Prediction And Anomaly Detection In Water Quality With Explainable Hierarchical Learning Through Parameter Sharing

by

Ali Mohammad Mehr

B.Sc., Sharif University of Technology, 2018

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

September 2020

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Prediction And Anomaly Detection In Water Quality With Explainable Hierarchical Learning Through Parameter Sharing

submitted by **Ali Mohammad Mehr** in partial fulfillment of the requirements for the degree of **Master of Science** in **Computer Science**

Examining Committee:

David Poole, Computer Science
Supervisor

Giuseppe Carenini, Computer Science
Supervisor Committee Member

Abstract

Decisions made on water quality have high implications for diverse industries and general population. In a 2020 study, Guo et al. report that the current literature on modeling spatiotemporal variabilities in surface water quality at large scales across multiple catchments is very poor. In this thesis, we introduce a simple, explainable, and transparent machine learning model that is derived from linear regression with hierarchical features for efficient prediction and for anomaly detection on large scale spatiotemporal datasets. Our model learns offsets for various features in the dataset while utilizing a hierarchy among the features. These offsets can enable generalization and be used in anomaly detection. We show some interesting theoretical results on such hierarchical models. We built a water pollution platform for exploratory data analysis of water quality data in large scales. We evaluate the predictions of our model on the Waterbase - Water Quality dataset by the European Environmental Agency. We also investigate the explainability of our model. Finally, we investigate the performance of our model in classification tasks while analyzing its ability to do regularization and smoothing as the number of observations grows in the dataset.

Lay Summary

Predicting water pollution in large catchment areas is a difficult task. Some versions of recent artificial intelligence models used to do prediction in water pollution are black box models in that the scientists provide some data to the model and the model outputs some predictions without any explanation provided. We introduce an explainable model to make predictions about, and detect anomalies in water pollution data. Our model is built on linear regression, one of the simplest prediction methods studied since 19th century; however we exploit hierarchical structures among the features. We built a water pollution platform for exploratory data analysis on large scale water pollution data. We evaluate the performance of our model compared to other models which can be used in such prediction tasks.

Preface

This thesis is based on a project done with Wan Shing Martin Wang under supervision of Dr. David Poole. The ideas for the parameter sharing model and its variants were developed by Dr. David Poole. The initial tests on the model were done by the author and Martin Wang. Theorem 2.1 was developed in collaboration by Dr. David Poole, Wan Shing Martin Wang, and the author. Theorem 2.2 was developed by the author. The implementation of the models on Waterbase - Water Quality dataset were done by the author. Water pollution platform was designed with the help of Minerva Intelligence Inc., especially Clinton Smyth and Jake McGregor, and it was implemented by the author. Anomaly detection using the hierarchical parameter sharing model was conjectured by Dr. David Poole, improved by Dr. David Poole with the help of the author, and implemented by the author. Chapter 4 was conducted by the author. Wan Shing Martin Wang evaluated variants of hierarchical parameter sharing model on movie recommendation task, especially cold start problem on Movielens dataset and on fashion datasets, specifically the ModCloth and Renttherunway datasets. Writing of the material, creation of the figures, and all code implementations in this thesis were conducted by the author.

A briefer version of chapters 2 and 3 will be submitted to a 2021 conference. This paper also includes Wan Shing Martin Wang's work on movie recommendation task.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to express my gratitude to those that have helped me throughout this journey. My sincere gratitude goes to my supervisor, David Poole, who guided me at every step of the way. With his patience and invaluable expertise, he supported me in developing an understanding of artificial intelligence. My thanks goes to Minerva Intelligence Inc., specially Clinton Smyth and Jake McGregor who advised me and inspired me throughout this project. Next in line is my friendly colleague, Martin Wang, who helped me develop this project.

I would like to thank my parents, who have supported and encouraged me all my life. It is thanks to them that I was able to take this incredible path towards my dreams. I would also like to thank my dear sister who has always listened to what I had to say.

This research is supported by Minerva Intelligence Inc., MITACS and NSERC; and it was enabled in part by support provided by WestGrid (www.westgrid.ca) and Compute Canada (www.computecanada.ca).

Chapter 1

Introduction

Despite the outstanding momentum that machine learning has seen in the recent years, the entire community stands in front of the barrier of explainability[1]. With the increasing use of AI in diverse fields, the implications of decisions made based on its AI are increasingly growing. This has led to more concerns regarding potential bias in machine learning. Such concerns regarding AI stretch to ethical domains as well. Model explainability and interpretability can improve public trust in AI[9].

Water quality in lakes and the rivers can be an important issue in economy, public health, and biological variability of key natural resources[25]. Rivers are shared among diverse industries including tourism, fishing, farming, steelworks, and maritime transport. Such a diversity causes decisions regarding water quality to have huge impacts. Note that decisions regarding water quality are mainly made in government levels with multiple stakeholders and are sometimes entangled with politics. Therefore, transparency and explainibility are important factors for an AI model that wants to be trusted in water quality tasks.

When a water contamination event occurs, it is important to detect and warn of such events. Extrapolation of water quality datasets into the future can potentially help early warning systems for water quality. Although the use of automatic sensors for water quality is increasing, water quality monitoring in major parts of the world is still done through manual sampling of water[6] which is analyzed in laboratories. The resulting data of such monitoring is usually sparse in space and time[16] sampled irregularly in time with high correlations in space. This means that many recent machine learning models which need temporally regular data or make stationarity assumption in space cannot be directly applied on surface water quality datasets.

There is currently a lack of capacity to model spatiotemporal variabilities in surface water quality at large scales across multiple catchments[10]. The objective of this research is to find a machine learning model for prediction and anomaly detection tasks in such large scale surface water quality datasets. A model that is able to learn the spatiotemporal variability of surface water pollution in large regions and in long periods of time can potentially be informative in large scale catchment management and policy making[10]. Our model is designed to be simple and explainable while also eliminating the need for preprocessing of noisy data. The model can also deal with missing feature values in observations.

Our model works based on commonalities and deviations in the dataset. It first extracts the com-

monality among all the data points. Then it identifies how groups of related observations deviate from the extracted commonality in a hierarchical manner. This method can help us explain the data in terms of commonalities plus deviations. Anomalies can be extracted as the deviations with high values or observations with high deviation from the model's prediction. We develop some theorems about how the deviations interact with each other. We analyze the performance of the proposed model in a regression task using a water quality dataset and in classification tasks using multiple real-world and synthetic datasets.

1.1 Problem Definition and Scope

We are mainly interested in supervised prediction tasks or unsupervised anomaly detection tasks on a dependant random variable y with n measurements which depends on multiple feature variables X. Examples of such tasks include prediction and anomaly detection on water pollution datasets where concentrations y of multiple chemicals are dependent on the time and location of the measurements, X. The dataset can have missing feature values in X.

For our model to have explainable results, we also exploit any hierarchy structure among the features. An example of a hierarchical structure among features is the parent-child relationship one can imagine between the set of the instances measured in January 2019 and the set of instances measured in 2019. In this example, the set of instances measured in January 2019 is a subset of the set of instances measured in 2019.

Our model can only work with discrete feature variables, X. In case of datasets with continuous features, we can discretize the features either using expert knowledge, e.g. discretizing time by month, or we can use existing discretization methods, e.g. methods introduced by Liu et al. [18] such as binning with equal frequency.

1.2 Literature review

There are two groups of methods that closely relate to our method: the multiple linear regression model, and matrix multiplication. The baseline form of our model is a special case of a multiple linear regression model [14]: A multiple linear regression model can be constructed with discrete features that makes the same predictions as our model. From another point of view, our model is similar to matrix factorization models. Koren et al. [13] motivate the matrix factorization model in terms of an average and learned offsets. A matrix factorization model with some fixed features can be designed to have the same predictions as our model.

The closest body of work to our model in water pollution task is the work done by Guo et al. [10]. Guo et al. [10] report that most of existing studies on surface water pollution either study the spacial variations of time-aggregated water quality data, e.g. Tramblay et al. [24], or use regression for prediction of water pollution from some other features in a single location, e.g. Kisi and Parmar [12]. In a survey done by Tiyasha et al. [23], the authors reviewed more than 200 research articles which have addressed the river water quality modelling using machine learning models to predict some water

quality feature directly from other water quality features or contaminants.

There are diverse models that are applied in spatiotemporal tasks. Jin et al. [11] apply Bayesian spatiotemporal models on air pollution datasets. Blangiardo and Cameletti [4] train Generalized Linear Models (GLM) on spatiotemporal datasets. For predicting particulate matter (PM_{10}) or rain fall, they use stochastic partial differential equation (SPDE) approach [17] to model spacial effects and random walk to model time effects. They also model different types of interactions between the spatial and temporal effects. When using SPDE for spatial effects, it is assumed that the covariance between every two points in space is only dependant on the distance between the points. This type of stationarity is not applicable for surface water pollution because the covariance is high along a water stream while it is low between streams that have not joined. Blangiardo and Cameletti [4] also use conditional autoregressive (CAR) models to account for spatial effects in disease mapping. A similar approach might be suitable for surface water datasets because of its ability to model spacial dependencies using a graph. Yet, the method they use only models bidirectional spacial dependencies whereas in rivers water flows only in one direction. We compare the performance of our model with this approach.

Ruybal et al. [22] use spatiotemporal regression kriging for groundwater-level predictions, which they recommend for datasets that do not contain consistent spatially located data over all relevant temporal periods. Note that kriging is not really applicable to surface water datasets where the water flows in streams.

Our model utilizes a hierarchy of features to learn explainable offsets. Taxonomies and Ontologies can be used in structuring of such hierarchy among features.

1.3 Thesis Organization

In Chapter 2, we introduce our model and develop some theory for it. In Chapter 3, we introduce a platform we built for exploratory data analysis on a large-scale water pollution dataset and evaluate our model on that dataset. In Chapter 4, we investigate our model in binary classification tasks.

Chapter 2

Parameter Sharing Model

We assume we have a number of discrete features, e.g. month, year, location. An instance is an assignment to some of the features. An observation is an instance whose value is known. Suppose we have a dataset of n observations $y_1, \ldots, y_i, \ldots, y_n$, for example, measurements of Phosphate in river waters. A class is a set of instances. There are two types of classes: One type is described by a boolean combination of features (the set of instances for which the formula is true) and the other type is a singleton class that contains a single observation. For example, all instances measured in 2017, all instances measured in May, all instances measured in May 2017, and all instances measured in station s_1 in May are four classes in each of which we might have multiple observations. We assume we have $m+n+1$ classes in total ($C_0, \ldots, C_{m+n}$): a universal class that includes all instances (C_0), m classes defined as boolean combination of features ($C_1, \ldots, C_m$), and n singleton classes for each observation ($C_{m+1}, \ldots, C_{m+n}$). A *parameter sharing model* assumes the following:

- **Offsets for classes:** There exists one parameter for each of the $m+n+1$ classes. σ_j is the offset for class C_j. Note that we use the words parameter and offset interchangeably in this context.
- **Model constraint:** The value of each observation is equal to the sum of offsets of classes to which the observation belongs: $y_i = \sum_{i \in C_j} \sigma_j$, where σ_j are the classes which y_i belongs to.

The offsets for singleton classes are called singleton offsets or noise parameters. In the next section, we motivate this naming.

2.1 Hierarchical Parameter Sharing Model

A hierarchical representation of classes can be used to demonstrate how learning happens in a parameter sharing model. It also allows for a structured learning of offsets based on the hierarchies among their respective classes. In the hierarchy of classes, if class A is a subset of class B, meaning that all instances in class A exist in class B, class A is considered to be under class B in the hierarchy. This subset hierarchy is a subset lattice that can be represented as a directed acyclic graph(DAG) in which class A is a child, descendent, or subclass of class B if $A \subset B$. The DAG is constructed using only the classes for which we have assigned an offset.

Table 2.1: A summary of symbols and expressions in parameter sharing models

Symbol or Expression	Definition
n	Number of observations
$\{y_1, \ldots, y_i, \ldots, y_n\}$	Set of all observations
i	Index for observations: $1 \leq i \leq n$
$\{C_0, \ldots, C_j, \ldots, C_{m+n}\}$	Set of all classes
$\{C_{m+1}, \ldots, C_j, \ldots, C_{m+n}\}$	Set of all singleton classes (classes with a single observation)
$m+n+1$	Number of all classes: one universal class, m classes defined as boolean combination of features, and n singleton classes for each observation
j	Index for classes: $0 \leq j \leq m+n$
$\{\sigma_0, \ldots, \sigma_j, \ldots, \sigma_{m+n}\}$	Set of all offsets (or parameters) in a parameter sharing model
σ_0	Parameter for the universal class(universal parameter)
$\{\sigma_{m+1}, \ldots, \sigma_{m+i}, \ldots, \sigma_{m+n}\}$	Set of singleton offsets (noise parameters)
$y_i = \sum_{i \in C_j} \sigma_j$	Model constraint equation for observation: $i \in \{1, \ldots, n\}$

Every node in the DAG hierarchy is representative of a class and its associated offset. The singleton classes, which are leaves in the DAG, are representative of the noise parameters σ_{m+i} for each observation. Explainability is a main focus in parameter sharing model; therefore, the goal is to set σ_j such that they have meaningful values. In this thesis, the goal is to set the value of σ_j to be the deviation that best fits all of the observations.

