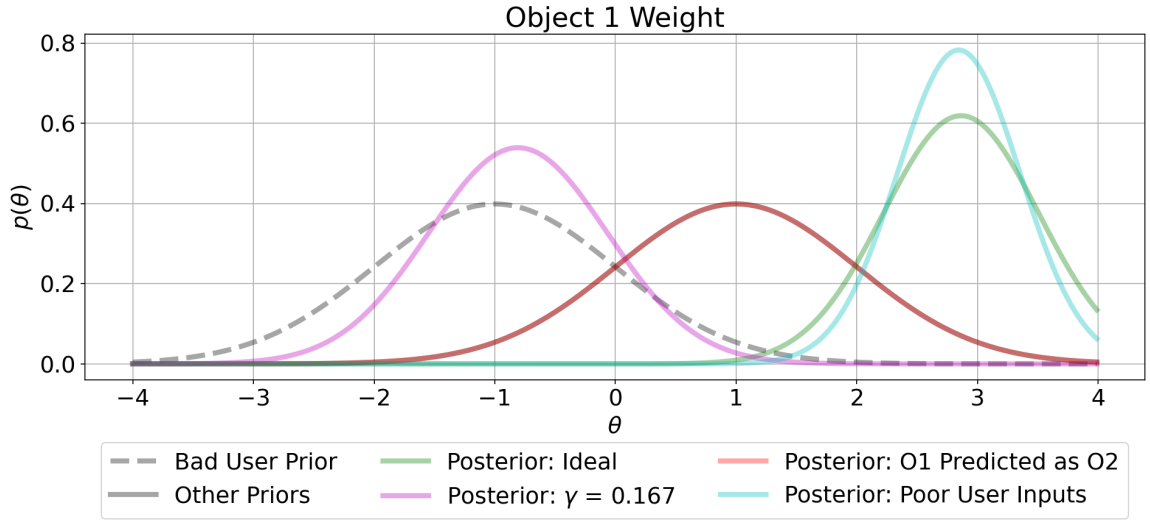


Figure 4.9: Marginal prior and posterior intercept weight distributions for all tested cases.



Figures 4.10-4.15 show the distributions for the object type weights. For the Object 1 weight, in the ideal case and the case with poor user inputs, the posterior shifts significantly to the right. This indicates a positive preference for Object 1 (which is expected). For the case of $\gamma = 0.167$, the posterior shifts to the left because the object type cannot be used to predict interest. In the case where Object 1 is typed as Object 2 with probability 1, the posterior matches the prior. This is again expected behavior because the model never sees any object that has any probability of being Object 1. The weight distributions for all other object types shift to the left in every test case except for $\gamma = 0.167$ (for all weights) and Object 1 predicted as Object 2 (only the object 2 weight). This behavior makes sense for $\gamma = 0.167$ because object type is not particularly predictive of interest. It is also expected for the latter case because the model sees that every track labeled as interesting has a 100% chance of being Object 2.

Figure 4.10: Marginal prior and posterior Object 1 weight distributions for all tested cases.



Marginal priors and posteriors are shown for geographic location weights in Figures 4.16-4.23. Note that all training tracks were located in the Middle East, Russia, Europe, or the United States. These locations correspond to discrete geographic locations of Asia, Europe, and North America. As such, the posterior match the priors for all other geographic

Figure 4.11: Marginal prior and posterior Object 2 weight distributions for all tested cases.

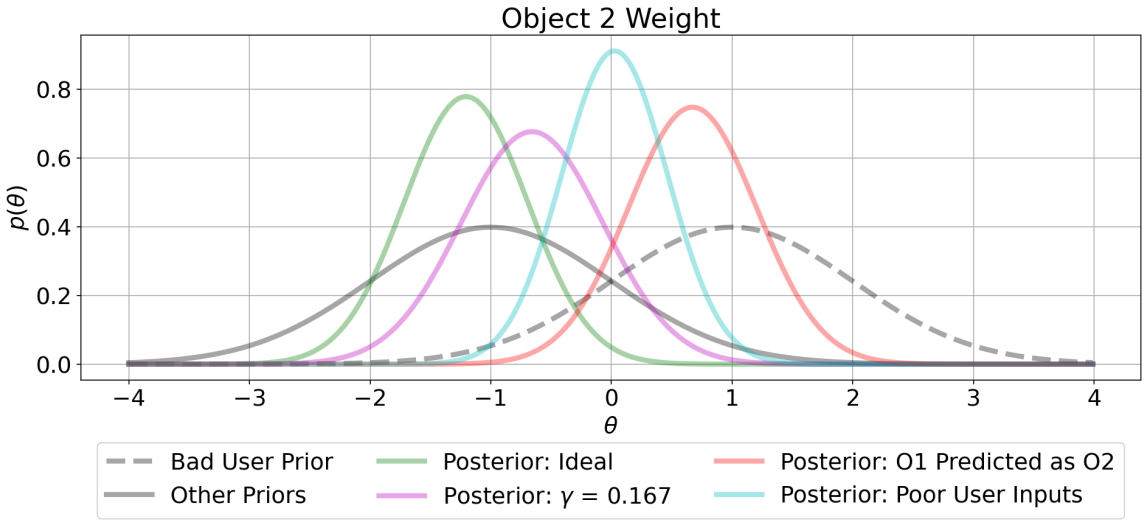


Figure 4.12: Marginal prior and posterior Object 3 weight distributions for all tested cases.

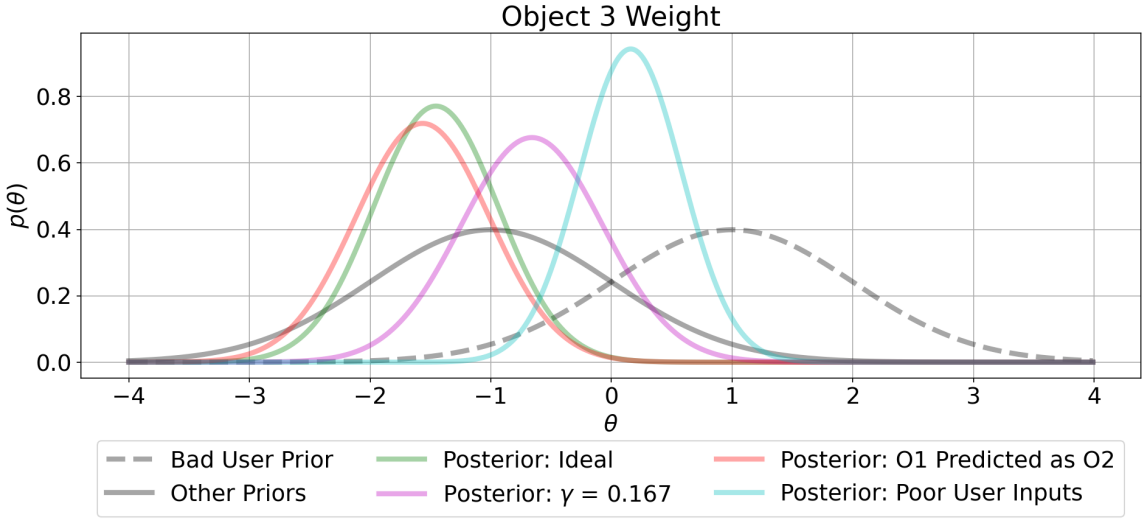


Figure 4.13: Marginal prior and posterior Object 4 weight distributions for all tested cases.

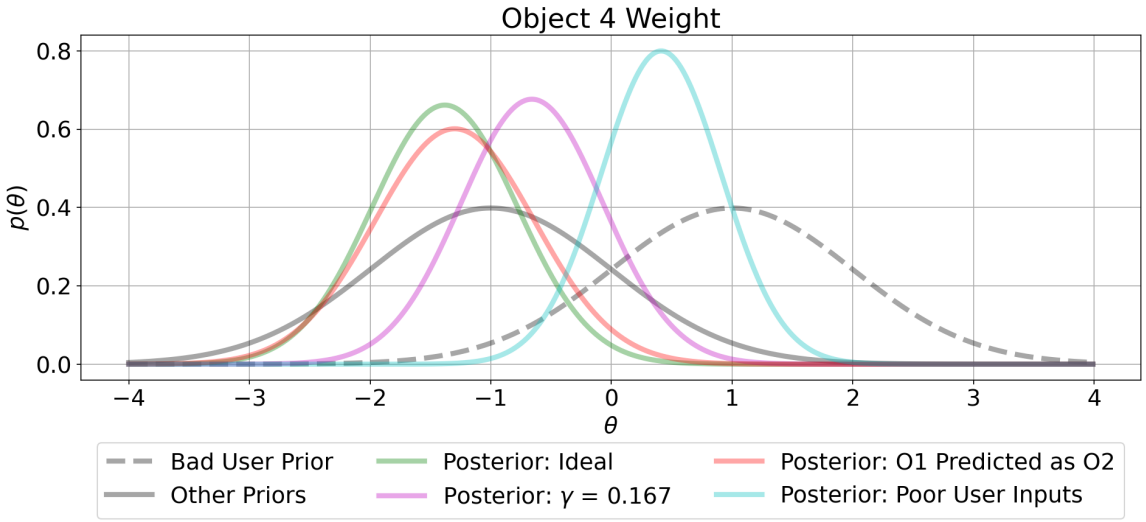


Figure 4.14: Marginal prior and posterior Object 5 weight distributions for all tested cases.

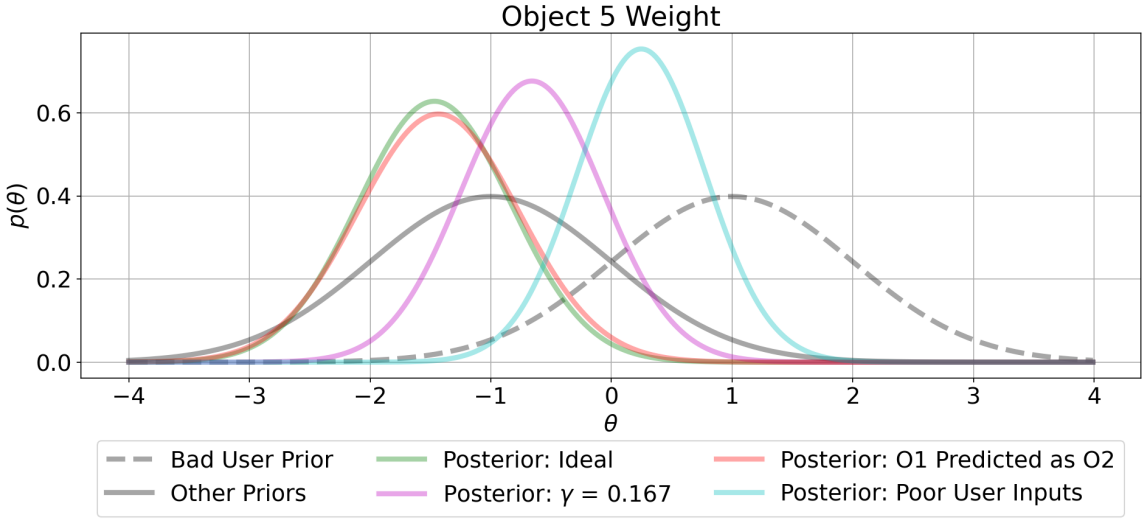
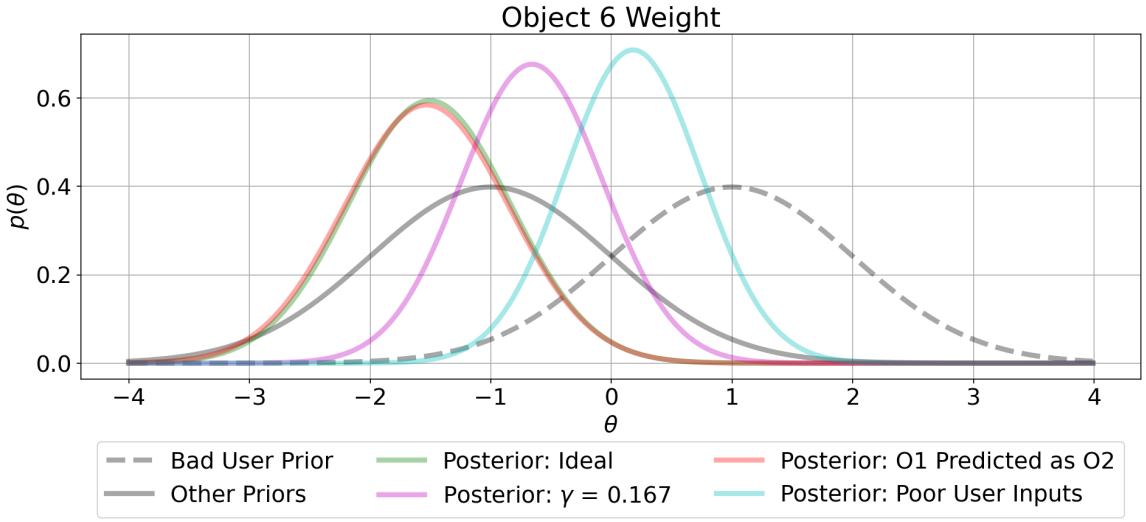
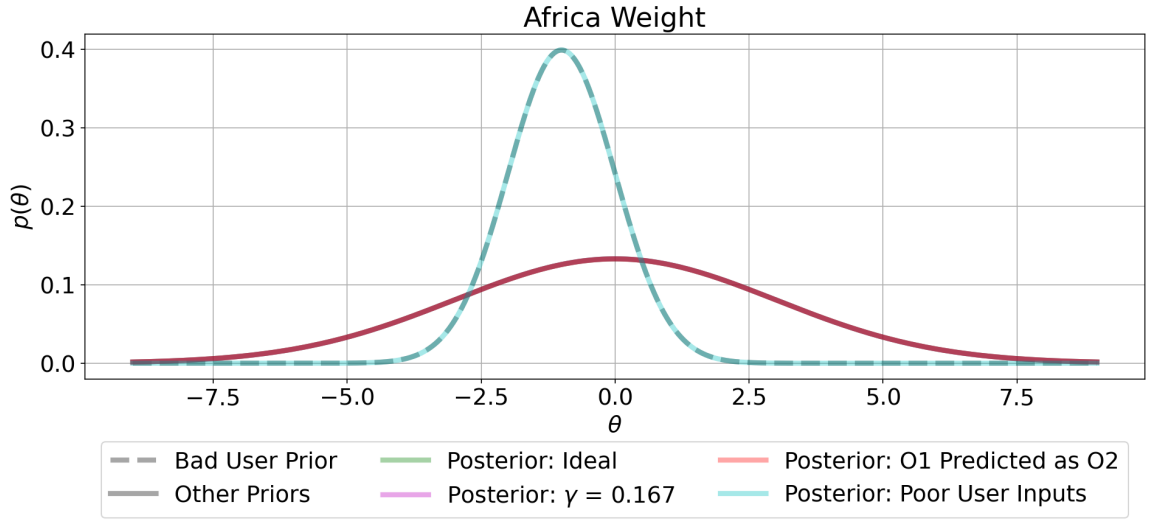


Figure 4.15: Marginal prior and posterior Object 6 weight distributions for all tested cases.



locations (because the model did not see targets in these locations). In general, the posteriors for Asia, Europe, and North America have similar expected values as the priors with decreased variance. Given that the simulated operator made no interest labeling decisions based off of geographic location, this behavior is expected.

Figure 4.16: Marginal prior and posterior Africa weight distributions for all tested cases.



Finally, marginal prior and posterior weight distributions for derived features are shown in Figures 4.24-4.26. From Figures 3.6-3.8, it can be seen that Object 1 (the true object of interest) has:

- Relatively low scaled maximum altitude;
- Relatively high scaled maximum intensity;
- Relatively low scaled maximum speed.

In all tested cases, the maximum altitude weight posterior shifts to the left, the maximum intensity weight posterior shifts to the right, and the maximum speed weight posterior shifts to the left. This is consistent with the features associated with Object 1 tracks.

As a whole, the weight distribution plots offer insight as to how the model predicts interest after training. In less challenging cases (the ideal and poor user input cases), object

Figure 4.17: Marginal prior and posterior Asia weight distributions for all tested cases.

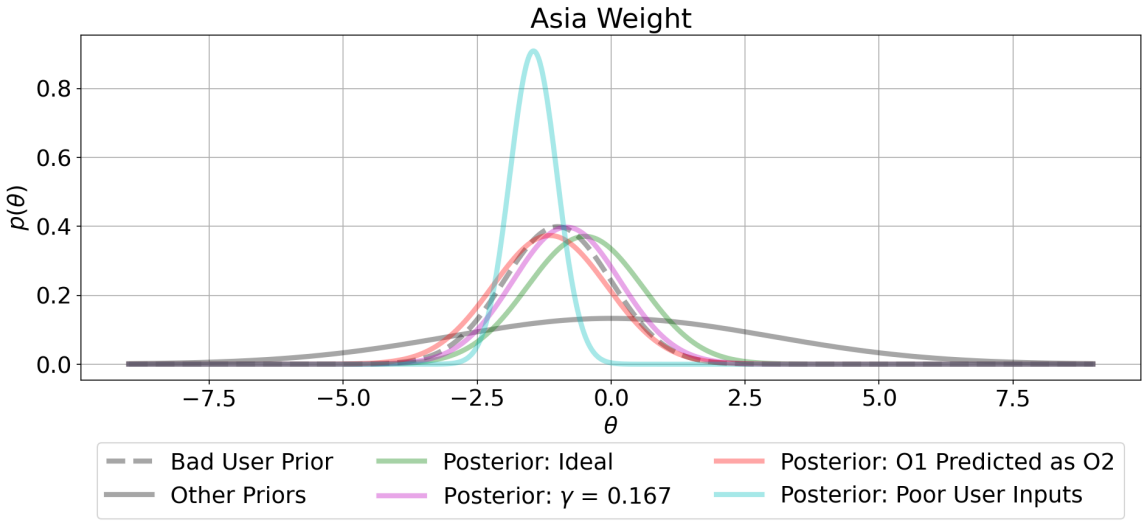


Figure 4.18: Marginal prior and posterior Caribbean weight distributions for all tested cases.