Example 2.1. Suppose we have a dataset of n phosphate measurements. Measurement i is done in a specific year Y^i, month M^i, and location L^i. A parameter sharing model will define offsets $\{\sigma_0, \ldots, \sigma_{m+n}\}$ based on the different classes the observations can fall into. An example of a parameter sharing model for this dataset is as follows:

$$y_i = \sigma_0 + \sigma_{Y^i} + \sigma_{M^i} + \sigma_{L^i} + \sigma_{Y^i M^i} + \sigma_{Y^i L^i} + \sigma_{Y^i M^i L^i} + \sigma_{m+i}, \tag{2.1}$$

where y_i is the measurement of phosphate. σ_0 is the parameter for the class of all observations (universal class). We will see that the model will predict σ_0 for an instance for which we do not have any features. Since the most reasonable prediction for such an instance is the the mean of all the observations, σ_0 can represent universal mean. Note that parameter σ_0 is shared among all observations. σ_{Y^i} is the offset for year Y^i Note that offset σ_{Y^i} is shared between all observations done in year Y^i. Figure (2.1) shows these classes and their hierarchy in a DAG. $\sigma_{Y^i M^i}$ is the offset for the class of observations done in year Y^i and month M^i. σ_{m+i} is the offset for the singleton class of observation i and is unique to observation i, meaning it is not shared with any other observation. σ_{m+i} represents the noise in observation i which could not be explained using the classes that the observation is in. For example, σ_{m+i} might be a high positive value for a sample taken in a day when there was a spill of phosphate close by that caused a one-day increase in phosphate in that location. The existing classes which are based on year, month, and location of the measurement cannot model the one-day increase, so the one-day increase will be

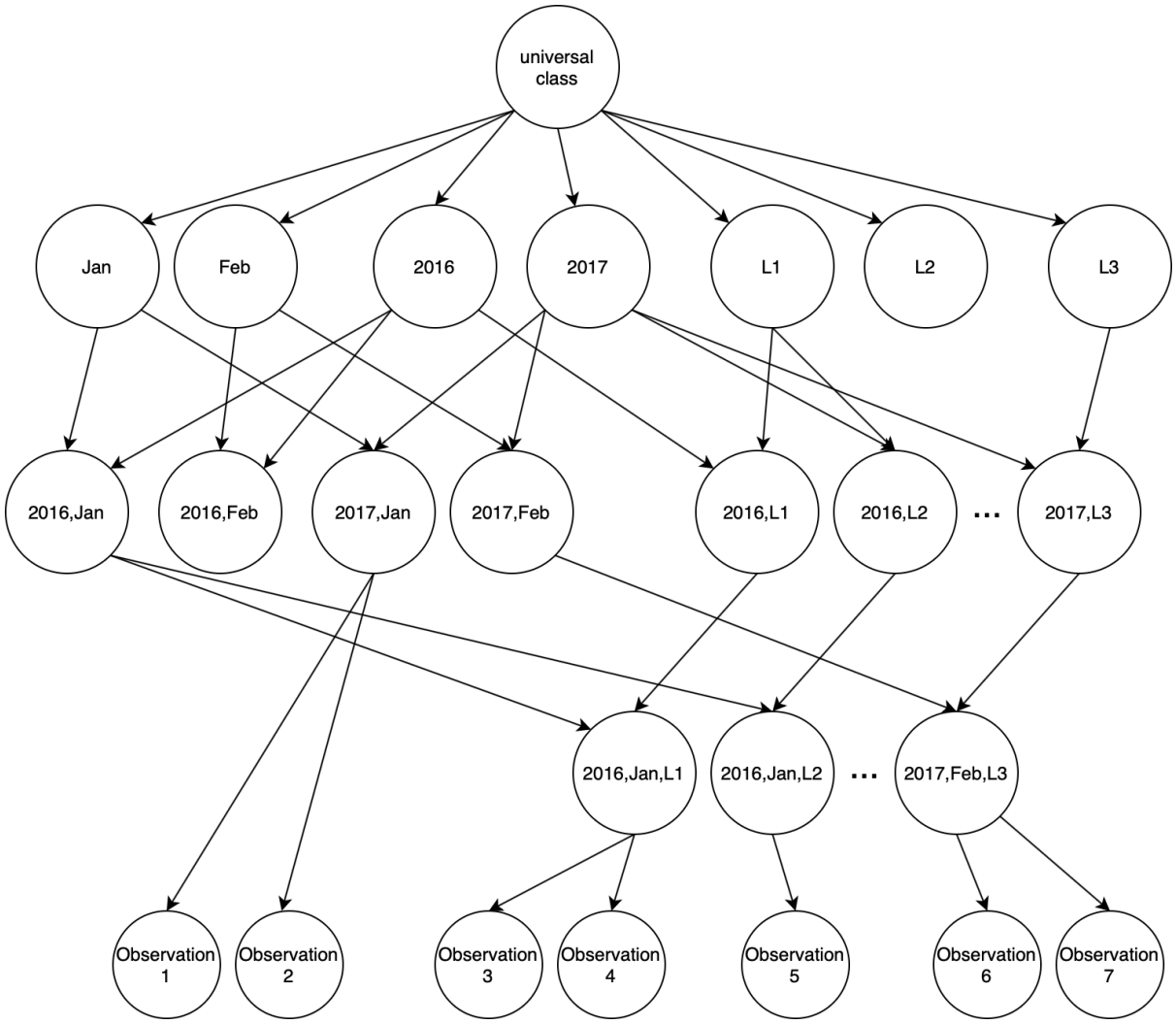


Figure 2.1: The hierarchical relationship between classes in example 2.1 is shown in a DAG. This figure assumes that all observations are done in 2016 or 2017, in January or February, and in 3 locations $\{L1, L2, L3\}$. 7 observations can be seen in this figure.

modeled in the noise parameter σ_{m+i}. Nonetheless, all offsets used for this observation will inevitably be influenced by this spill and they will experience a small increase.

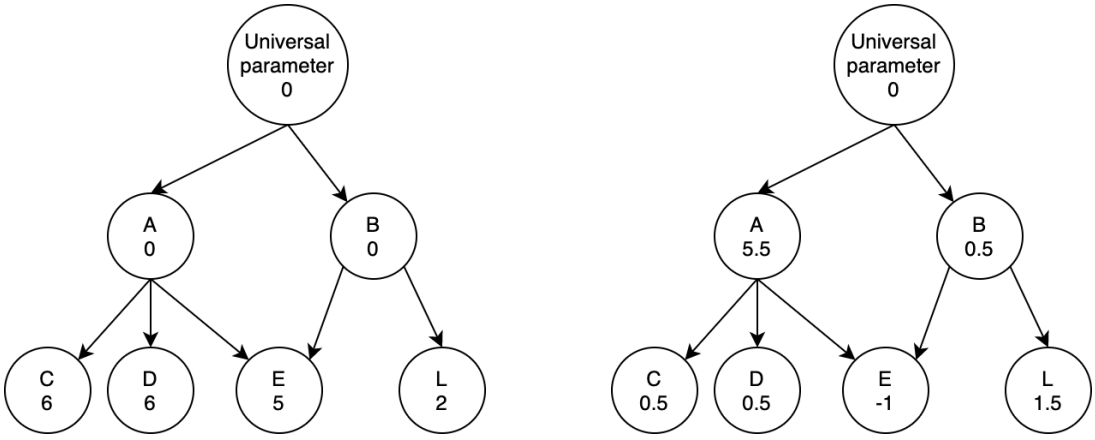
In Figure 2.1, a hierarchical relationship between classes is shown in a DAG when all observations are done in 2016 or 2017, in January or February, and in 3 locations $\{L1, L2, L3\}$. 7 observations can be seen in this figure. The structure of the DAG allows us to see what classes every observation belongs to. Observations y_3 and y_4 were measured in January 2016 in location L1. Observations y_1 and y_2 are only connected to the $2017, Jan$ class, which means the location of these measurements is missing in the dataset.

Note that in this example, we did not define any classes for observations done in month M^i and location L^i, which resulted in not having $\sigma_{M^i L^i}$ in Equation (2.1). This is allowed in parameter sharing model. Such decisions can be made using expert knowledge.

A parameter sharing model is realized in a DAG if the constraint for the parameter sharing model holds for all observations: $y_i = \sum_{i \in C_j} \sigma_j$ where σ_j are all the offsets reachable from the node for observation y_i in the reverse DAG.

There are infinitely many models that can explain the data in this fashion. For instance, in example 2.1, if we have high observations in July 2017, it is not trivial what part of the high values should be explained through σ_{July}, σ_{2017}, or $\sigma_{2017,July}$. Therefore, we need bias and regularization that goes beyond the data to be able to learn the parameters. For instance, in the previous example, a naive models is to set $\sigma_{m+i} = y_i$ and set all other offsets to zero. This would mean that every observation is simply some unexplained noise.

Learning in a hierarchical parameter sharing model happens when we explain the noise σ_{m+i} using other offsets. This can happen through regularization. The following is an example of how learning can happen.



(a) All data is noise **(b)** Some data is pushed up

Figure 2.2: A simple DAG model on a small dataset with four observations: 6, 6, 5, and 2. In (a) all the observations are explained as noise using the singleton offsets. Based on (a) we can see that three of the observations are in class A and two of the observations are in class B. Classes A and B share an observation. In (b) a value of 5 is pushed up to A and a value of 0.5 is pushed up to B.

Example 2.2. In Figure 2.2, we can see an example of a DAG with 6, 6, 5, and 2. The first three observations are in class A, and the last two observations are in class B. Classes A and B share an observation. In Figure 2.2a all the observations are explained as noise using the singleton offsets. In Figure 2.2b a value of 5 is pushed up to A and a value of 0.5 is pushed up to B.

2.2 Prediction and Anomaly Detection in Parameter Sharing Models

As seen above, the model is over-parameterized and its prediction for an observation is equal to the value of that observation. In case of an unobserved instance, the singleton offset (noise parameter) for that instance is assumed to be zero. With this assumption, the prediction of model for instance i, $\hat{y}_i$,

is defined to be the sum of offsets for all classes that the instance belongs to: $\hat{y}_i = \sum_{i \in C_j} \sigma_j$. Table 2.1 shows a summary of different symbols and expressions in a parameter sharing model.

Using a trained hierarchical parameter sharing model, we can do unsupervised anomaly detection in the following three fashions:

- **Class parameters with large absolute values:** If a class offset has a large absolute value relative to other class offsets, it means that knowing that a observation belongs to this class raises the alarm that this sample will have an anomalously large or small value. As an example, if we have a class of observations done in summer for Phosphate, we might observe that the offset for summer is high. This type of anomaly does not necessarily suggest an alarming event. For example, the high offset for summer could be due to the fact that farming activity in summer causes an increase in Phosphate level.

- **Noise parameters with large absolute values:** If the noise parameter for an observation has a large absolute value, it can mean that an anomalous event led to a high or low value for that observation, but this event was not explained or captured by the classes we derived from the dataset. For instance, the noise parameter could be high for an observation of phosphate done close to a spill that resulted a daily increase in phosphate levels, but we did not have a class capturing this and the high value for the observation remained unexplained in the noise parameter.

- **Group of observations with similar noise parameters:** If a group of observations have similar values for their noise parameters, we might be able to define a new class which includes those observations. To learn a new class which includes such observations, we need extra information about the observations than the current classes derived from the dataset. For example, if some observations of Phosphate that were measured in locations close to each other have a similar noise parameter, we can make a new class for these observations spatially close to each other. Making a new class for these observations will cause the noise parameter in all these observations to decrease. Note that creation of this class would not be possible if we did not have information about spatial closeness of observation. This spatial closeness may not be previously captured through the existing classes.

The above three methods of anomaly detection look for anomalously large absolute values in the list of class or noise parameters. Since largeness is relative, we can sort the parameters based on absolute value and look for anomalies among the first parameters in the sorted list.

Example 2.3. Figure 2.3 shows hierarchical parameter sharing model represented using a DAG. The leaves of the DAG model correspond to five singleton offsets for five observations (at various times and locations) of Phosphate: $1.5mg/L$, $3mg/L$, $2.5mg/L$, $2mg/L$, and $0.5mg/L$. Sample 5 is from March 2018 in region B, and therefore it is under these two classes. At every node, the value of the offset for that class is written. For example, the offset for region B (which includes samples 4 and 5) is $-0.62mg/L$. The prediction for any unobserved instance is the sum of the offsets of the classes that the instance belongs to. For example, the observed value for observation 5 is the sum of all offsets

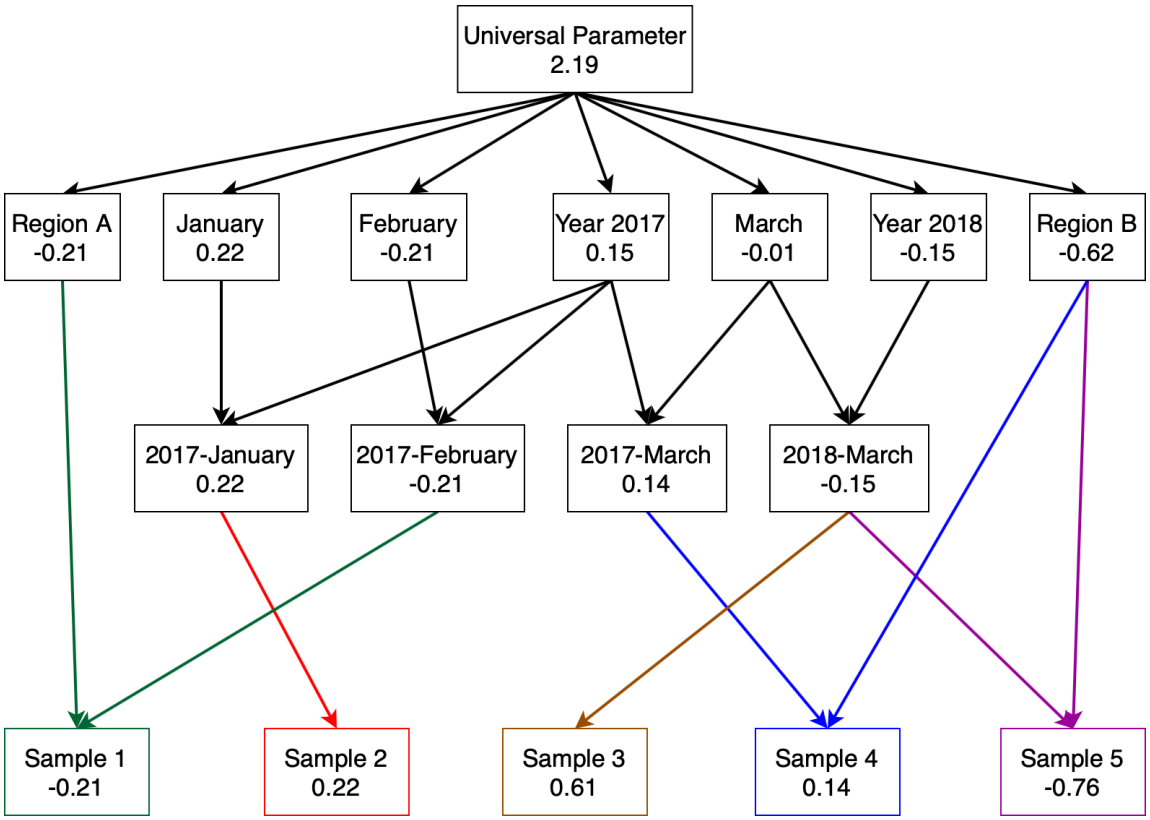


Figure 2.3: An example of a hierarchical parameter sharing model modeling a chemical observations dataset. The observations for observations 1 to 5 are: $1.5mg/L$, $3mg/L$, $2.5mg/L$, $2mg/L$, and $0.5mg/L$. The vertices are offsets and all values are in mg/L. The observations and the edges touching them are colored.

for sample 5: noise parameter for sample 5, 2018-March, 2018, March, region B, and the universal parameter: $-0.76-0.15-0.15-0.01-0.62+2.19=0.5mg/L$.