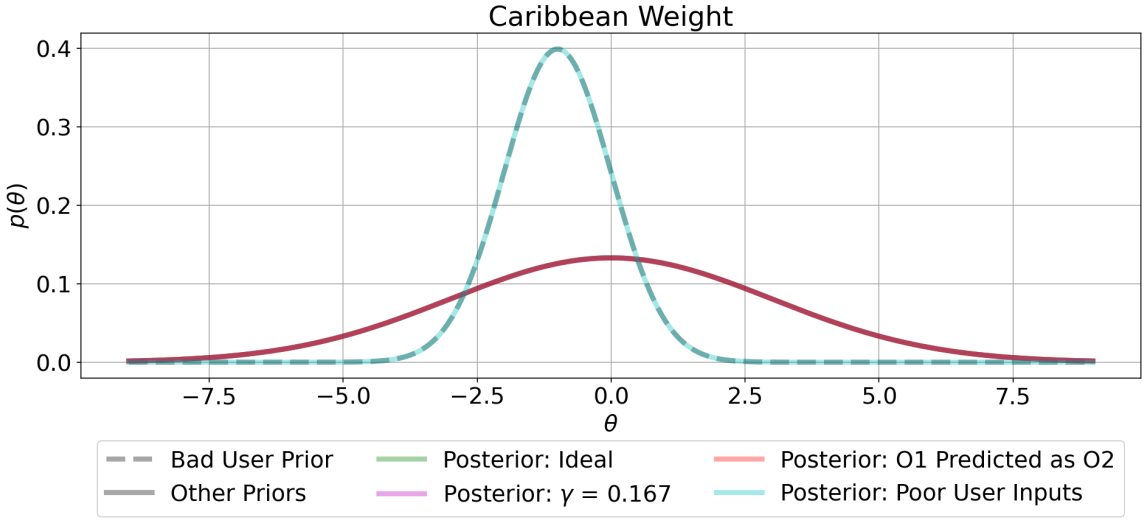


Figure 4.19: Marginal prior and posterior Central America weight distributions for all tested cases.

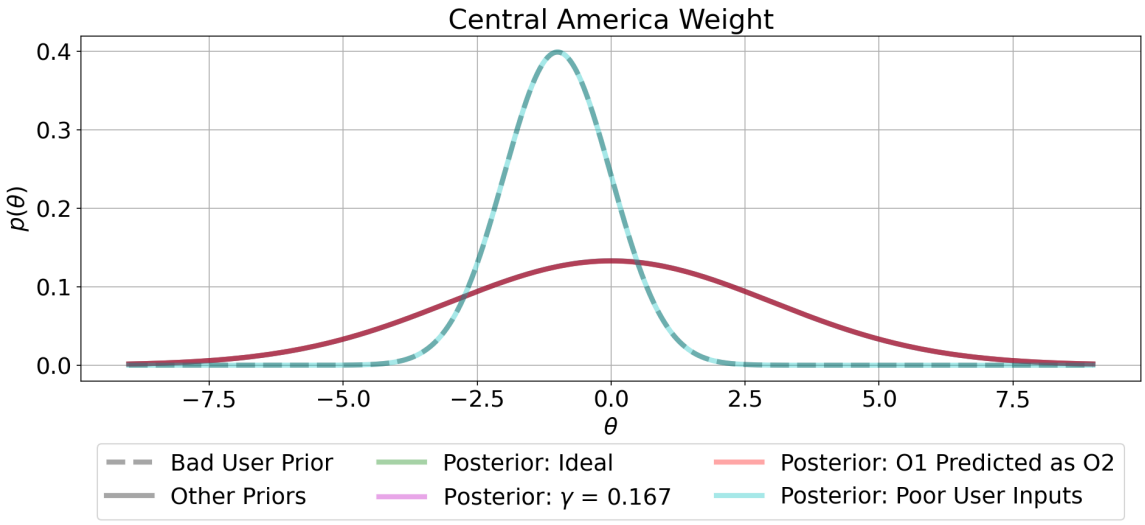


Figure 4.20: Marginal prior and posterior Europe weight distributions for all tested cases.

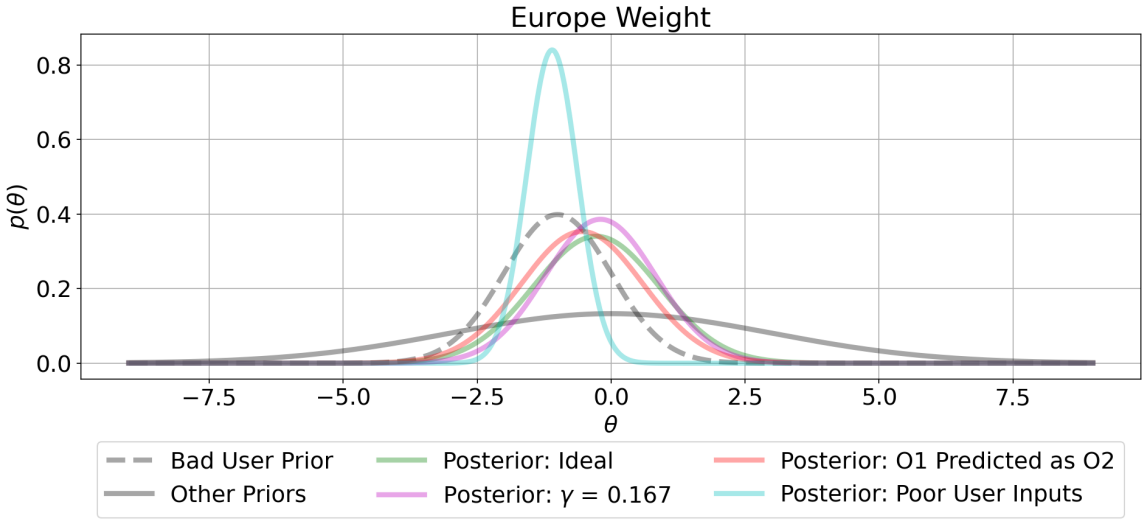


Figure 4.21: Marginal prior and posterior North America weight distributions for all tested cases.

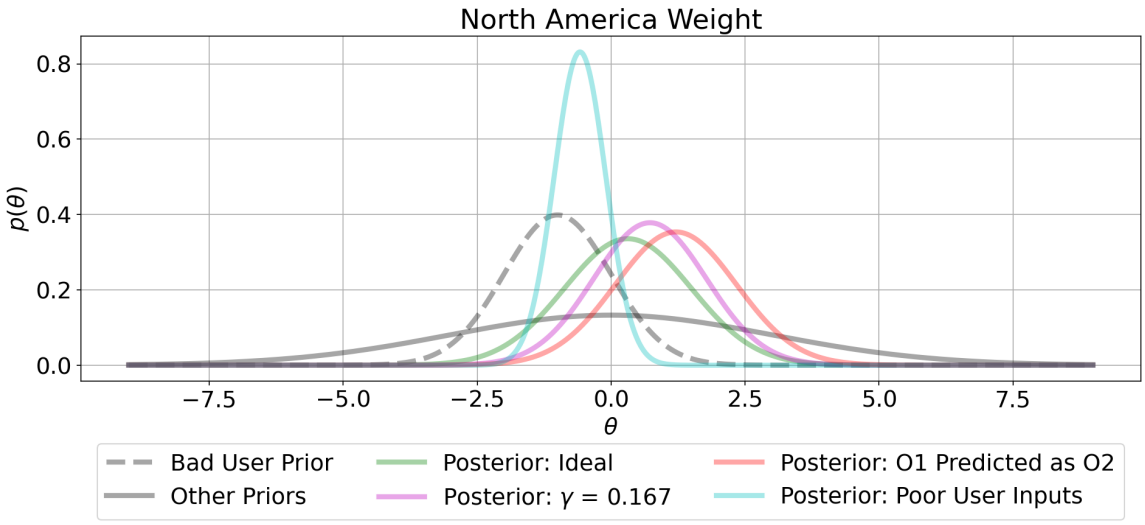


Figure 4.22: Marginal prior and posterior Oceania weight distributions for all tested cases.

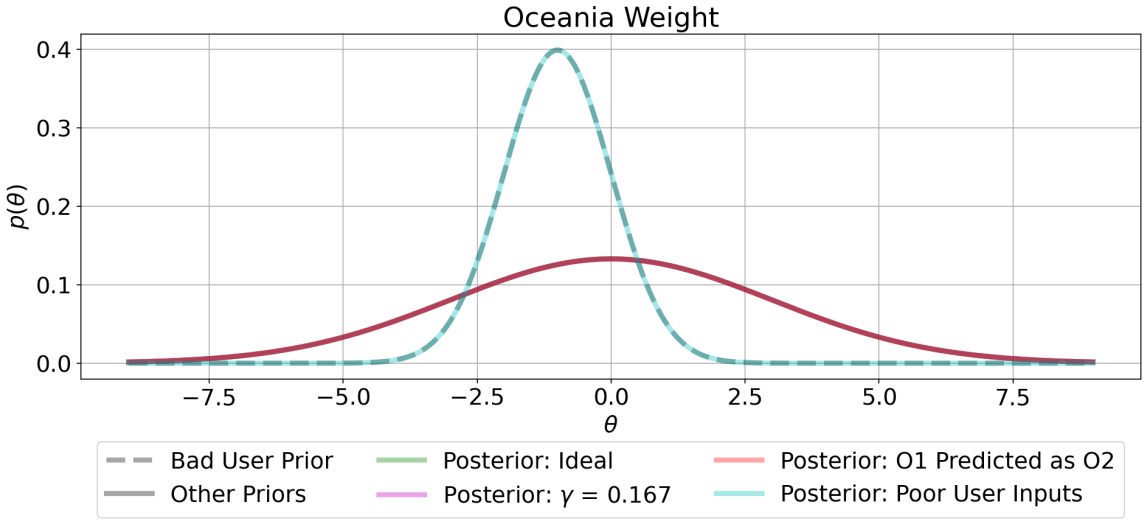


Figure 4.23: Marginal prior and posterior South America weight distributions for all tested cases.

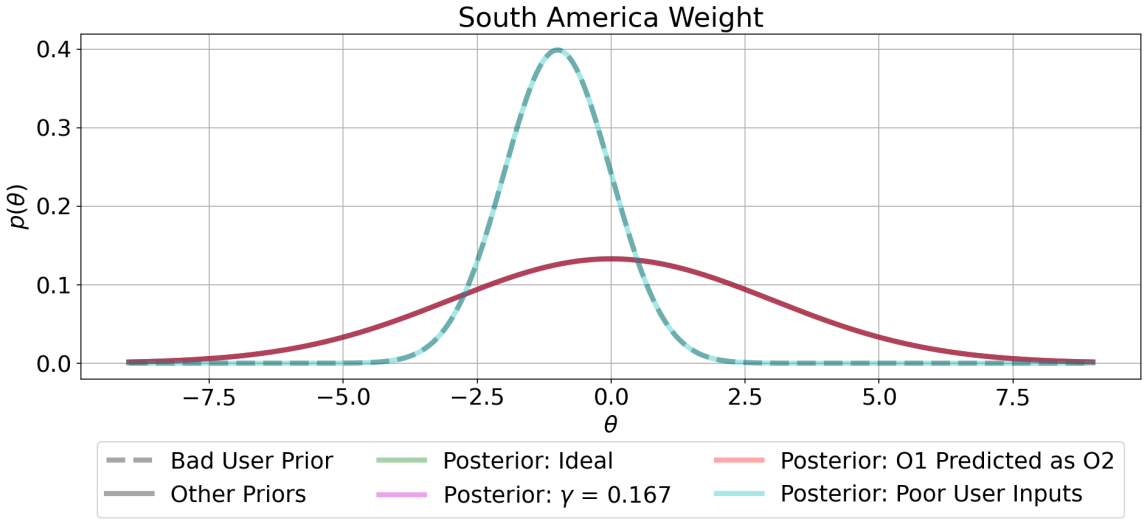


Figure 4.24: Marginal prior and posterior scaled maximum altitude weight distributions for all tested cases.

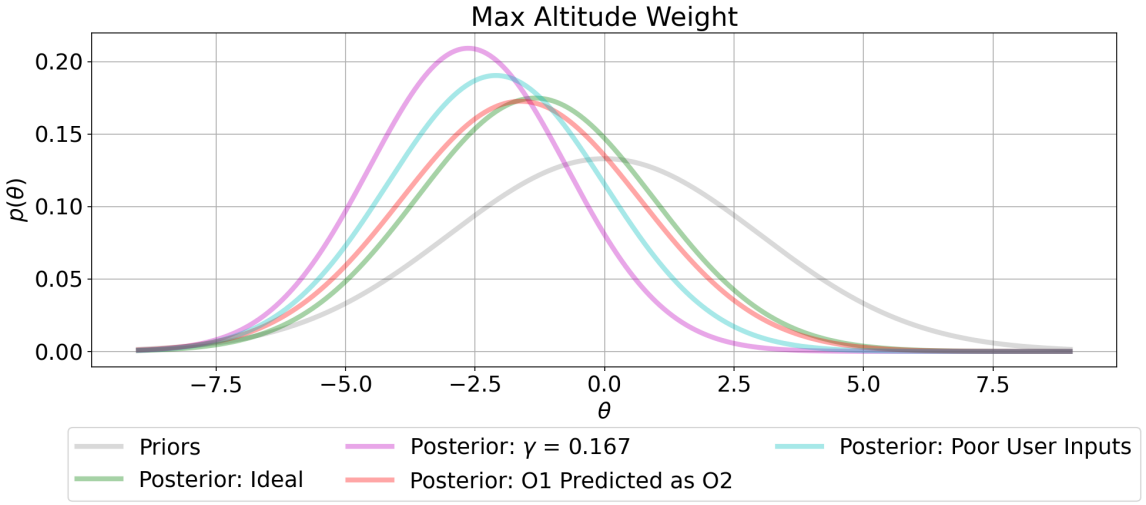


Figure 4.25: Marginal prior and posterior scaled maximum intensity weight distributions for all tested cases.

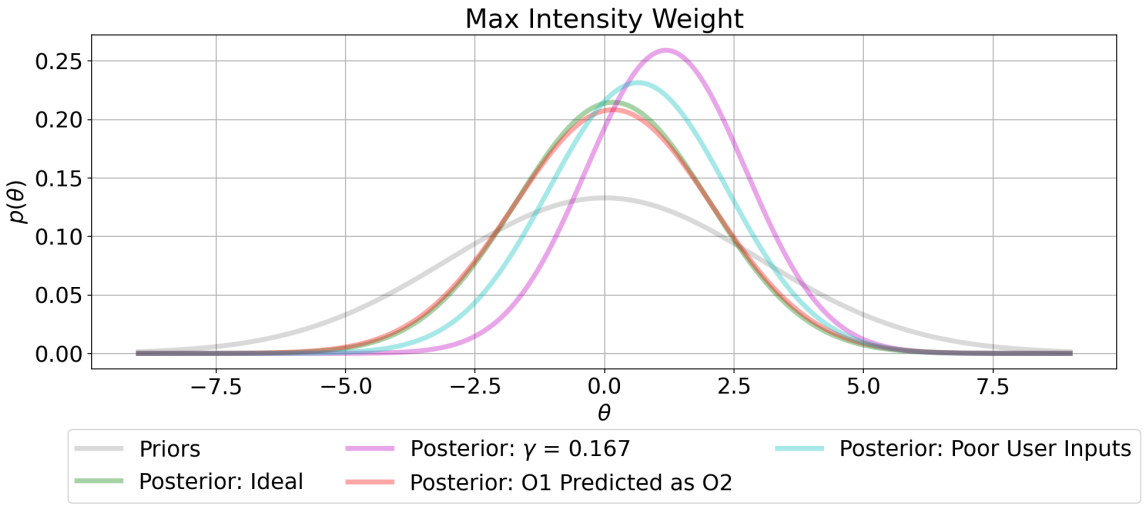
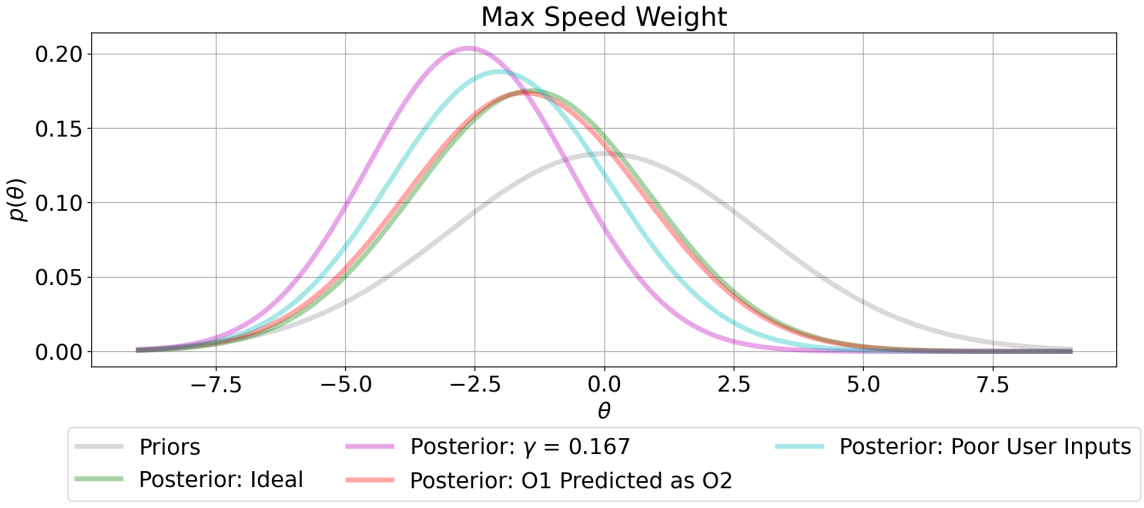