The model can also predict values for unobserved instances. As an example, although the model in Figure 2.3, has not seen any observation from region B in February 2017, it can give a prediction for such instance as $-0.21-0.21+0.15-0.62+2.19=1.3mg/L$

2.3 Tree DAGs as a Simple Structure of DAG Hierarchy

In general, the DAG representing the subset relationship among classes can be any subset lattice, but this thesis considers tree DAGs as a special case of subset lattices and builds up the analysis of the properties of hierarchical parameter sharing models by first studying their workings on tree DAGs. A DAG is considered to be a tree when each except the top node has one parent. A tree DAG is simpler than a general DAG because in a tree DAG moving the information up to the parents is simpler; every node has a single parent, so all the commonality between the siblings has to be explained by a single parent. A general DAG is more complex in that it is not trivial how much of the commonality among siblings has to be explained by each of theirs parents.

For instance, one of the properties of a tree DAG is that starting from any initial state, for the model constraints to hold, all the siblings have to change by the same value.

2.4 Learning Hierarchical Parameter Sharing Models

In the following sections, different methods for learning the σ_j parameters will be explored.

2.4.1 L2-Regularized Baseline Parameter Sharing Model

To the learn offsets in an L_2-Regularized baseline parameter sharing model, we minimize the L_2 norm of all the offsets except the parameter for the universal class (universal parameter). In this context, L_2 norm of some elements is defined to be the square root of the sum of the squares of the values. Note that the model constraints ($y_i = \sum_{i \in C_j} \sigma_j$) have to hold while we minimize the L_2-norm of offsets. In other words, to learn an L_2-regularized baseline parameter sharing model, we minimize the following loss function with the condition that the model constraints are not violated. In the following, as mentioned in table 2.1, σ_0 is the universal parameter, which is not regularized because the universal average can be any value and there is no reason to believe it is close to zero:

$$Loss = \sum_{j \in \{1,\ldots,m+n\}} (\sigma_j)^2 \tag{2.2}$$

One way of looking at this is that we start with all non-singleton offsets to be zero, and singleton parameters to be the observed value. In every layer of the hierarchy if sum squared of children is greater than that of their parents, then the model will push the signal from children to the parent by subtracting a value from children and adding some value to the parents. In the end, the mean squared of the children will be closer to zero while they are pushed to their parents because this is a state with smaller L_2-norm. It is possible to add an informed prior by regularizing the offsets towards a default value instead of zero, especially if the dataset has very few number of observations.

We do not regularize the universal parameter because the universal parameter is supposed to capture the universal mean in the dataset, which can be any value. All other offsets are regularized. In Equation (2.2), we can separate the sum between σ_k^i and observation parameters. Then, we can use the model constraint to reach a different formula for the loss:

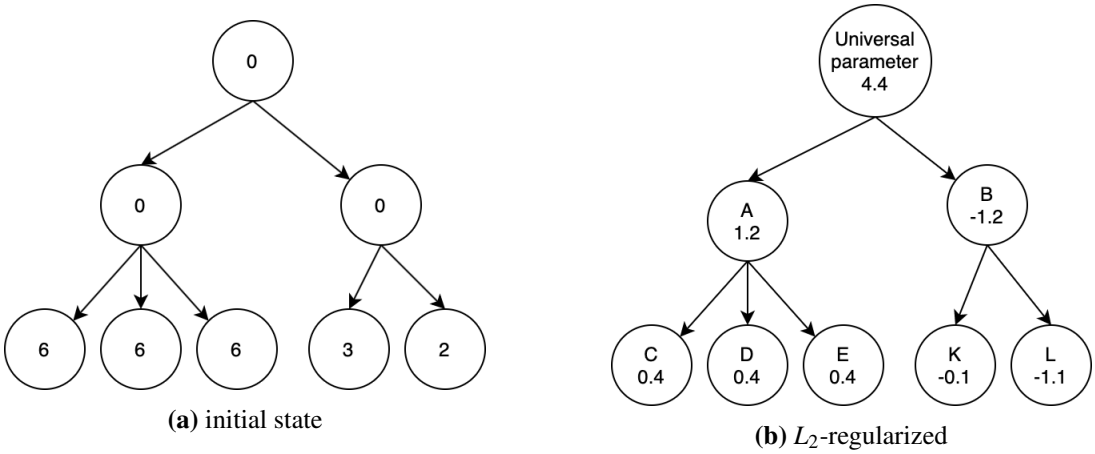
$$\begin{aligned} Loss &= \sum_{j \in \{1,\ldots,m+n\}} (\sigma_j)^2 && (2.3)\\ &= \Big(\sum_{i \in \{1,\ldots,n\}} (\sigma_{m+i})^2\Big) + \Big(\sum_{j \in \{1,\ldots,m\}} (\sigma_j)^2\Big) && (2.4)\\ &= \Big(\sum_{i \in \{1,\ldots,n\}} (y_i - \sum_{i \in C_k, k \neq m+i} \sigma_k)^2\Big) + \Big(\sum_{j \in \{1,\ldots,m\}} (\sigma_j)^2\Big) && (2.5) \end{aligned}$$

In Equation (2.5), the first term is the error term denoting the difference between predictions of parameter sharing model(excluding the noise parameter) and the observed value while the second term

is the sum of the class offsets squared - except the universal parameter and noise parameters. This equation can be used to learn an L_2-regularized baseline parameter sharing model on a dataset. This shows that this model is equivalent to an L_2-regularized linear regression model $y_i = \sigma_0 + XB$, where the universal parameter σ_0 is the intercept and doesn't get regularized. The class offsets $\{\sigma_1, \ldots, \sigma_m\}$ are the offsets B in the linear regression model which are regularized, and the independent variables X are 0 or 1 denoting the classes that the observation belongs to.

Model Properties

To get more insight on how this model learns the class offsets in a tree, we can assume that we initialize all class offsets to zero except the noise parameters σ_{m+i} which will be set to $\sigma_{m+i} = y_i$. Starting from this initial state and using some iterative method that converges to the global minimum in Equation (2.5), we can get insight on how parents are learned from the values of their children in a tree. The following example, illustrates this.



(a) initial state

(b) L_2-regularized

Figure 2.4: A simple tree DAG model on a simple dataset with five observations. Based on (a) we can see that three of the observations have a value of 6, and the other two observations have values of 3 and 2.

Example 2.4. Figure 2.4a demonstrates the initial state of a hierarchical parameter sharing model as described above, and Figure 2.4b shows the L_2-regularized version of the model where the loss is minimized. The prediction of model for an instance in class A is $4.4 + 1.1 = 5.5$ while all the observations in A are 6. This is due to the effects of regularization which causes the predictions of siblings A and B to be closer to each other. The prediction of model for an instance in class B is $4.4 - 1.1 = 3.3$ while there were two observations of 3 and 2 in class B.

In the following two theorems, we analyze some properties of an L_2-regularized baseline parameter sharing model in trees and DAGs:

Theorem 2.1. In a trained L_2-regularized baseline parameter sharing model on a tree hierarchy, at every layer except the top layer (with universal parameter), the sum of the children is equal to the parent.

Proof. Consider a parameter sharing model with the minimum loss defined in Equation (2.2) in which there is a parent with value p with c children with values $v_1, \ldots, v_c$, where $S = \sum_{t=1}^{c} v_t$. Assume $\sum_{t=1}^{c} v_t \neq p$. This means:

$$\varepsilon = \frac{p - \sum_{t=1}^{c} v_t}{c+1} \neq 0 \tag{2.6}$$

The argument is that if we add ε to all the children and subtract ε from the parent, we will arrive at a lower loss while the model constraints are still held. Since through this activity only the values of p and v_t will change, we will only look at the part of loss function that is composed of these offsets:

$$loss_{old} = (p)^2 + \sum_{t=1}^{c} (v_t)^2 \tag{2.7}$$

$$loss_{new} = (p-\varepsilon)^2 + \sum_{t=1}^{c} (v_t + \varepsilon)^2 \tag{2.8}$$

$$= (p-\varepsilon)^2 + \sum_{t=1}^{c} \left((v_t)^2 + (\varepsilon)^2 + 2v_t\varepsilon \right) \tag{2.9}$$

$$= (p)^2 + (\varepsilon)^2 - 2p\varepsilon + c(\varepsilon)^2 + 2\varepsilon(\sum_{t=1}^{c} v_t) + \sum_{t=1}^{c} (v_t)^2 \tag{2.10}$$

$$= (p)^2 + \sum_{t=1}^{c} (v_t)^2 + (c+1)(\varepsilon)^2 - 2\varepsilon(p - \sum_{t=1}^{c} v_t) \tag{2.11}$$

$$= (p)^2 + \sum_{t=1}^{c} (v_t)^2 + (c+1)(\varepsilon)^2 - 2\varepsilon(c+1)\varepsilon \tag{2.12}$$

$$= (p)^2 + \sum_{t=1}^{c} (v_t)^2 - (c+1)(\varepsilon)^2 < loss_{old} \tag{2.13}$$

□

Notice in Figure 2.4b, offset B is equal to the sum of its children; this holds in general.

Theorem 2.2. In an arbitrary DAG, if there exists a set of disjoint classes $\{C_1, \ldots, C_l\}$ with offsets $\{\sigma_1, \ldots, \sigma_l\}$ whose union is all the observations(i.e, the classes are a partition of all of the data), then in the L_2-regularized baseline parameter sharing model, the following holds: $\sum_{t=1}^{l} \sigma_t = 0$

Proof. Assume we have a solution with minimum loss, but $\sum_{t=1}^{l} \sigma_t \neq 0$. This means that ζ, the average of σ_t, is not zero. We show that if we subtract ζ from each σ_t and add ζ to the universal parameter, the loss defined in Equation (2.2) will decrease. The model constraints will still hold because for every observation the increase in universal parameter has been compensated for by the decrease in σ_t. Note that universal parameter does not appear in loss expression. We will only look at the part of the loss function that includes σ_t offsets

$$loss_{old} = \sum_{t=1}^{l} (\sigma_t)^2 \tag{2.14}$$

$$loss_{new} = \sum_{t=1}^{l} (\sigma_t - \zeta)^2 \tag{2.15}$$

$$= \sum_{t=1}^{l} (\sigma_t)^2 - 2\zeta \sum_{t=1}^{l} \sigma_t + l\zeta^2 \tag{2.16}$$

$$= \sum_{t=1}^{l} (\sigma_t)^2 - 2\zeta l\zeta + l\zeta^2 = \sum_{t=1}^{l} (\sigma_t)^2 - l\zeta^2 < loss_{old} \tag{2.17}$$

□

Note that the above case happens frequently in real datasets. For example, in Figure 2.3, which is already L_2 minimized (and rounded to 2 decimal places) you can see that the sum of the offsets for January, February, and March; the sum of offsets for years 2017 and 2018; and the sum of offsets for 2017-January, 2017-February, 2017-March, and 2018-January are 0 because all of them partition the observations. Even the sum of 2017 and 2018-March or the sum of 2017-March, 2018-March, Sample 1, and sample 2 are zero.

2.4.2 Top-Down Parameter Sharing Model

An extension to L_2-regularized baseline parameter sharing model would be to learn the L_2-regularized baseline model layer by layer from top to bottom until the noise parameters are learned. This allows training of each class offset to have direct access to observation residuals under that class. In addition, this allows explicit use of a hierarchy during training. For example, the parents, siblings, and children of a node can explicitly be used to learn it.

To learn a top-down parameter sharing model, first we initialize all offsets to be zero. Then, we go down from top of the hierarchy (universal parameter) to the leaves, layer by layer. At every layer, we will learn the offsets in that layer while keeping offsets on other layers fixed. At every layer, we minimize the L_2-regularized mean squared error of prediction of observation. In other words, assuming offsets $\sigma_{l,t}$ are the offsets in layer l of the hierarchy ($1 \leq t \leq c_l$), at layer l we minimize the following loss function:

$$\min_{\sigma_{l,1},\ldots,\sigma_{l,c_l}} \Big(\sum_{i \in \{1,\ldots,n\}} (y_i - \sum_{i \in C_k, k \neq m+i} \sigma_k)^2 \Big) + \lambda \sum_{t=1}^{c_l} (\sigma_{l,t})^2 \tag{2.18}$$

where λ is the regularization rate. Note that only the parameters in layer l are minimized in above. When layers 1 to l are learned, parameters in layers $> l$ are zero.

Model Properties

In a tree hierarchy, assume the observations under class C are $o_1, \ldots, o_c$ and the parent of C is P and the sum of all the ancestors of C (i.e. prediction for the class of P) is $\hat{P}$. Since the hierarchy is a tree, the

parameter for C, σ_C, will be calculated by minimizing the part of loss function relevant to σ_C:

$$\sigma_C = \arg\min_{\sigma_C}(\sum_{t=1}^{c}(\hat{P}+\sigma_C-o_t)^2)+\lambda(\sigma_C)^2 \tag{2.19}$$

$$\implies \frac{d}{d\sigma_C}\left((\sum_{t=1}^{c}(\hat{P}+\sigma_C-o_t)^2)+\lambda(\sigma_C)^2\right)=0 \tag{2.20}$$

$$\implies \sigma_C = \frac{(\sum_{t=1}^{c}(o_t-\hat{p}))}{c+\lambda} = \frac{(\sum_{t=1}^{c}o_t)-c\hat{p}}{c+\lambda} = \frac{(\sum_{t=1}^{c}o_t)+\lambda\hat{p}}{c+\lambda}-\hat{p} \tag{2.21}$$

This means that the prediction for the class of C (i.e. $\hat{C}$) is:

$$\hat{C} = \hat{P}+\sigma_C = \frac{(\sum_{t=1}^{c}o_t)+\lambda\hat{P}}{c+\lambda} \tag{2.22}$$

This means that in a tree, the prediction for class of C, $\hat{C}$, is a weighted average of $\hat{P}$ (with weight λ) and the observations in C (with weight c).

2.5 Explainibility in parameter sharing models

Barredo Arrieta et al. [1] consider a model to be transparent if it is understandable by itself. They also give three levels of transparency for models, namely simulatability, decomposability and algorithmic transparency. Both versions of our model explained above are transparent: They are simulatable because the interactions between offsets are through summation and each offset corresponds to a well-defined class, e.g. offset for month; they are decomposable because all of the offsets have the same unit of measure as the observations; and they are algorithmically transparent because they use the sum of the relevant (shared) offsets to make predictions. This transparency can be utilized for explainability in the following fashions:

- **Observation explainability:** We can explain every observation and every prediction in terms of a linear sum of weights, all of which have a semantic meaning. For example, we can use the model to answer the question of whether an extraordinarily high Phosphate observation can be explained through the month it was sampled in or not.
- **Gestalt explainability:** We can explain the whole model in terms of anomalies, which are offsets with a high (positive or negative) value.

Chapter 3

Testing on Water-Quality Dataset

3.1 Water Quality Dataset

Waterbase - Water Quality is an open dataset [8] by the European environmental agency (EEA) which includes over 33 million pollutant readings all over Europe from 1985 to 2018. Each reading is measuring a chemical element, at a specific time, and at a specific water pollution monitoring station. Some stations monitor pollutants in ground water and some monitor surface water, e.g. rivers, lakes and etc. We are interested in surface water pollution. The dataset does not include how surface water monitoring stations are connected to each other by rivers. To determine this, we downloaded the river data from OpenStreetMap[19] and matched stations with rivers. A station is matched to a river if the station is less than 50 meters away from the thalweg of the river extracted from OpenStreetMap. With this data, we can determine which surface monitoring stations are upstream and downstream of each other. We filtered out stations that were not matched with any river (further than 50 meters from any river center line), assuming that they were ground water monitoring stations.

3.2 Water Pollution Platform

In order to analyze the water-quality dataset and get insight about the data, we built an exploratory data analysis platform for the dataset. This platform allows the user to visualize different aspects of data. The user can also use it to visualize the models trained on the dataset as well as to visualize anomalies detected on the dataset. The preprocessing of data for the platform is done using Python while a web-based front-end built with HTML and JavaScript accesses the processed data and generates the visualizations. The main page of the platforms allows the user to visualize the dataset along with some extra information extracted from dataset. Figure 3.1 shows an annotated image of this page. The main component of this page is the map pointed at by marker #1. The user can see surface water pollution stations as yellow dots in the map. Every measurement is done in a water pollution station. The rivers are also shown in blue lines. In the following, different functionalities available to the user in this page are introduced.

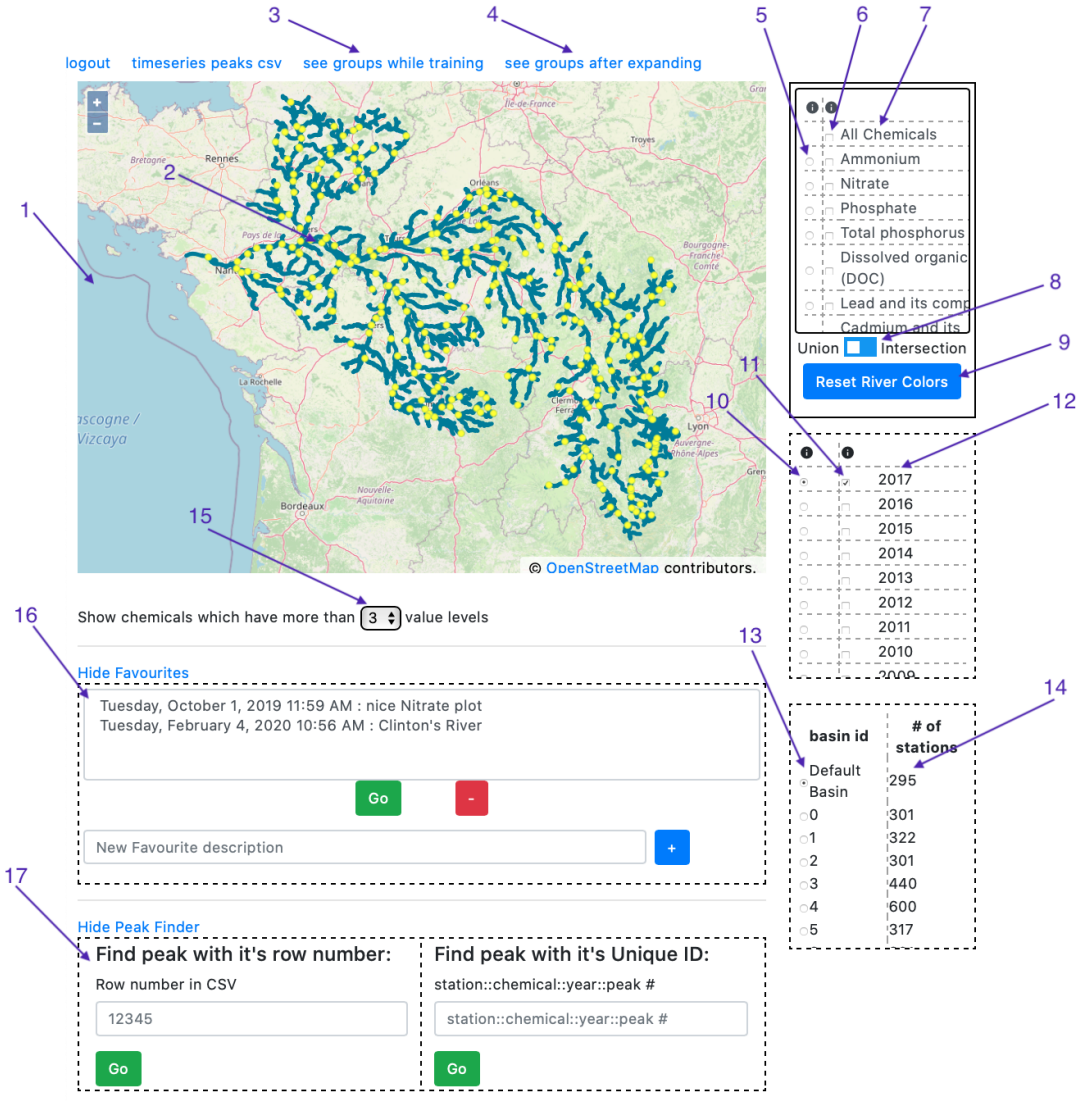


Figure 3.1: Annotated main page of the water pollution platform. This page is mainly used for exploratory data analysis.

3.2.1 Stations with Measurements

Since not all stations have measurements of all chemicals, the first step for a user interested in certain chemicals is for them to identify the stations that have measurements of those chemicals. The user can select a number of chemicals using check boxes in marker #6. They also need to select a number of years in check boxes in marker #11. Selecting chemicals will change the color of the stations that have measurements of those chemicals in the selected years into black. Figure 3.2 shows the map on the main page when check boxes for chemicals ”Nitrate” and ”Lead and its components” where selected in marker #6, and year 2017 was selected in marker #11. In that figure, the black stations are the stations that have measurements of ”Nitrate” or ”Lead and its components” in 2017.

Marker #8 shows a slider for selecting either ”union” or ”intersect.” The choice will be used in

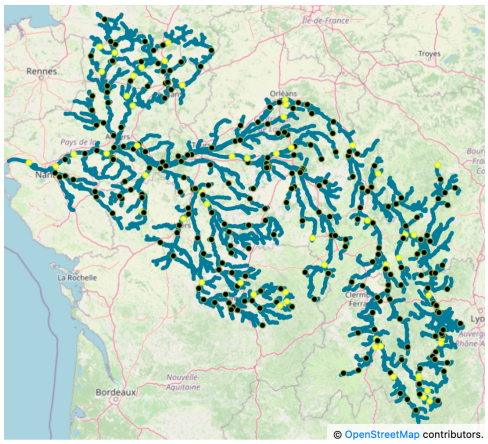


Figure 3.2: The map on the main page when check boxes for chemicals ”Nitrate” and ”Lead and its components” where selected in marker #6. The black stations are the stations that have measurements of ”Nitrate” or ”Lead and its components” in 2017. Note that ”union” was selected in marker #8 and 2017 was selected in marker #11. The yellow stations do not have any measurements of ”Nitrate” or ”Lead and its components” in 2017

coloring the stations to black. If union is selected, the stations that have measurements of *any* of the selected chemicals will turn to black. If intersect is selected, only the stations that have measurements of *all* the selected chemicals will turn black.

3.2.2 Pollutant Added to River Section

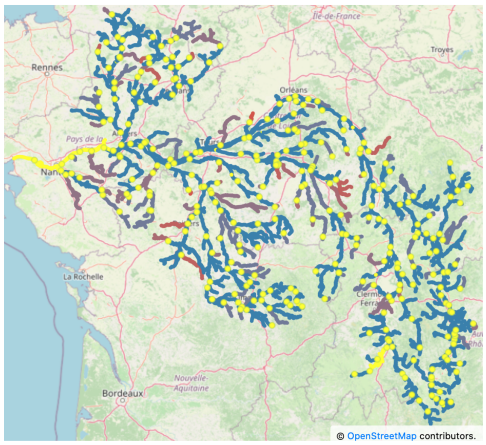


Figure 3.3: The map on the main page of water pollution platform, when Nitrate is selected in #5 and year 2015 is selected in #10. Red segments of the river show the segments where Nitrate was added substantially in 2015.

Radio buttons in marker #5 allow the user to select a single chemical element. Based on the chemical element selected in #5 and the year selected in #10, the river segments will be colored red to blue with red meaning that chemical was added to the water in that segment of the river more than the other segments of the rivers. Blue means that the selected chemical was not added substantially in that segment of the

river in that year. To do this, first we calculate the average concentration of every chemical for each year for each station. We do not have access to water discharge in rivers, and we only have access to chemical concentrations in water as amount per litre. In a segment of river with no confluence, the difference between downstream and upstream concentrations is a measure of how much chemical was added in that segment. For a river segment with confluences, to estimate amount of chemical added in a river segment with multiple upstream stations in different upstream branches and a single downstream station, we calculate downstream concentration minus maximum of upstream concentrations. If this value is above zero, it means that a nonzero amount of the chemical was added to the water in that segment. Figure 3.3 shows how the colors of segments of rivers change when Nitrate is selected in radios in #5 and the radio button for year 2015 is selected in #10. Note that it is not trivial to extract this information from the water-quality dataset since it does not have the data about which stations are connected with water. Only after matching the stations with their corresponding rivers could we extract this data.

The button in marker #9 allows the user to reset the colors of rivers to their original blue color.

3.2.3 Plots of Measurements in a Station

The user can select a water pollution station on the map by clicking on it. This causes the station to change color to red, and the closest stations upstream to the selected station get assigned different colors ranging from orange to green. In addition, at the bottom of the page, plots of measurements for all chemicals in the selected station appear. In Figure 3.4, we have selected the station pointed at by marker #2. The figure shows the map zoomed in on the selected station and the plots for two chemicals shown at the bottom of the page. The rest of the plots are drawn underneath the two plots.

If a set of chemicals are selected using check boxes in marker #6, those chemicals will appear at the top of the plots.

The plots will only show the measurements in years selected by marker #11. Marker #15 is allows selection of a number between zero and seven. This number is used to filter uninteresting plots that have less than some number of distinct values. For example, some chemicals, e.g Carbendazim, are always measured to be 0 in some stations. Since such measurements are of no interest in visualization, using a large value in this selector will filter such plots with only few distinct measurements.

3.2.4 Saving and Restoring Favourite States

The section pointed at by marker #16 allows the user to store favourite selections, e.g. selection of station, chemicals, and year. The user can return to the stored selection in a later time.

3.2.5 Finding Peaks in the Plots

This section is used to find a peak in the chemical plots for each measurement. Analyzing peaks in the measurements is part of the exploratory data analysis process.

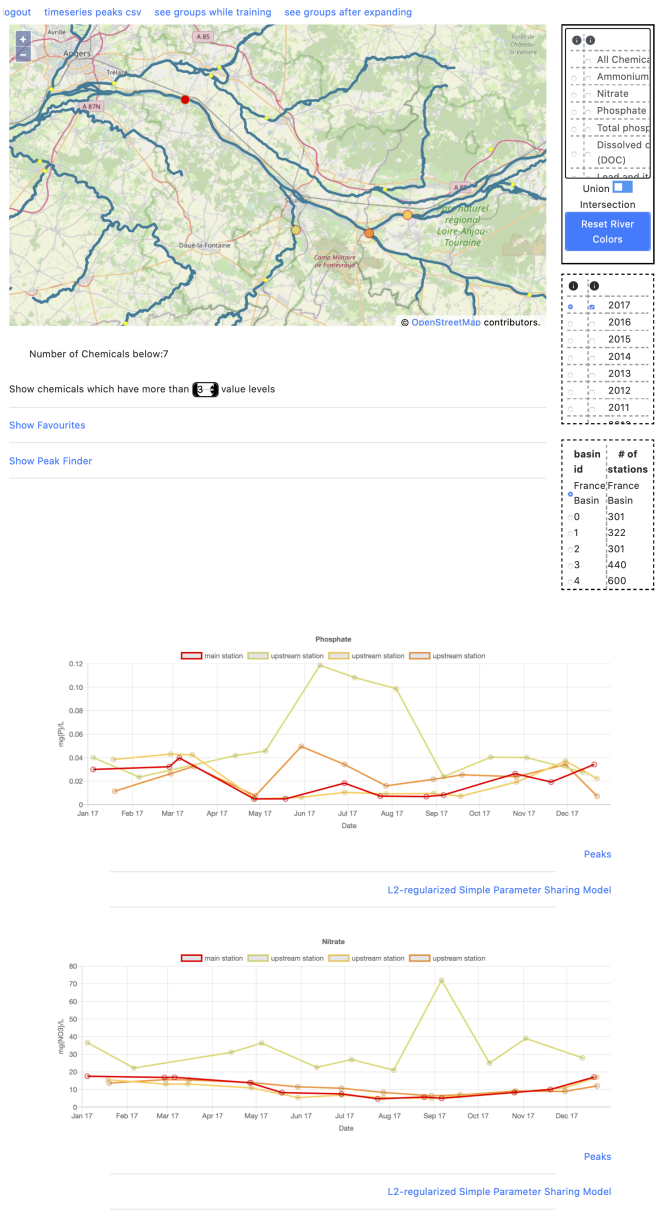


Figure 3.4: In this figure, the station pointed at by pointer #2 in Figure 3.1 is selected. The map is zoomed in on the selected station. The upstream stations (colored with different shades of orange) to the selected station (colored with red) can be seen. Two plots show the measurements for two chemicals done in the selected station in 2017. The rest of the plots are drawn underneath these two plots.

3.2.6 Visualizing Anomalies

Hyperlinks pointed at by markers #3 and #4 direct user to two other pages where the user can visualize the detected anomalies. The method used for learning anomaly classes and the content of these pages are explained in Section 3.3.3.

3.3 Hierarchical Parameter Sharing on Water Pollution

3.3.1 Dataset Generation

In order to prepare a dataset for training and testing different models, we only kept the readings from 2013 to 2017 for phosphate for the stations in the Loire basin in France shown in figures 3.1, 3.3, and 3.2. This basin was chosen because the dataset includes many samples in this basin. This basin includes 295 stations and 9051 phosphate readings. Note that the readings in the dataset are done irregularly and at intervals usually longer than a month. Some stations in the Loire basin do not have any phosphate readings in 2013 to 2017.