Figure 4.26: Marginal prior and posterior scaled maximum speed weight distributions for all tested cases.



type is an important predictor of interest. However, in the more challenging cases (the poor object type classifier cases), derived features are used to make predictions on human interest. This analysis indicates that including more track features as indicators of human interest could make the model more robust to challenging edge cases.

4.5 Realistic Case

In this section, a comprehensive view of model performance is presented for a more realistic test scenario. In this case, γ (as defined in Equation 4.3) is sampled from $\mathcal{U}[0.3, 0.9]$ for each training and test track, with the intention of introducing randomness/imperfections. The simulated operator also does not label every training track as interesting or not interesting, and instead labels with probability, f (labeling is still according to u_T^O and u_T^G). The operator also provides no prior information such that $u^O = u^G = \vec{0}$. The test methodology for this section is:

(1) Randomly sort training tracks;

(2) Present user with a single training track;

(3) Update weight distribution given labeled training track (if user labeled it);

(4) Score 20 test tracks of each object type for interest probability (where the same 20 test tracks are used after each training track);

(5) Repeat steps 2-4 for 100 training tracks;

(6) Repeat steps 1-5 for 50 total trials.

Average probability of interest over 20 test tracks of each object type by number of tracks seen are shown in Figures 4.27-4.29. These plots represent $f = 1$, 0.5, and 0.25, respectively, and include 50 random trials (where training tracks are randomly sorted).

In all plots, Object 1 interest probability is compared to all other object types, as it is the only object of true interest. In all cases, after 100 seen training tracks, Object 1 is

Figure 4.27: Average probability of interest over 20 test tracks of each object type for 50 trials. In this case $f = 1$ (operator always labels training tracks).

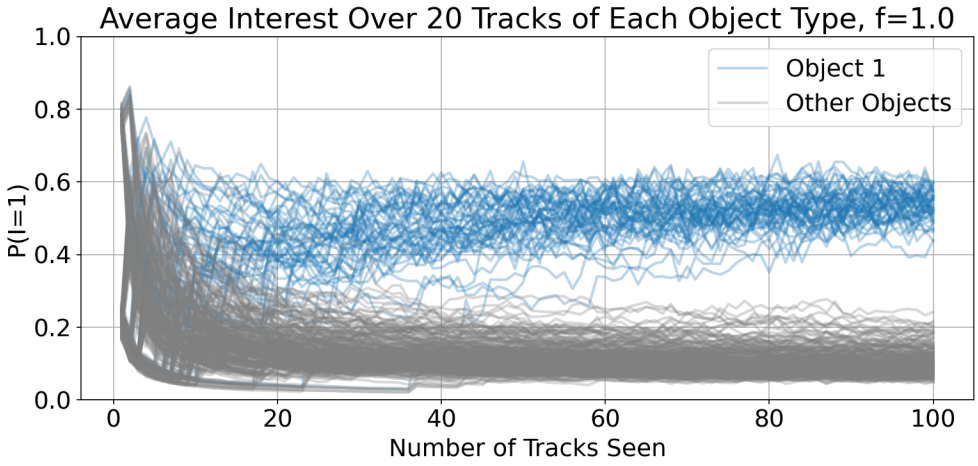


Figure 4.28: Average probability of interest over 20 test tracks of each object type for 50 trials. In this case $f = 0.5$ (operator labels training tracks with 50% probability).

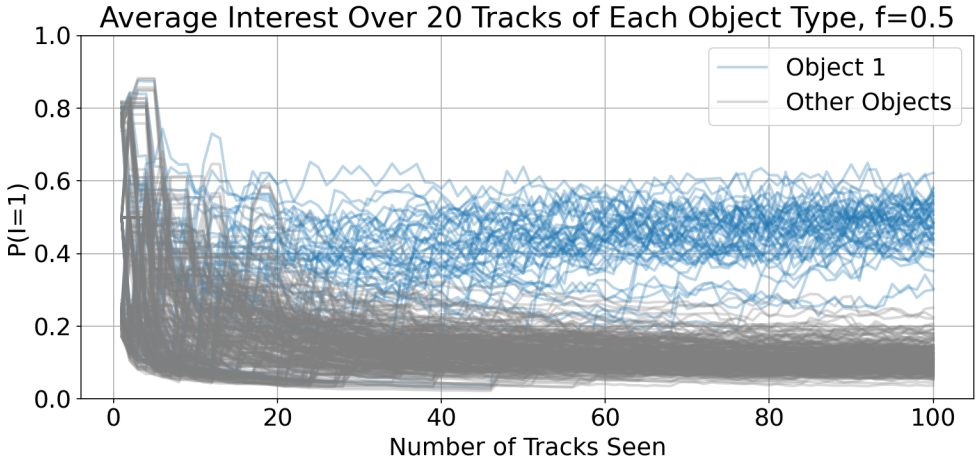
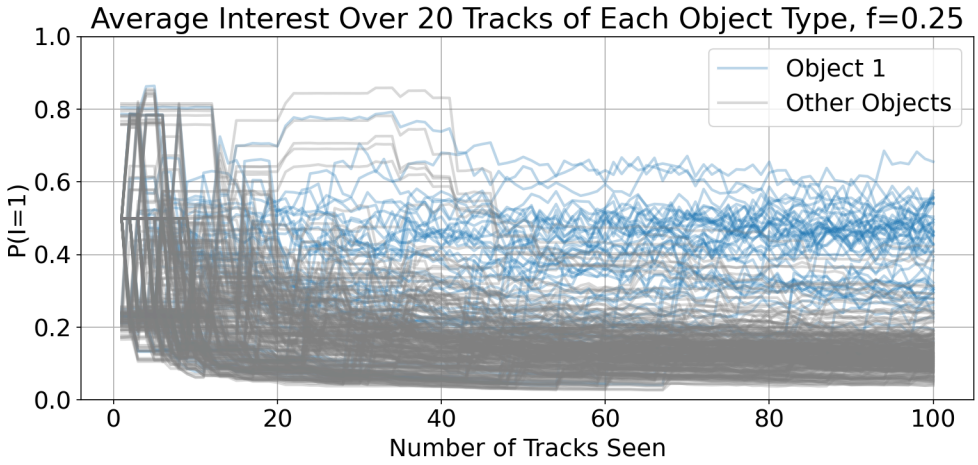


Figure 4.29: Average probability of interest over 20 test tracks of each object type for 50 trials. In this case $f = 0.25$ (operator labels training tracks with 25% probability).



generally considered to be the most interesting object type over 50 trials. As is expected, as f decreases, the model takes longer to learn true operator interest because it has less labeled training data.

In an effort to objectively quantify model performance in a realistic test case, the figure of merit called Discounted Cumulative Gain (DCG) is introduced [12]. This is a metric to determine the functionality of a ranking system and is defined as:

$$DCG = \sum_{i=1}^{n} \left[\frac{\mathrm{rel}_i}{\log_2(i+1)} \right]. \tag{4.6}$$

In the above, n represents the number of presented tracks (in this case $n = 120$, 20 for each object type) and rel is a list of true interest sorted by predicted interest, where the individual elements are defined by:

$$\mathrm{rel}_i = \begin{cases} 1 & \text{track is interesting,} \\ -1 & \text{track is not interesting.} \end{cases} \tag{4.7}$$

DCG is used to penalize objects (in this case tracks) for being sorted in the wrong position. DCG by number of tracks seen is plotted over 50 trials in Figure 4.30-4.32. These plots correspond to $f = 1$, 0.5, and 0.25, respectively.