We do two sets of tests on this dataset to analyze the performance of different hierarchical parameter sharing model in interpolation and future extrapolation. For interpolation test, we split the dataset to 1000 test samples and 8051 training samples. For extrapolation test, we split the measurements from year 2015 and before as training set and the measurements for 2016 and later as test set.

3.3.2 Model Training

We train various models on the interpolation and future extrapolation datasets:

Mean Predictor

This is a very simple model which calculates the mean of the training measurements and predicts that value for all the test samples.

L_2-Regularized baseline parameter Sharing Model

To train this model on the dataset, we need to decide on some pre-defined classes. We chose seven sets of classes. We have classes for:

- each station: 295 classes
- each year: 5 classes
- each season: 4 classes
- each combination of every year and season: $5 \times 4 = 20$ classes
- each month: 12 classes
- each combination of every station and year: $295 \times 5 = 1475$ classes
- each combination of every station and season: $295 \times 4 = 1180$ classes
- each combination of every station, year, and season: $295 \times 5 \times 4 = 5900$ classes

In total we have 8891 possible classes, but some of these classes are empty, therefore we actually end up with 7291 nonempty classes. Note that many of these classes have non empty intersections and most of them have more than one sample.

Top-down Parameter Sharing Model

We use the same sets of classes chosen for the L_2-regularized baseline parameter sharing model. We set $\lambda = 1$. The results are not very sensitive to different values of λ.

Spatial BYM Model with a Temporal Random Walk

This model is an adaptation of the model introduced by Blangiardo and Cameletti [5]. it is a generalized linear model from the Gaussian family with an identity link function and a linear predictor as follows:

$$y_{lt} \sim Normal(\mu = \eta_{lt}, \sigma^2 = \frac{1}{\tau}) \tag{3.1}$$

$$\eta_{lt} = b_0 + \gamma_t + \phi_t + v_l + u_l \tag{3.2}$$

with y_{it} being the measurements at location l at time step t. The parameters in this model are explained in the following:

- b_0 quantifies the average measurements in all the data,
- γ_t represents a random walk of order 2 defined as:

$$\gamma_t | \gamma_{t-1}, \gamma_{t-2} \sim Normal(2\gamma_{t-1} + \gamma_{t-2}, s_\gamma^2) \tag{3.3}$$

- ϕ_t is represented by independent and identically distributed(iid) Gaussian variables for each time step as follows: $\phi_t \sim Normal(0, 1/\tau_\phi)$.
- v_l is an unstructured residual modeled using independent and identically distributed(iid) Gaussian variables for the different locations as $v_l \sim Normal(0, s_v^2)$,
- u_l is a specially structured residual modeled as a the conditional autoregressive (CAR) model used to model spatial interactions [2] named Besag-York-Mollié (BYM) model introduced by Besag et al. [3]. This is a model that allows us to specify that variables u_l for locations *close* to each other are correlated with each other. Assuming there are L locations $\{u_1, \ldots, u_L\}$, the closeness is specified with a graph of these locations. Defining N_l to be the number of neighbors of location l, $\mathbf{u}_{-l}$ as the list of all locations except location l, the BYM model is specified as follows:

$$u_l | \mathbf{u}_{-l} \sim Normal(\mu_l + \frac{1}{N_l} \sum_{k=1}^{L} a_{lk}(u_k - \mu_k), s_l^2) \tag{3.4}$$

  where a_{lk} is 1 if locations l and k are neighbors. In this model, the constraints of $\sum_{k=1}^{L} = 0$ and $\mu_k = 0$ are typically set (for example, Lee [15] introduces this model as the simplest CAR prior) to form the following final distribution over u_l:

$$u_l | \mathbf{u}_{-l} \sim Normal(\frac{1}{N_l} \sum_{k=1}^{L} a_{lk} u_k, s_l^2) \tag{3.5}$$

This model was trained using R-INLA[21]. The R-INLA default priors were used for parameters τ, b_0, s^2_γ, τ_ϕ, s^2_v, and s^2_l. The random walk in this model can only work on a dataset with temporally regular samples; therefore, we only test this model on future extrapolation task where we interpolate the training set and take weekly samples of the interpolated training set. In addition, R-INLA cannot handle the large number of samples generated from the weekly interpolated dataset, so we train this model only on the interpolated measurements from 2014/11/1 to 2016/1/1.

3.3.3 Anomaly Detection

We do unsupervised anomaly detection on the water-quality dataset using the hierarchical parameter sharing models. In Section 2.2, we discussed three types of anomalies that can be extracted using a parameter sharing model. In this section, we discuss how those anomalies can be applied to our phosphate dataset. To identify the three types of anomalies in the dataset, we first need to train a hierarchical parameter sharing model on the dataset. For this, we use the L_2-regularized baseline parameter sharing model trained on the dataset (explained in Section 3.3.2) which we will call the initially trained L_2-regularized model.

Anomalies from class parameters and noise parameters with large absolute values can be easily extracted from the initially trained L_2-regularized model by sorting the class parameters and noise parameters and reporting the ones with the largest absolute value. For learning groups of observations with similar noise parameters, we learn such groups step by step by picking an observation with a large noise parameter and expanding the group in time and space to incorporate close samples until the group becomes as populated as possible. We can learn as many groups as possible in this fashion. In the following, we explain in detail how an anomaly group can be learned starting with one sample with a large noise parameter.

Learning groups of observations with similar noise parameters

To learn anomaly classes, we start with the sample that has the largest absolute noise. We want to expand this anomaly class greedily in space and time to make it as populated as possible while achieving the lowest possible training loss. In our implementation, every anomaly class includes a set of stations and has a time interval $[t_1, t_2]$ where t_1 is the first day of some month, e.g. Oct 1st, 2016 and t_2 is the last day of some month, e.g. April 30th, 2017; we assume the time interval of an anomaly cannot be shorter than a month.

Having picked the sample that has the largest absolute residual error e_j, we create a new anomaly class A with the station that the sample is in and the month that the sample is in. As a result, one new offset parameter σ_A is introduced for this anomaly class. Note that this anomaly class might start with more than one sample if there is any other sample in the same month and station as the initially picked sample, but for now assume the anomaly starts with only one sample. We initialize σ_A using Theorem 2, so $\sigma_A = \frac{e_j}{2}$ and e_j changes to $\frac{e_j}{2}$. Having introduced this new anomaly class, we can observe that we have already decreased the training loss by $e_j^2 - \left(\left(\frac{e_j}{2}\right)^2 + \left(\frac{e_j}{2}\right)^2\right) = \left(\frac{e_j}{2}\right)^2$. Note that loss is computed using Equation (2.2). Now, we try to expand the anomaly class greedily step by step by either expanding

it temporally or spatially. At each step, we accept the expansion that gives us the least final loss. There are 6 candidates for expanding the anomaly class temporally: Expanding the time interval 1 to 3 months forward and backward. The candidates for expanding the anomaly class spatially are as follows:

- Expand the set of stations by adding in one of the upstream stations of one of the existing stations in the anomaly class
- Expand the set of stations by adding in 1 to 3 closet stations to one of the existing stations in the anomaly class

When expanding the anomaly using the candidates mentioned above, we assume that Theorem 2.1 holds locally and we use it to compute the new value for σ_A after expansion. We also approximate the new errors of the samples in the anomaly class as their original values minus the new σ_A. These approximations are refined by training the model for 150 gradient descent steps every time we learn 10 anomaly class. It turns out that simply having the sample with the maximum absolute residual error individually in one anomaly class yields a lot of decrease in loss. This is a trivial anomaly with only one sample which we do not want. Therefore, we filter out the candidates that cannot achieve a training loss lower than their maximum absolute residual error individually being in one anomaly class.

We stop learning anomaly classes after we have learned 300 of them. There is a special case not covered in the previous paragraph: If the sample that has the largest absolute residual error could not be expanded and ended up being the single sample in the new anomaly class, we destroy the anomaly class and we store the sample so that in the next iteration we can skip this sample and pick the next sample that has the largest absolute residual error. The set of stored samples is reinitialized every 100 anomaly classes learned. The following examples, illustrate in more detail how the anomaly classes were learned.

Example 3.1. In Figure 3.5a, we are showing training errors for each observation after initial training. The initial training loss is 10^2+8^2 plus the sum squared of all existing class parameters and all errors for other observations(not shown in Figure 3.5a). From now on, we will not mention the existence of "sum squared all existing class parameters and all errors for other observations" when we are computing training loss; therefore, for the purpose of this example, the initial training loss is $10^2+8^2=164$. Observation 1 has the largest absolute training error, which is 10. We create a new class including only this observation. This will cause the addition of a new class parameter, and as we said, we assume its value can be computed using theorem 2.1; therefore, the new class parameter will be $\frac{10}{2}=5$ and having this class will result in the error of observation 1 to be 5. Simply adding this class will result in the training loss to change to $5^2+8^2+5^2=114$. Now, we see if we can expand this anomaly to new samples, to reduce training loss as much as possible. Expanding the anomaly to one month forward is one of our search spaces, so we will see what will be the new training loss if we expand the anomaly to next month. This will result in observation 2 being added to our anomaly. The parameter for this anomaly will become $\frac{10+8}{3}=6$. The new error values for observations 1 and 2 will be 4 and -2 respectively. The new training loss will be $4^2+(-2)^2+6^2=56$. This is less than 114, so this expansion will be greedily accepted if there is no other expansion that will result in a lower training loss than 56.

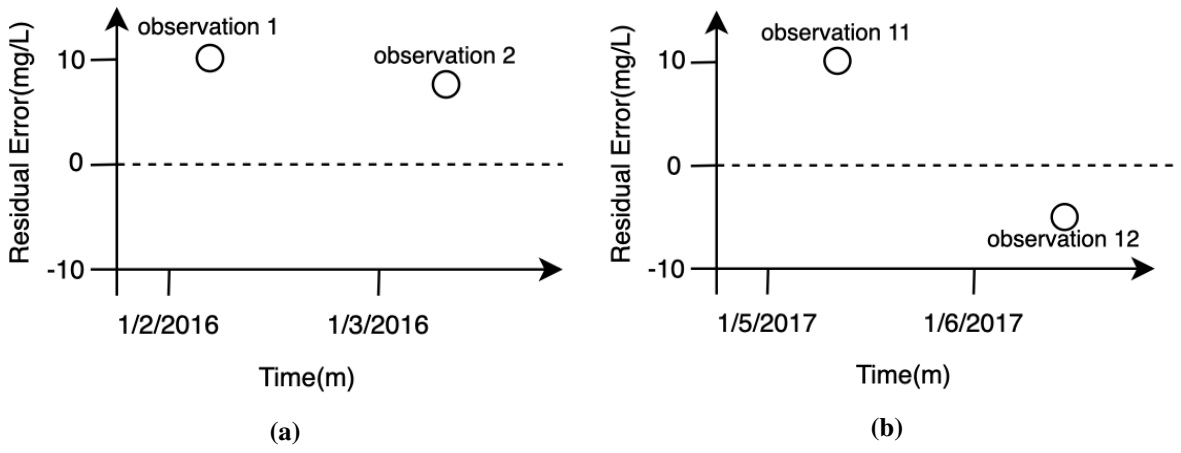


Figure 3.5: Residual errors for four training observations after initial training. The residual errors for observations 1,2,11,and 12 are 10,8,10, and -4 respectively.

Now, consider the example in Figure 3.5b. In this example, the initial training loss is $10^2 + (-4)^2 = 116$. Assume we construct a new anomaly with observation 11. The new class parameter will be $\frac{10}{2} = 5$ and the new training loss will be $5^2 + (-4)^2 + 5^2 = 66$. Now we compute training loss if we expanded the anomaly to one month forward. The new class parameter for the anomaly will be $\frac{10-4}{3} = 2$. The error values for observations 11 and 12 will be 7 and -7 respectively. The new training loss will be $7^2 + (-7)^2 + 3^2 = 104$. This is greater than 66, so this expansion candidate will never be accepted. If the other candidates for expanding this anomaly class lead to training losses greater than 66, observation 11 will be put in a set so that in the next iteration the next observation with the largest absolute training error will be picked to start a new anomaly class.

Continuing the example for Figure 3.5b, assume that after a few iterations, we see that observation 12 is the observation with the largest absolute training error. The initial training loss is $10^2 + (-4)^2 = 116$. Now, we create a new anomaly class with observation 12. The class parameter will be $\frac{-4}{2} = -2$ and the new error for observation 12 will be -2. The new training loss will be $(-2)^2 + 10^2 + (-2)^2 = 108$. Now, let us examine what the new training loss will be if we expand this anomaly to one month backward. As seen in the previous paragraph, having observations 11 and 12 in an anomaly class leads to the training loss of $7^2 + (-7)^2 + 3^2 = 104$. This is smaller than 108, so this growth is about to be accepted! But we do not want to accept this expansion because we want the expansion to be symmetrical: Starting from observation 11 and starting from observation 12 should lead to the same result. We will add a new constraint for when expansion candidates are accepted greedily. In addition to the requirement of reduction in training loss, an expansion candidate is only accepted if its training loss is less than the loss we could have achieved by putting the worst predicted observation in the anomaly into one class by its own. For example, we saw that having observation 11 in its own class results in a loss of 66. We do not accept having observations 11 and 12 in a class because that yields a loss of 104 which is greater than 66.

In Figure 3.6, the first 400 steps are for initial training of an L_2-regularized baseline parameter

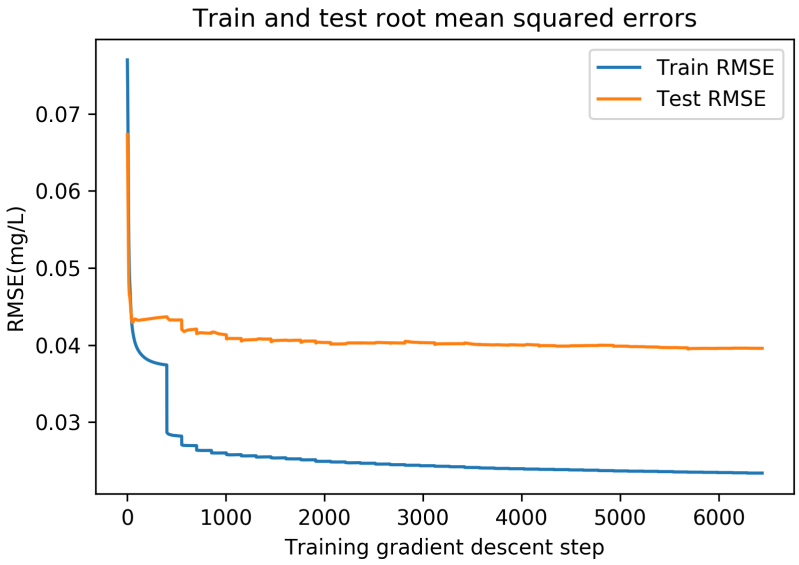


Figure 3.6: An example of train and test root mean squared errors(RMSE) during initial training (first 400 steps) and learning constructed anomalies for an L_2-regularized baseline parameter sharing model.

sharing model. The training and test loss reduce dramatically as we train for 150 steps when we learn 10 newly constructed anomalies.

Visualizing Process of Learning Anomaly Groups

In our water pollution platform, it is possible to visualize the anomaly groups while they are being learned. On this page of the platform, every step of learning anomalies is listed. When the user selects a step, the group that is being processes will be visualized alongside information about current loss value and how the group will be expanded in the next step.

Visualizing Learned Anomaly Groups

In our water pollution platform, it is also possible to visualize the final learned anomaly groups. A list of all anomaly groups learned ordered based on parameter value is shown to the user. The user can select an anomaly group in which case the time period of the anomaly and the stations where the anomaly is located is highlighted.

3.3.4 Model Evaluation

Interpolation

In Figure 3.7, we compare four different models as explained in Section 3.3.2:

- the baseline model, which predicts training mean for the test set,

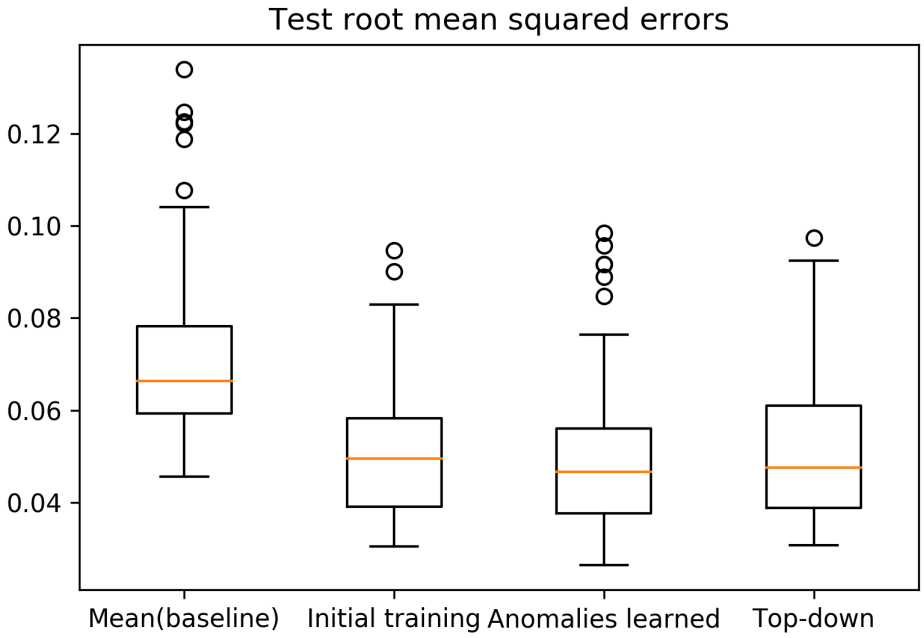


Figure 3.7: Test mean squared errors(MSE) for the four compared models.

- L_2-regularized baseline parameter sharing model after initial training
- L_2-regularized baseline parameter sharing model after learning constructed anomalies, as explained in Section 3.3.3
- top-down hierarchical parameter sharing model.

We do this comparison for 100 different random train-test splits with 8051 and 1000 samples, respectively. We can see that the initial training model on average has a 28% lower test error than the baseline model. Anomalies learned model has a 30.5% lower test error compared to baseline and a 3.3% lower test error compared to the initially trained model. The comparison between anomalies learned and initially trained model shows that the anomalies learned model is learning meaningful signals via learning constructed anomalies. The top-down model has a 12% higher test error compared to the initially trained model.

Future Extrapolation

Figure 3.8 compares the performance of five different models in interpolation and future extrapolation(predicting future observations). The mean predictor model, the $_2$-regularized baseline parameter sharing model after initial training and after learning anomalies, and the top-down model are the same models as previous section. In this section we are also comparing the Spacial BYM model with a temporal random walk explained in Section 3.3.2, referred to here as the BYM model. For $t > 2016/1/1$, train set is all observations in time range $[2013/1/1,\ 2016/1/1]$ and the test set is observations in

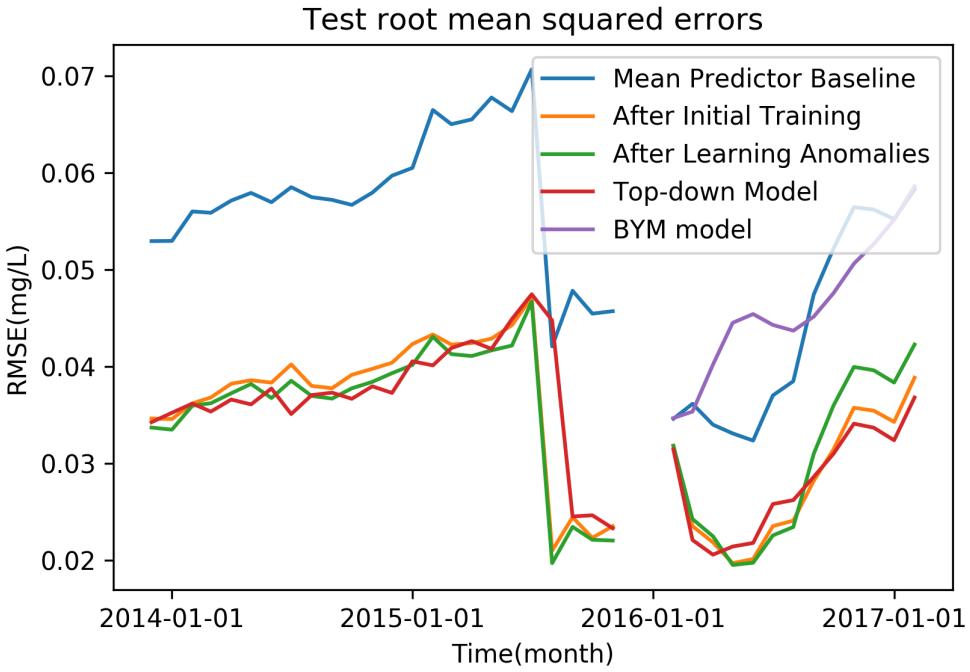


Figure 3.8: Comparing the performance of five models in interpolation ($t < 2016/1/1$) and extrapolation setting ($t > 2016/1/1$).

$[2016/1/1, t]$; this is when out model is predicting samples in the future. For $t < 2016/1/1$, test set is 30% randomly selected observations in $[t, 2016/1/1]$ and the training set is observations in $[2013/1/1, 2016/1/1]$ excluding test samples. For $t > 2016/1/1$, we did the experiments only once because the train and test sets are deterministic. Our training is also deterministic with GD steps with parameter initialization of 0 except the global parameter initialized to train mean. For $t < 2016/1/1$, the average of 10 runs with different random samples are shown. We can see that learning constructed anomalies has caused overfitting for extrapolation. Comparing model for t before and after 2016/1/1 shows that it outperforms the baseline by same margin in interpolation and future extrapolation. We can see that the BYM model performed poorly because the data was interpolated for the BYM model, which might have caused the data to become harder to learn.

Chapter 4

Parameter Sharing in Binary Classification

In the previous section, we applied the hierarchical parameter sharing models introduced in Chapter 2 on Waterbase-Water Pollution dataset[8], which was a regression task. In this section, we analyze the performance of the hierarchical parameter sharing models in classification tasks. In this chapter we try to use model confidence to get insight on how models work and to understand their differences. Model confidence in this context refers to the general ability of model to make smoothed predictions. A hierarchical parameter sharing model is considered to be overconfident when its prediction for a class is more intense than what is warranted by the observations in that class, e.g. only one negative example in a class should not make its probability very close to zero. A non-overconfident model will utilize samples in other classes and prior knowledge alongside the samples in the class to make a prediction about it. Overconfidence becomes an important issue in the context of datasets with small number of samples. In small datasets, the model should not be confident on solely the samples in a class to make a prediction for that class. In such cases, the samples in other classes can act as a guide to help smooth the model's prediction for a class. Smoothing can also be towards a default value; for example, Laplace smoothing smooths binary predictions towards 50%.

4.1 Overconfidence in Hierarchical Parameter Sharing Models

The following example shows that the original hierarchical parameter sharing models introduced in section 2 can be overconfident in some classification tasks.

Example 4.1. Consider the case of a binary classification task where we have observed 3 samples in class A all of which were observed to be zero. Figure 4.1 shows the result of learning an L_2-regularized baseline parameter sharing model on this dataset. A top-down model learned on the same dataset would result in the same learned parameters. As shown in this figure, all parameters will be learned to be 0; therefore, the model's prediction for an unobserved sample under class A is 0. We consider this prediction to be overconfident.

To improve both of the hierarchical parameter sharing models above, we can regularize the global

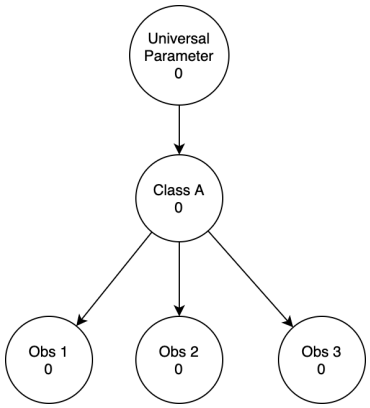
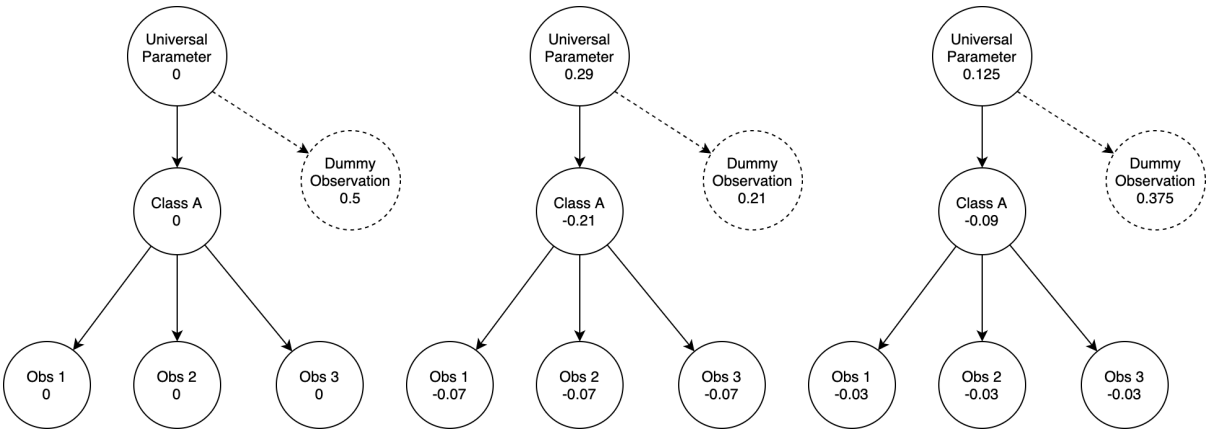


Figure 4.1: An L_2-regularized baseline parameter sharing model or a top-down hierarchical parameter sharing model learned on a binary task with 3 samples under class A all of which were observed to be zero.

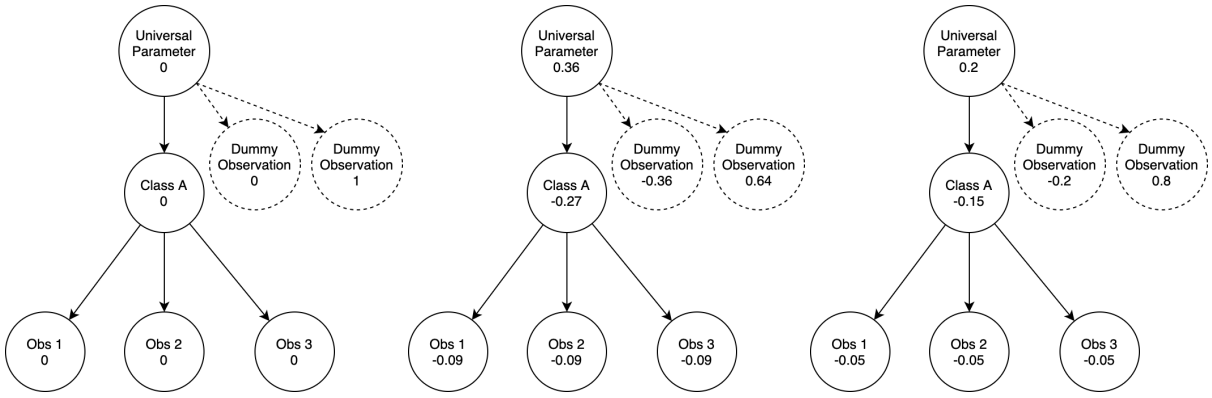
parameter towards 0.5 by adding the term $(\sigma - 0.5)^2$ to the loss functions in expressions (2.5) and (2.18), where σ is the global parameter. Note that same effect could have been achieved by adding a dummy observation with value of 0.5 under global parameter[20]. This change is investigated in the following example:



(a) initial state **(b)** L_2-regularized model learned **(c)** Top-down model learned

Figure 4.2: a) Initial state of a hierarchy with three binary observations all of which are observed to be 0. A dummy observation with value 0.5 is added under global parameter to facilitate regularization of the global parameter towards 0.5 b) An L_2-regularized model learned on the hierarchy shown in (a). c) A top-down model learned on the hierarchy shown in (a).

Example 4.2. To improve the overconfidence issue shown in example 4.1, we add a regularization term in the loss function of that model for the global parameter regularizing it towards 0.5. This can be achieved by adding the term $(\sigma - 0.5)^2$ to its loss function or alternatively by adding a dummy observation under the global parameter with a value of 0.5. The second method has been used in drawing the hierarchy shown in Figure 4.2. This figure shows that the prediction of the L_2-regularized model for class A is now $0.29 - 0.21 = 0.08$ while the prediction of the top-down model for class A is now



(a) initial state **(b)** L_2-regularized model learned **(c)** Top-down model learned

Figure 4.3: a) Initial state of a hierarchy with three binary observations all of which are observed to be 0. Two dummy observations with values 0 and 1 are added under global parameter to facilitate regularization of the global parameter towards 0.5 b) An L_2-regularized model learned on the hierarchy shown in (a). c) A top-down model learned on the hierarchy shown in (a).