These figures compare sorting provided by the model and random sorting. In all cases, the model quickly begins to exceed the capability of random sorting, and eventually approaches ideal sorting. As the probability of an operator labeling a track decreases, learning slows, and DCG takes longer to approach the ideal sorting. Figures 4.33-4.35 provide snapshots of the DCG distribution for all tested values of f vs. random sorting at discrete numbers of tracks seen.

For all presented cases, the model performs significantly better than random sorting. After 25 tracks seen, $f = 1$ approaches the ideal DCG across all trials, with $f = 0.5$ and $f = 0.25$ lagging behind. After 50 tracks seen, $f = 1$ sorts ideally, $f = 0.5$ approaches ideal

Figure 4.30: DCG given 20 test tracks of each object type for 50 trials. In this case $f = 1$ (operator always labels training tracks).

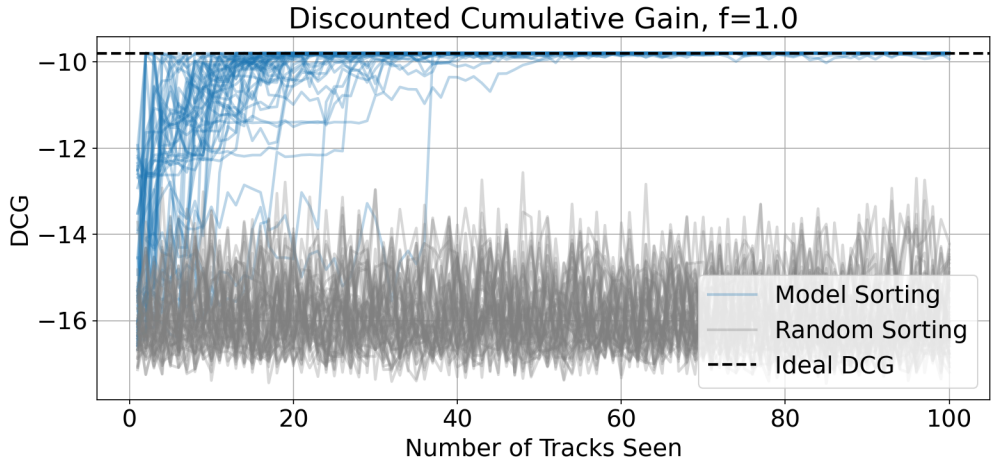


Figure 4.31: DCG given 20 test tracks of each object type for 50 trials. In this case $f = 0.5$ (operator labels training tracks with 50% probability).

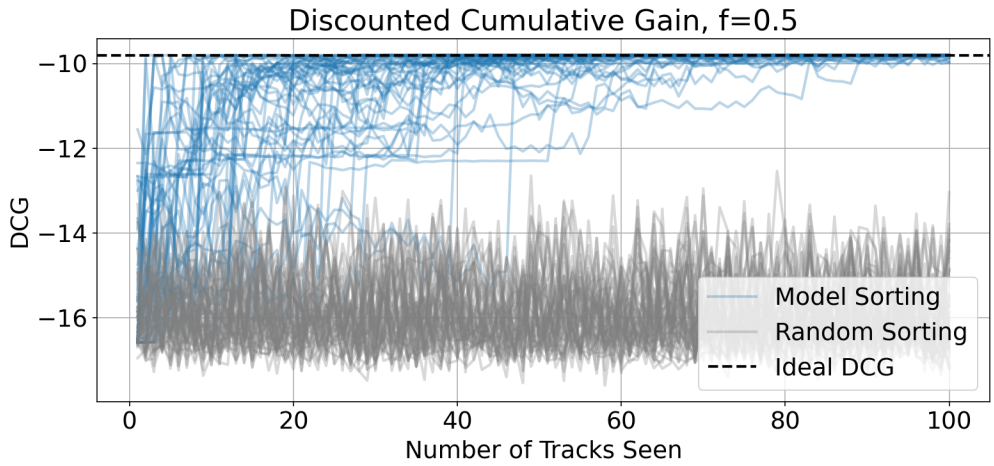


Figure 4.32: DCG given 20 test tracks of each object type for 50 trials. In this case $f = 0.25$ (operator labels training tracks with 25% probability).

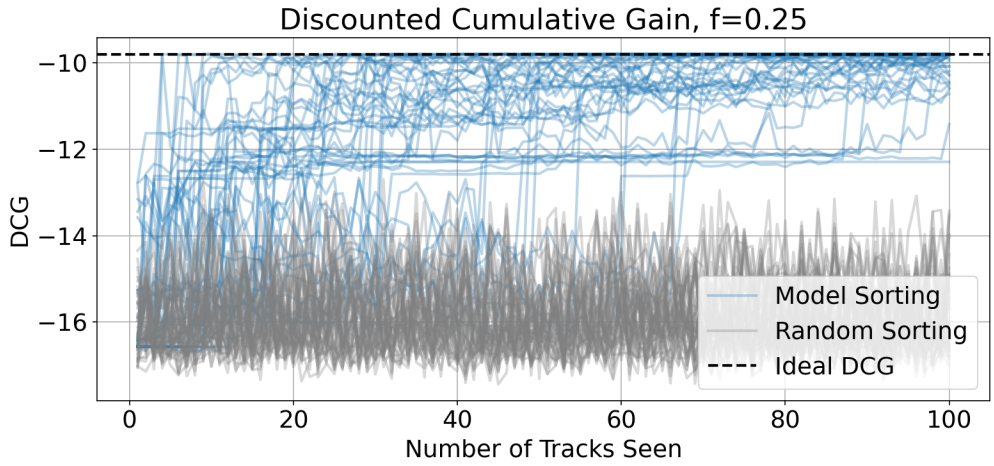


Figure 4.33: Histogram of DCG given 20 test tracks of each object type over 50 trials. In this case the number of tracks seen is 25.

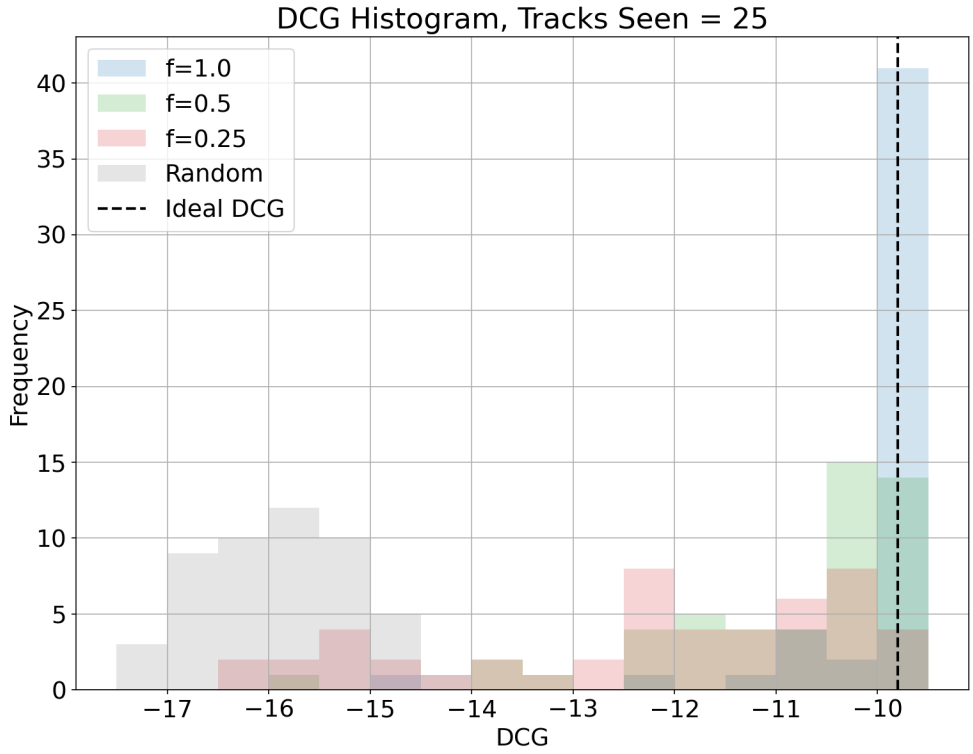


Figure 4.34: Histogram of DCG given 20 test tracks of each object type over 50 trials. In this case the number of tracks seen is 50.