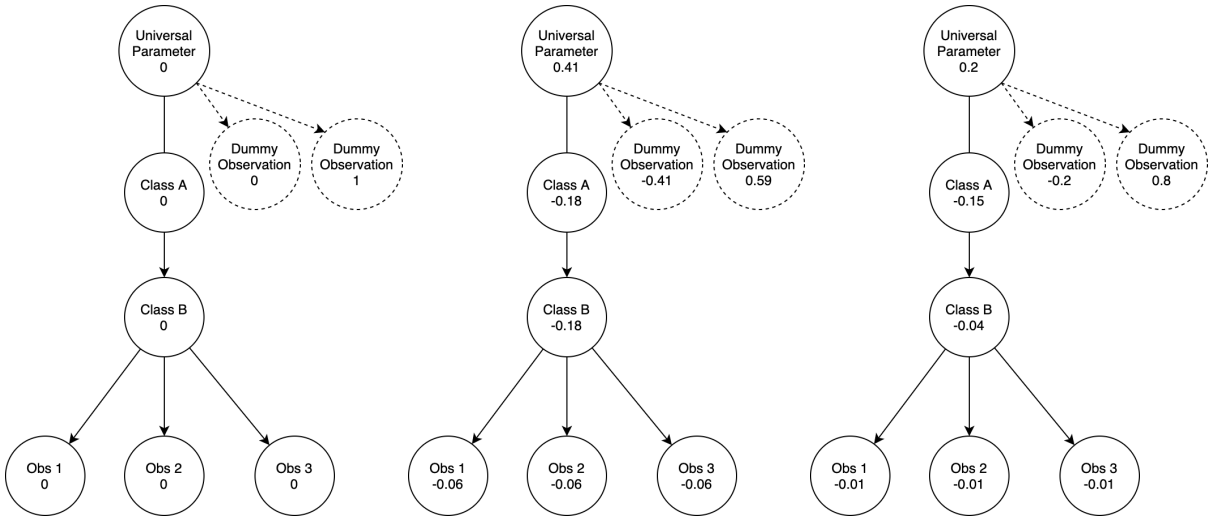
$0.1 - 0.075 = 0.025$.

Instead of adding only a single dummy observation under the global parameter, we could have added one positive dummy and one negative dummy observation under the global parameter. This is analyzed in the following example:

Example 4.3. In this example, to improve the overconfidence issue shown in example 4.1, we add two dummy observations under the global parameter: one positive and one negative observation. The resulting models are shown in Figure 4.3. Note that the universal parameter in Figure 4.3c is exactly what would be produced by Laplace smoothing.

In the next example, we see that in the top-down model the smoothing vanishes as the hierarchy becomes deeper.

Example 4.4. In Figure 4.4, we have altered the hierarchy in the previous example by simply adding a new class B under class A. This figure shows the result of learning the two parameter sharing models on this new hierarchy. In the L_2-regularized baseline model the universal parameter has increased compared to the previous example. For both of the models, predictions for class B are more confident (closer to zero) compared to the previous example. This shows that the effects of the smoothing introduced in previous example diminishes as the hierarchy becomes deeper. This vanishing is stronger in the top-down model compared to the L_2-regularized baseline model. With just three examples, predictions for instances in class B are 0.06 and 0.01 for the two models. This is arguably more confident than one should be with just three observations.



(a) initial state **(b)** L_2-regularized model learned **(c)** Top-down model learned

Figure 4.4: a) A dataset with similar hierarchy to the one in Figure 4.3 except that a new class B is added under class A b) An L_2-regularized model learned on the hierarchy shown in (a). c) A top-down model learned on the hierarchy shown in (a).

4.2 Experiment Setup

In this section we introduce multiple real-world and synthetic datasets with a focus on small datasets because smoothing and regularization are most important in small datasets. We also introduce multiple models that we train on the datasets. The models will be trained on 1000 random train-test splits for different number of train samples. The goal is to compare performances of the models as the number of training samples increases.

4.2.1 Preparing Datasets

The following datasets are the datasets that we use to evaluate our models on. For each dataset, we also introduce the set of classes we use to train a hierarchical parameter sharing model on the dataset.

Promoters

The promoters dataset from UCI machine learning repository[7].

- Title of Database: E. coli promoter gene sequences (DNA) with associated imperfect domain theory
- Number of Instances: 106
- Used Attributes:
  - Label: positive or negative
  - 57 sequential nucleotide (”base-pair”) positions: These 57 features are used to predict the label. Each feature has one of the values A, T, C, or G.

- Classes Defined for Training Hierarchical Parameter Sharing Model:
  - $57 \times 4 = 228$ classes for each feature in the dataset.
  - $\binom{57}{2} \times 4 \times 4 = 25536$ classes for each pair of features in the dataset.

Wisconsin Prognostic Breast Cancer(WPBC)

Wisconsin prognostic breast cancer(WPBC) dataset from UCI machine learning repository[7].

- Title of Database: Wisconsin Prognostic Breast Cancer (WPBC)
- Number of Instances: 198
- Used Attributes:
  - Label: R = recur, N = nonrecur
  - 30 real-valued features used to predict the label.
- Classes Defined for Training Hierarchical Parameter Sharing Model:
  - $30 \times 10 = 300$ classes derived by descretizing each feature in the dataset. We discretize each of the real-valued features into 10 discrete levels: First level includes all the values smaller than the 10th percentile; second level includes all the values between 10th and 20th percentiles; and so on.
  - $\binom{30}{2} \times 10 \times 10 = 43500$ classes for each pair of the 30 discretized features

Wisconsin Diagnostic Breast Cancer(WDBC)

Wisconsin diagnostic breast cancer(WDBC) dataset from UCI machine learning repository[7].

- Title of Database: Wisconsin Prognostic Breast Cancer (WPBC)
- Number of Instances: 569
- Used Attributes:
  - Label: M = malignant, B = benign
  - 30 real-valued features used to predict the label.
- Classes Defined for Training Hierarchical Parameter Sharing Model:
  - $30 \times 10 = 300$ classes derived by descretizing each feature in the dataset. This was done similar to WPDC dataset.
  - $\binom{30}{2} \times 10 \times 10 = 43500$ classes for each pair of the 30 discretized features.

Breast Cancer

Breast Cancer dataset from UCI machine learning repository[7].

- Title of Database: Breast cancer data
- Number of Instances: 286
- Used Attributes:
  - Label: no-recurrence-events, or recurrence-events
  - 9 discrete features used to predict the label.
- Classes Defined for Training Hierarchical Parameter Sharing Model:
  - Each level of the discrete features at first layer of the hierarchy
  - Combinations of pairs of the discrete features at second layer of hierarchy

Synthetic Dataset

This dataset was created using the Bayesian network shown in Figure 4.5 as ground truth. All the nodes have binomial distribution. Nodes $X_1, \ldots, X_5$ are five boolean features which determine the value of y. Nodes $ObxX_1, \ldots, ObsX_5$ model the missingness in the measurements. They determine if their respective feature was observed or is unknown. For example, if $ObsX_1$ is true, it means that the value of X_1 is known in measurement, but if $ObsX_1$ is false, the value of X_1 is unknown in that measurement. 25000 measurements were sampled from this ground truth. The prior probabilities for X_i and the conditional probabilities of $p(y|X_1, \ldots, X_5)$ were chosen uniformly from $[0.1, 0.9]$ and fixed among 25000 measurements. The conditional probabilities of $p(ObsX_i = T|X_i = T)$ and $p(ObsX_i = T|X_i = F)$ were chosen uniformly from $[0.6, 0.9]$ and fixed among 25000 measurements.

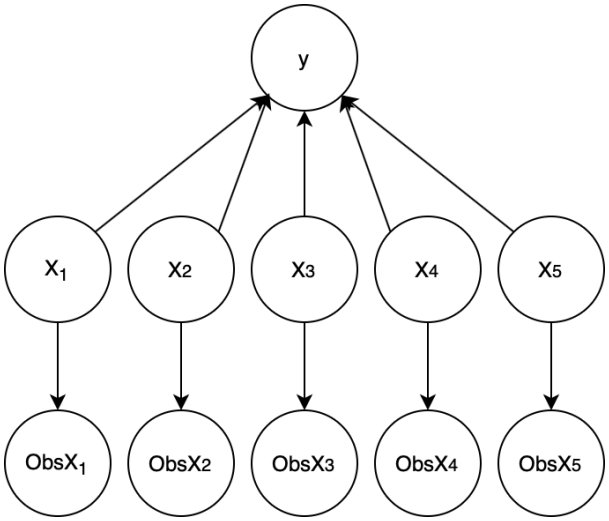


Figure 4.5: Bayesian network used as ground truth to create the *Synthetic Dataset*.

4.2.2 Model Training

In this section, we introduce the models we train on the datasets of section 4.2.1.

Naive Bayes

Naive Bayes model for datasets with continuous and discrete features. L_k are the K different outcomes (classes or labels). For continuous features $C_1, \ldots, C_T$, and discrete features $D_1, \ldots, D_R$:

$$p(C_t|L_k) \sim Normal(\mu_{t,k}, s_{t,k}) \tag{4.1}$$

$$p(D_r|L_k) \sim Multinomial(p_{r,k}) \tag{4.2}$$

$$p(L_k|c_1, \ldots, c_T, d_1, \ldots, d_R) \propto (\prod_{t=1}^{T} p(c_t|L_k))(\prod_{r=1}^{R} p(d_r|L_k)) \tag{4.3}$$

where parameters $\mu_{t,k}$, $s_{t,k}$, and $p_{r,k}$ are estimated using maximum likelihood.

Logistic Regression

L_2-regularized logistic regression model minimizing the following cost function:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1) \tag{4.4}$$

where y_i are the observations, w is a one-dimensional array of all the weights and X is a two-dimensional array of all the features for each observation.

L2-Regularized Baseline Parameter Sharing Model

The L_2-regularized baseline parameter sharing model as explained in section 2.4.1. The classes chosen for each dataset is explained for each dataset in section 4.2.1.

Top-Down Hierarchical Parameter Sharing Model

The Top-Down hierarchical parameter sharing model as explained in section 2.4.2. The same set of classes as the L_2-regularized baseline parameter sharing model are used for each dataset.

L2-Regularized Baseline Parameter Sharing Model - T/F smoothed

The L_2-regularized baseline parameter sharing model smoothed with two dummy samples of 0 and 1 under the global parameter, as explained in Example 4.3. To train this model, the same set of classes as the L_2-regularized baseline parameter sharing model are used.

Top-Down Hierarchical Parameter Sharing Model - T/F smoothed

The Top-Down hierarchical parameter sharing model smoothed with two dummy samples of 0 and 1 under the global parameter, as explained in Example 4.3. To train this model, the same set of classes as

the L_2-regularized baseline parameter sharing model are used.

4.3 Experiment Results

The models were trained on 1000 random train-test splits for different number of train samples, k. We use logloss for classification datasets and root mean squared error for regression datasets. In figures 4.6,4.7,4.8,4.9, and 4.10 the test errors for each model were averaged over 1000 random train-test splits.

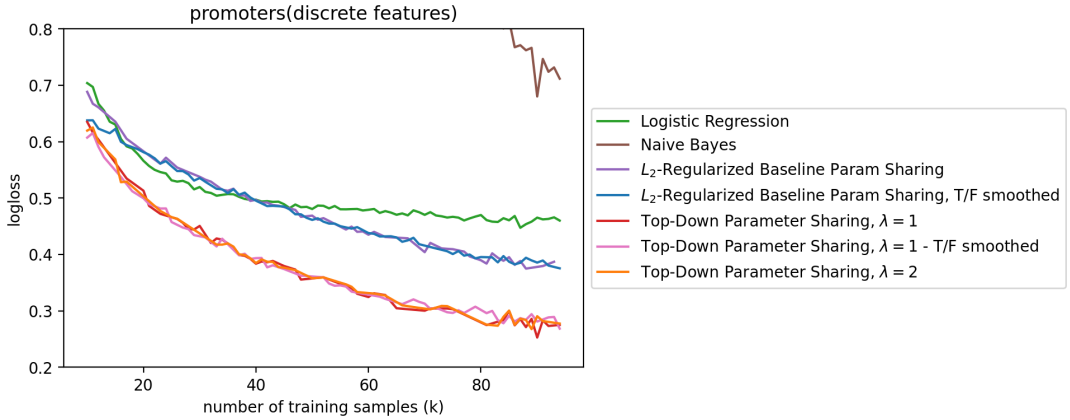


Figure 4.6: Test loss for multiple models compared on promoters dataset with different number of training samples, k. For each k, we average test loss over 1000 random train-test splits. Smaller is better.

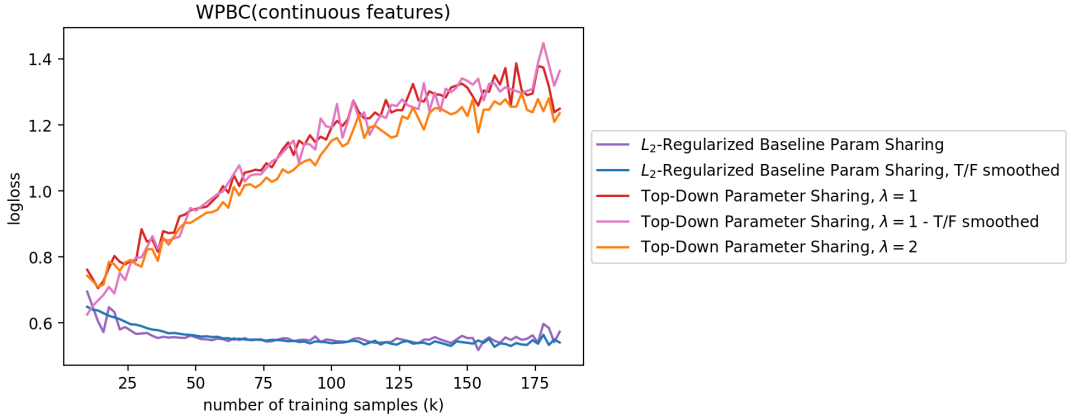


Figure 4.7: Test loss for multiple models compared on WPBC dataset with different number of training samples, k. For each k, we average test loss over 1000 random train-test splits. Smaller is better.

According to these figures, it is not possible to draw a conclusion about the performance of L_2-regularized baseline model compared to the top-down model. More studies are needed to improve these model in binary classification task. The plots also show that the smoothed version of the models work better than the nonsmoothed versions when the size of the training set is small.

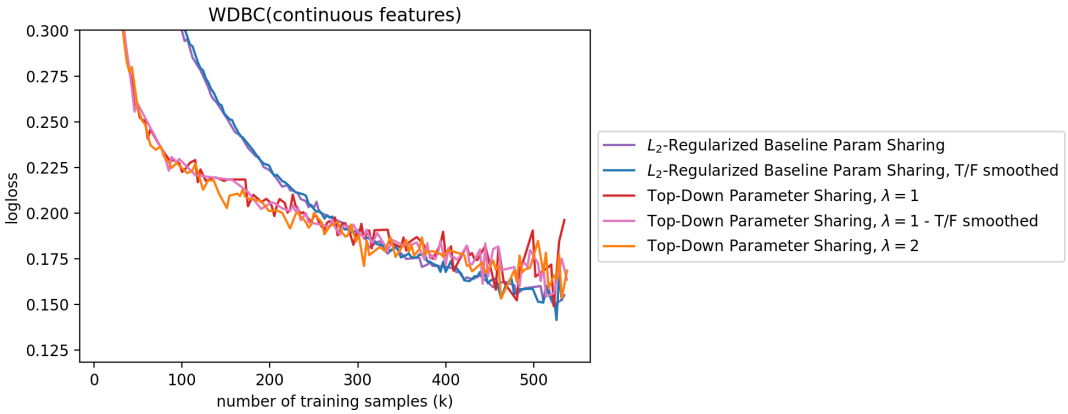


Figure 4.8: Test loss for multiple models compared on WDBC dataset with different number of training samples, k. For each k, we average test loss over 1000 random train-test splits. Smaller is better.

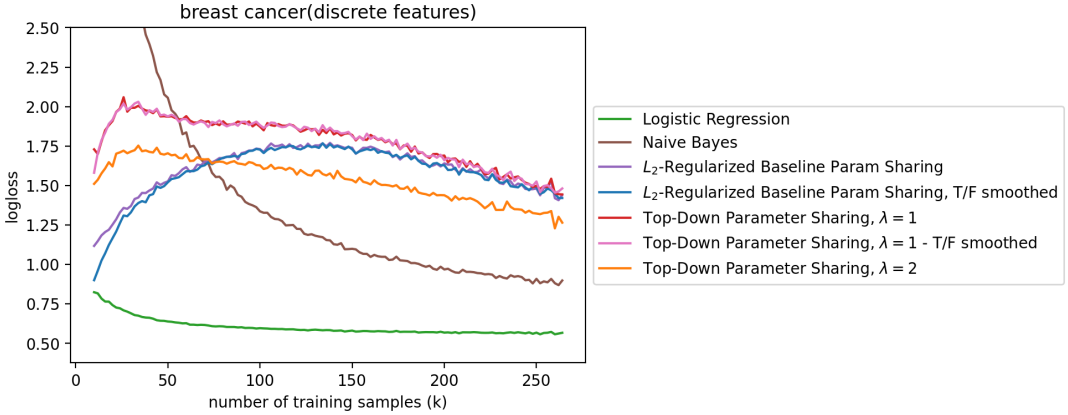


Figure 4.9: Test loss for multiple models compared on breast cancer dataset with different number of training samples, k. For each k, we average test loss over 1000 random train-test splits. Smaller is better.

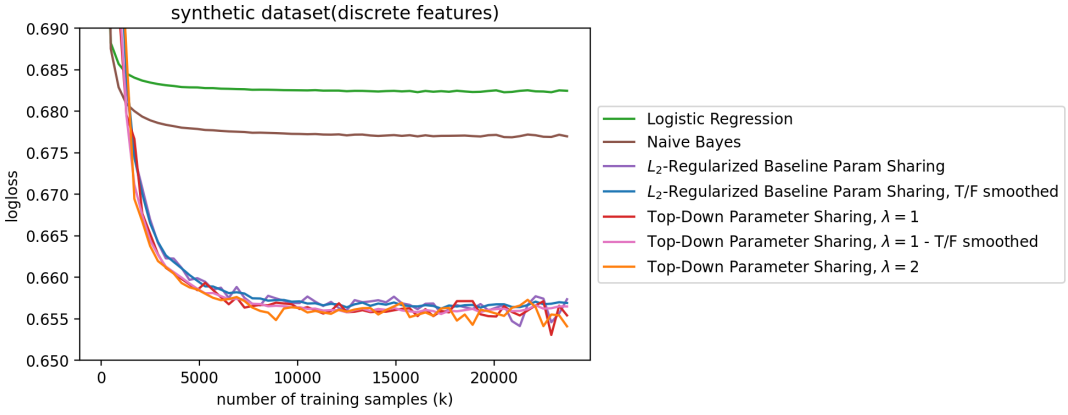


Figure 4.10: Test loss for multiple models compared on synthetic dataset with different number of training samples, k. For each k, we average test loss over 1000 random train-test splits. Smaller is better.

Chapter 5

Conclusion and future work

5.1 Future directions

Based on the examples given in Chapter 4, it seems like both the top-down and baseline parameter sharing models suffer from overconfidence and the suggested method does not completely solve this issue. In addition, both models suffer from other issues that render the offsets difficult to explain. For instance, in figure 4.4b, the learned offsets for class A and B are equal while it is more reasonable that the offset for class B be zero because class A has already learned all the information about the samples underneath. Knowing that a sample is under class B rather than class A should not change the prediction of the model. We have tried improving the top-down model by trying to bound the amount of information that is passed down to the child from the parent. We have also tried to modify the top-down model in a fashion that learning the offset for a class is dependent on the ratio of number children underneath versus under the siblings. Our engineered designs did not improve the test loss, but future studies might be able to solve these issue using similar perspectives on the workings of the models.

Future studies can investigate the relationship between our models and hierarchical Bayesian models. Our method can make predictions about a sample under one class or multiple classes. For example, our model can make different predictions about a sample in 2017, a sample in location l, or a sample in 2017 and l. Note that in this case class l and class of 2017 are disjoint, but neither of them is a subset of the other. It is not clear how this type of setting can be modeled through hierarchical Bayesian models. For example, in the famous example of patient mortality in different hospitals modeled through a hierarchical Bayesian model, it is not clear how one is supposed to model a patient that was hospitalized in two hospitals. In a tree DAG, where classes are either disjoint or subset of each other, a hierarchical Bayesian model with a network similar to the DAG hierarchy can be fit to the data. Future studies can investigate the difference between the predictions of our model and the predictions of the hierarchical Bayesian model for the data.

In a general DAG, classes can have multiple parents. In such cases, it is not trivial how the signal should be split between the parents. Future studies can investigate the workings of how the proposed hierarchical models split the signal between multiple parents.

Future studies can investigate the relationship between our models and a feedforward artificial neural network. One can imagine that an extension to our models is to set the values of each layer of the hierarchy as a function of the values of offsets in previous layers. This will result in a model similar to neural networks.

5.2 Conclusion

To summarize, we propose and investigate hierarchical parameter sharing models as an explainable model. The explainibility in the proposed model is achieved by the fact that the parameters or offsets in the model are all of the same unit of measure and they correspond to well-defined classes. The model can be utilized to provide two different types of explainability: Observation explainability and gestalt explainability. We train the model on the water-quality dataset and we are able to show that the model has a better performance than a baseline model and the BYM model, which is a type of generalized linear model. In this process, we also created a water pollution platform for exploratory data analysis on the water-quality dataset and anomaly detection using the proposed model. We learned that our model can learn explainable offsets for classes that can improve extrapolation of data in the near future. Finally, we investigated the performance of the parameter sharing model on the boolean classification where smoothing and regularization were deemed important specially in small datasets. The suggested smoothing method on both variants of our model was able to improve its predictions in small dataset setting.

Bibliography

[1] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82 – 115, 2020. ISSN 1566-2535. doi:https://doi.org/10.1016/j.inffus.2019.12.012. URL http://www.sciencedirect.com/science/article/pii/S1566253519308103. → pages 1, 14

[2] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):192–236, 1974. ISSN 00359246. URL http://www.jstor.org/stable/2984812. → pages 21

[3] J. Besag, J. York, and A. Mollié. Bayesian image restoration, with two applications in spatial statistics. *Annals of the Institute of Statistical Mathematics*, 43:1–20, 03 1991. doi:10.1007/BF00116466. URL https://doi.org/10.1007/BF00116466. → pages 21

[4] M. Blangiardo and M. Cameletti. *Spatial and Spatio-temporal Bayesian Models with R - INLA*. Wiley, 2015. ISBN 9781118326558. doi:10.1002/9781118950203. URL http://dx.doi.org/10.1002/9781118950203. → pages 3

[5] M. Blangiardo and M. Cameletti. *Spatial and Spatio-temporal Bayesian Models with R - INLA*, pages 238–240. Wiley, 2015. ISBN 9781118326558. doi:10.1002/9781118950203. URL http://dx.doi.org/10.1002/9781118950203. → pages 21

[6] E. M. Dogo, N. I. Nwulu, B. Twala, and C. Aigbavboa. A survey of machine learning methods applied to anomaly detection on drinking-water quality data. *Urban Water Journal*, 16(3): 235–248, 2019. doi:10.1080/1573062X.2019.1637002. URL https://doi.org/10.1080/1573062X.2019.1637002. → pages 1

[7] D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml. → pages 31, 32, 33

[8] European Environmental Agency. Waterbase - water quality, Apr 2019. URL https://www.eea.europa.eu/data-and-maps/data/waterbase-water-quality-2. Prod-ID: DAT-163-en, Created: 05 Apr 2019, Published: 09 Apr 2019, Accessed: 20 Jun 2019. → pages 15, 28

[9] K. Gade, S. Geyik, K. Kenthapadi, V. Mithal, and A. Taly. Explainable ai in industry. pages 3203–3204, 07 2019. ISBN 978-1-4503-6201-6. doi:10.1145/3292500.3332281. → pages 1

[10] D. Guo, A. Lintern, J. Webb, D. Ryu, U. Bende-Michl, S. Liu, and A. Western. A data-based predictive model for spatiotemporal variability in stream water quality. *Hydrology and Earth System Sciences*, 24:827–847, 02 2020. doi:10.5194/hess-24-827-2020. → pages 1, 2

[11] J.-Q. Jin, Y. Du, L.-J. Xu, Z.-Y. Chen, J.-J. Chen, Y. Wu, and C.-Q. Ou. Using bayesian spatio-temporal model to determine the socio-economic and meteorological factors influencing ambient pm2.5 levels in 109 chinese cities. *Environmental Pollution*, 254:113023, 2019. ISSN 0269-7491. doi:https://doi.org/10.1016/j.envpol.2019.113023. URL http://www.sciencedirect.com/science/article/pii/S0269749119322298. → pages 3

[12] O. Kisi and K. S. Parmar. Application of least square support vector machine and multivariate adaptive regression spline models in long term prediction of river water pollution. *Journal of Hydrology*, 534:104 – 112, 2016. ISSN 0022-1694. doi:https://doi.org/10.1016/j.jhydrol.2015.12.014. URL http://www.sciencedirect.com/science/article/pii/S0022169415009622. → pages 2

[13] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009. ISSN 0018-9162. doi:10.1109/MC.2009.263. URL http://dx.doi.org/10.1109/MC.2009.263. → pages 2

[14] T. Lai, H. Robbins, and C. Wei. Strong consistency of least squares estimates in multiple regression ii. *J. Multivariate Anal.*, 9:343–361, 01 1985. doi:10.1007/978-1-4612-5110-1_47. → pages 2

[15] D. Lee. A comparison of conditional autoregressive models used in bayesian disease mapping. *Spatial and Spatio-temporal Epidemiology*, 2(2):79 – 89, 2011. ISSN 1877-5845. doi:https://doi.org/10.1016/j.sste.2011.03.001. URL http://www.sciencedirect.com/science/article/pii/S1877584511000049. → pages 21

[16] C. Leigh, O. Alsibai, R. Hyndman, S. Kandanaarachchi, O. King, J. McGree, C. Neelamraju, J. Strauss, P. Talagala, R. Turner, K. Mengersen, and E. Peterson. A framework for automated anomaly detection in high frequency water-quality data from in situ sensors. *Science of the Total Environment*, 664:885–898, May 2019. ISSN 0048-9697. doi:10.1016/j.scitotenv.2019.02.085. → pages 1

[17] F. Lindgren, H. Rue, and J. Lindström. An explicit link between gaussian fields and gaussian markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):423–498, 2011. doi:10.1111/j.1467-9868.2011.00777.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2011.00777.x. → pages 3

[18] H. Liu, F. Hussain, C. L. Tan, and M. Dash. Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, 6(4):393–423, Oct 2002. ISSN 1573-756X. doi:10.1023/A:1016304305535. URL https://doi.org/10.1023/A:1016304305535. → pages 2

[19] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017. → pages 15

[20] D. L. Poole and A. K. Mackworth. *Artificial Intelligence: Foundations of Computational Agents*, page 307. Cambridge University Press, second edition, 2010. ISBN 978-1-107-19539-4. doi:10.1017/9781107195394. → pages 29

[21] H. Rue, S. Martino, and N. Chopin. Approximate bayesian inference for latent gaussian models by using integrated nested laplace approximations. *Journal of the Royal Statistical Society Series B*, 71:319–392, 04 2009. doi:10.1111/j.1467-9868.2008.00700.x. → pages 22

[22] C. J. Ruybal, T. S. Hogue, and J. E. McCray. Evaluation of Groundwater Levels in the Arapahoe Aquifer Using Spatiotemporal Regression Kriging. *Water Resources Research*, 55(4):2820–2837, Apr. 2019. doi:10.1029/2018WR023437. → pages 3

[23] Tiyasha, T. M. Tung, and Z. M. Yaseen. A survey on river water quality modelling using artificial intelligence models: 2000–2020. *Journal of Hydrology*, 585:124670, 2020. ISSN 0022-1694. doi:https://doi.org/10.1016/j.jhydrol.2020.124670. URL http://www.sciencedirect.com/science/article/pii/S002216942030130X. → pages 2

[24] Y. Tramblay, T. Ouarda, A. St-Hilaire, and J. Poulin. Regional estimation of extreme suspended sediment concentrations using watershed characteristics. *Journal of Hydrology*, 380:305–317, 01 2010. doi:10.1016/j.jhydrol.2009.11.006. → pages 2

[25] C. A. Varotsos, V. F. Krapivin, F. A. Mkrtchyan, S. A. Gevorkyan, and T. Cui. A novel approach to monitoring the quality of lakes water by optical and modeling tools: Lake sevan as a case study. *Water, Air, & Soil Pollution*, 231(8):435, Aug 2020. ISSN 1573-2932. doi:10.1007/s11270-020-04792-8. URL https://doi.org/10.1007/s11270-020-04792-8. → pages 1