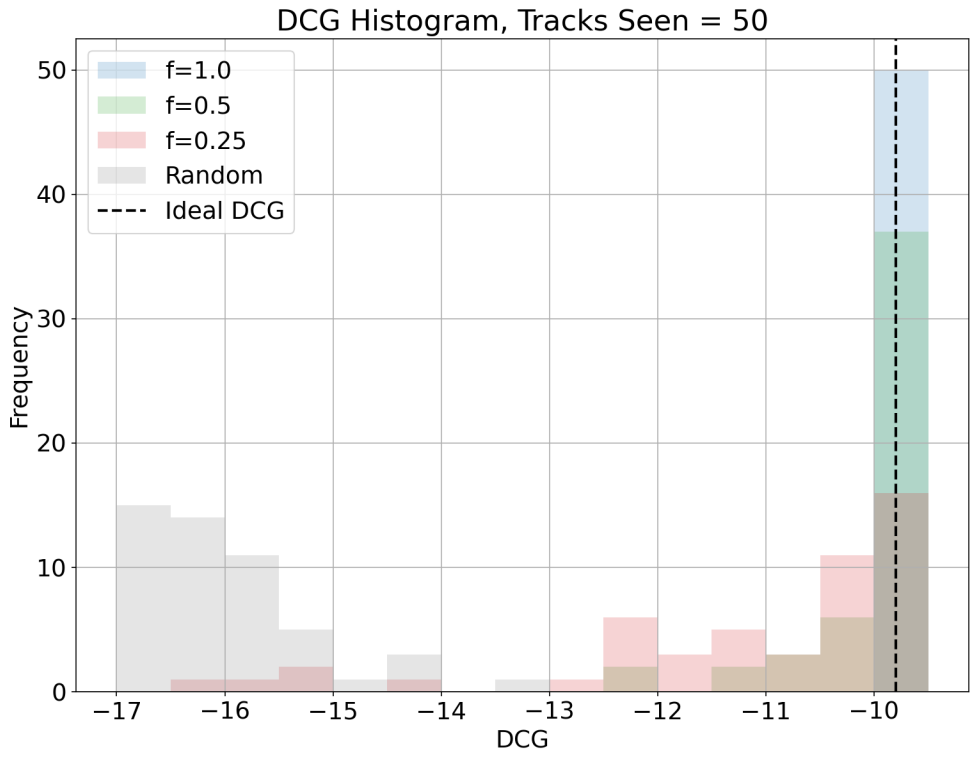
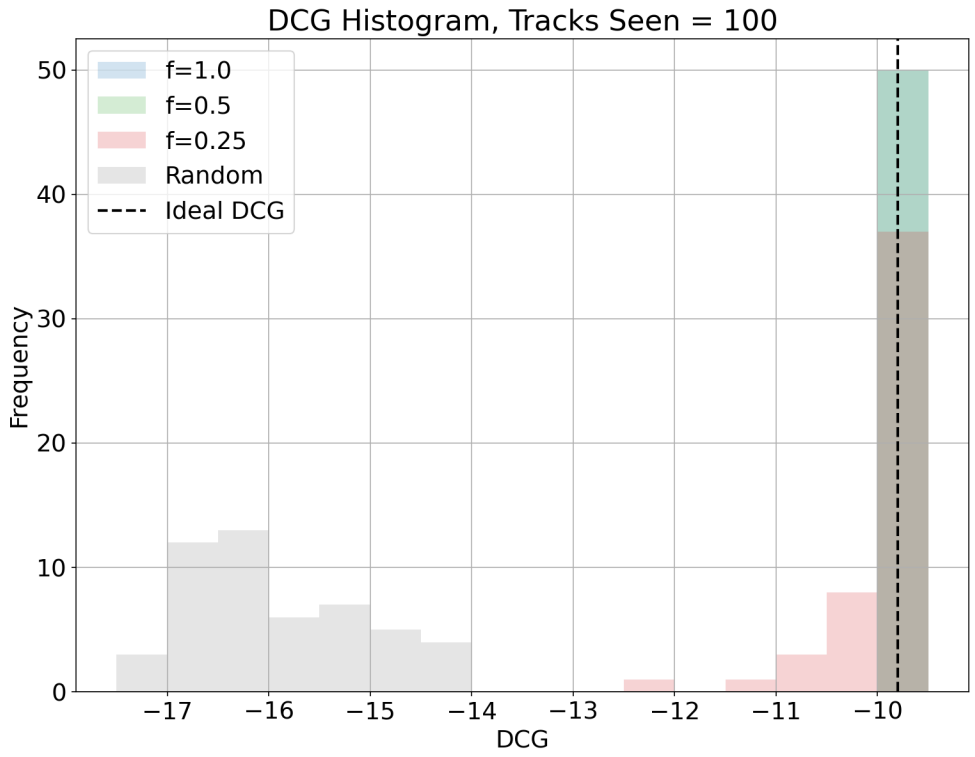


Figure 4.35: Histogram of DCG given 20 test tracks of each object type over 50 trials. In this case the number of tracks seen is 100.



sorting, and $f = 0.25$ continues to lag behind. After 100 tracks seen, $f = 1$ and $f = 0.5$ sort ideally, and $f = 0.25$ approaches ideal sorting. This analysis indicates that after about 25 labeled tracks, in the presented realistic scenario, the model approaches ideal sorting.

4.6 Remarks

The presented results show that the model performs well in an ideal case, is robust to poor object typing (both for a varying confidence in true object type and for high confidence in incorrect target types), and is robust to poor initial human inputs. In a realistic scenario, the model requires about 25 tracks to sort all tracks ideally. There are many other failure modes that could be considered, such as inconsistent human track labeling, or a combination of the studied results. However, many of the potential vulnerabilities could be remedied by the addition of a more comprehensive vector of derived track features, $f(\vec{x})$.

Chapter 5

Conclusions & Future Work

The main contribution of this thesis is the introduction of a probabilistic model to learn and predict human interest in the context of large-scale data-driven tracking and surveillance missions, inspired by OBAC operations. The model is similar to a Bayesian logistic regression, but includes dependencies between input variables, as well as partially observable input variables. While the model was specifically designed for the considered context, it could easily be updated to act as a discrete classifier for other problems.

This work introduced the probabilistic models for both parameter learning and human interest inference, while presenting the derivations necessary to calculate conditional posterior distributions. In the case of intractable posteriors, the Laplace approximation was used to estimate the learned regression parameter posterior distribution as a Gaussian. Despite its limitations, this approximation allows for real-time calculation of posteriors, such that it can easily be used in an online setting.

Experimental results were presented to show how the model behaves in a variety of settings. In an ideal case, the model's ability to classify human interest in a track was very favorable. It was shown that performance did suffer in challenging conditions, such as the case of a poor classifier or poor initial preference inputs by human operators. However, in both cases, the model was still able to accurately learn human interest given labeled training data. In a presented realistic scenario, it was determined that about 25 labeled tracks were required to approach perfect sorting of test tracks. The tested case included no

prior information, indicating that this number could be decreased significantly with provided human inputs. In general, the necessary amount of labeled data to accurately infer human interest was very minimal.

Future work may consider using an activation function other than a logistic sigmoid, which could make the model more robust to logical operators defining human interest. Non-linear functions will be considered in order to better classify non-linearly separable track feature data. Techniques to estimate intractable posteriors that do not assume a Gaussian form should also be investigated. However, these approximations should be constrained to be functional in real-time, as is the case with the Laplace approximation. More model validation would also be appropriate. This could include untested failure modes, a combination of the tested non-ideal cases, and testing with real operator inputs. Finally, the model could be expanded to include more track features $f(\vec{x})$ in an effort to better classify human interest.

Bibliography

[1] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. Heliyon, 4(11):e00938, 2018.

[2] Daniel Acuna, Paul Schrater, et al. Bayesian modeling of human sequential decision-making on the multi-armed bandit problem. In Proceedings of the 30th annual conference of the cognitive science society, volume 100, pages 200–300. Citeseer, 2008.

[3] Nisar Ahmed, Mark Campbell, David Casbeer, Yongcan Cao, and Derek Kingston. Fully Bayesian learning and spatial reasoning with flexible human sensor networks. In Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems, pages 80–89, 2015.

[4] Aitor Almeida and Gorka Azkune. Predicting human behaviour with recurrent neural networks. Applied Sciences, 8(2):305, 2018.

[5] Nikhil Bhat, Vivek F Farias, Ciamac C Moallemi, and Deeksha Sinha. Near-optimal ab testing. Management Science, 66(10):4477–4495, 2020.

[6] Christopher M Bishop and Nasser M Nasrabadi. Pattern recognition and machine learning, volume 4. Springer, 2006.

[7] Erik Brynjolfsson, Daniel Rock, and Chad Syverson. Artificial intelligence and the modern productivity paradox: A clash of expectations and statistics. In The economics of artificial intelligence: An agenda, pages 23–57. University of Chicago Press, 2018.

[8] George Casella and Edward I George. Explaining the Gibbs sampler. The American Statistician, 46(3):167–174, 1992.

[9] Jocelyn Cranefield, Michael Winikoff, Yi-Te Chiu, Yevgeniya Li, Cathal Doyle, and Alex Richter. Partnering with AI: The case of digital productivity assistants. Journal of the Royal Society of New Zealand, pages 1–24, 2022.

[10] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. The YouTube video recommendation system. In Proceedings of the fourth ACM conference on Recommender systems, pages 293–296, 2010.

[11] Longjun Dong, Johan Wesseloo, Yves Potvin, and Xibing Li. Discrimination of mine seismic events and blasts using the Fisher classifier, naive Bayesian classifier and logistic regression. Rock Mechanics and Rock Engineering, 49:183–211, 2016.

[12] Georges Dupret. Discounted cumulative gain and user decision models. In String Processing and Information Retrieval: 18th International Symposium, SPIRE 2011, Pisa, Italy, October 17-21, 2011. Proceedings 18, pages 2–13. Springer, 2011.

[13] Crícia Z Felício, Klérisson VR Paixão, Celia AZ Barcelos, and Philippe Preux. A multi-armed bandit model selection for cold-start user recommendation. In Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, pages 32–40, 2017.

[14] Alexander Genkin, David D Lewis, and David Madigan. Large-scale Bayesian logistic regression for text categorization. Technometrics, 49(3):291–304, 2007.

[15] Wally R Gilks, Nicky G Best, and Keith KC Tan. Adaptive rejection Metropolis sampling within Gibbs sampling. Journal of the Royal Statistical Society Series C: Applied Statistics, 44(4):455–472, 1995.

[16] Walter R Gilks and Pascal Wild. Adaptive rejection sampling for Gibbs sampling. Journal of the Royal Statistical Society: Series C (Applied Statistics), 41(2):337–348, 1992.

[17] Carlos A Gomez-Uribe and Neil Hunt. The Netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS), 6(4):1–19, 2015.

[18] Jong Woo Kim, Byung Hun Lee, Michael J Shaw, Hsin-Lu Chang, and Matthew Nelson. Application of decision-tree induction techniques to personalized advertisements on internet storefronts. International Journal of Electronic Commerce, 5(3):45–62, 2001.

[19] Ron Kohavi and Roger Longbotham. Online controlled experiments and a/b testing. Encyclopedia of machine learning and data mining, 7(8):922–929, 2017.

[20] David Kolo Kolo and Solomon A Adepoju. A decision tree approach for predicting students academic performance. Modern Education and Computer Science Press, 2015.

[21] Igor Kononenko. Semi-naive Bayesian classifier. In Machine Learning—EWSL-91: European Working Session on Learning Porto, Portugal, March 6–8, 1991 Proceedings 5, pages 206–219. Springer, 1991.

[22] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In Proceedings of the 19th international conference on World wide web, pages 661–670, 2010.

[23] Zhengyi Liu, Song Shi, Quntao Duan, Wei Zhang, and Peng Zhao. Salient object detection for rgb-d image by single stream recurrent convolution neural network. Neurocomputing, 363:46–57, 2019.

[24] Alexander Maedche, Stefan Morana, Silvia Schacht, Dirk Werth, and Julian Krumeich. Advanced user assistance systems. Business & Information Systems Engineering, 58:367–370, 2016.

[25] Hai Thanh Nguyen and Anders Kofod-Petersen. Using multi-armed bandit to solve cold-start problems in recommender systems at telco. In Mining Intelligence and Knowledge Exploration: Second International Conference, MIKE 2014, Cork, Ireland, December 10-12, 2014. Proceedings, pages 21–30. Springer, 2014.

[26] U.S. Department of Homeland Security. Geographic regions. `https://www.dhs.gov/geographic-regions`.

[27] Ginés Lifante Pedrola. Beam propagation method for design of optical waveguide devices. John Wiley & Sons, 2015.

[28] Russell Rumbaugh. The FY22 defense space budget request analysis. The Aerospace Corporation, 2021.

[29] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In Learning for Text Categorization: Papers from the 1998 workshop, volume 62, pages 98–105. Citeseer, 1998.

[30] Jürgen Schmidhuber. Deep learning in neural networks: An overview. Neural networks, 61:85–117, 2015.

[31] Aleksandrs Slivkins et al. Introduction to multi-armed bandits. Foundations and Trends® in Machine Learning, 12(1-2):1–286, 2019.

[32] Brent Smith and Greg Linden. Two decades of recommender systems at Amazon.com. IEEE Internet Computing, 21(3):12–18, 2017.

[33] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.

[34] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. Chemometrics and intelligent laboratory systems, 39(1):43–62, 1997.

[35] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika, 25(3-4):285–294, 1933.

[36] Chih-Chun Wang, Sanjeev R Kulkarni, and H Vincent Poor. Bandit problems with side observations. IEEE Transactions on Automatic Control, 50(3):338–355, 2005.

[37] Ling Zhang and Bo Zhang. A geometrical representation of McCulloch-Pitts neural model and its applications. IEEE Transactions on Neural Networks, 10(4):925–929, 1999.

[38] Fei Zheng and Geoffrey I Webb. Semi-naive Bayesian classification. Journal of Machine Learning Research, 2008.

Appendix A

System-level Workflow Algorithms

Algorithm 1 Initialize User Session

GIVEN: u^O, u^G, Session ID
if New Session ID **then**
$w^I \leftarrow 0$
$w^O \leftarrow u^O$
$w^G \leftarrow u^G$
$w^{f(\vec{x})} \leftarrow [0, 0, 0]$
$\vec{\mu}_\theta \leftarrow [w^I, w_1^O, ..., w_{N_O}^O, w_1^G, ..., w_{N_G}^G, w_1^{f(\vec{x})}, ..., w_n^{f(\vec{x})}]^T$
$\sigma^I \leftarrow 3$
$m \leftarrow u^O == 0$
$\sigma^O(m) \leftarrow 3$
$\sigma^O(!m) \leftarrow 1$
$m \leftarrow u^G == 0$
$\sigma^G(m) \leftarrow 3$
$\sigma^G(!m) \leftarrow 1$
$\sigma^{f(\vec{x})} \leftarrow [3, 3, 3]$
$\Sigma_\theta \leftarrow \text{diag}([\sigma^I, \sigma_1^O, ..., \sigma_{N_O}^O, \sigma_1^G, ..., \sigma_{N_G}^G, \sigma_1^{f(\vec{x})}, ..., \sigma_n^{f(\vec{x})}])$
$p(\vec{\theta}) \sim \mathcal{N}(\vec{\mu}_\theta, \Sigma_\theta)$
else
Use $p(\vec{\theta})$ from previous login
end if

Algorithm 1 provides a detailed description of initializing a user session. Important notes are outlined below:

- If the session is new (i.e. the user has changed their initial inputs):
  - It is assumed that no prior information is available regarding the logistic sigmoid

intercept, or user feature preferences. For this reason, w^I and $w^{f(\vec{x})}$ are set to zero.

- User preference inputs are used to assign prior means to the weights of object types and geographic locations such that $w^O = u^O$ and $w^G = u^G$.
- The covariance is assigned as a diagonal matrix due to lack of information regarding covariance terms.
- σ^I and $\sigma^{f(\vec{x})}$ are set to 3 to indicate lack of prior information.
- Individual values for σ^O and σ^G are set to 1 if the user input "Disinterested" or "Interested" and 3 otherwise (an assumption is made that a user is more certain about strong views on a feature than they are about inputting "No Preference").

- Otherwise:
  - Use previously learned $p(\vec{\theta})$.

Algorithm 2 Human Interest Learning & Inference

GIVEN: t, $p(\vec{\theta})$
while Session ID active **do**
  if New track labeled for interest **then**
    for All newly labeled tracks **do**
      $p(\vec{\theta} \mid I, \vec{x}, G) \leftarrow \mathcal{N}(\vec{\theta}^*, H^{-1})$
      $p(\vec{\theta}) \leftarrow p(\vec{\theta} \mid I, \vec{x}, G)$
    end for
  end if
  if New active track or time elapsed $> t$ **then**
    for All active tracks **do**
      $p(I = 1 \mid \vec{x}, G, \vec{\theta}) = \frac{\sum_O p(O \mid \vec{x}) p(G \mid O) p(I{=}1 \mid \vec{x}, O, G, \vec{\theta})}{\sum_O p(O \mid \vec{x}) p(G \mid O) p(I{=}1 \mid \vec{x}, O, G, \vec{\theta}) + \sum_O p(O \mid \vec{x}) p(G \mid O) p(I{=}0 \mid \vec{x}, O, G, \vec{\theta})}$
    end for
  end if
  Sort tracks by interest
  Pause for time t
end while

Algorithm 2 provides a detailed description of the online functionality of the work described in this thesis. Important notes are outlined below:

- If a new track has been labeled for interest:
  - For all newly labeled tracks, calculate the conditional posterior of $\vec{\theta}$ using Equation 3.23 and set the new prior as the calculated posterior.
- If a new active track exists or some specified time has elapsed:
  - Infer interest in each active track.
- Sort all active tracks by interest.