LONDON'S GLOBAL UNIVERSITY

Online Multi-Task Learning with Long-Term Memory for Temperature Anomaly Prediction

Experiments and Applications to Climate Data

16036189[1]

MEng Computer Science

Mark Herbster

Submission date: May 2021

[1]**Disclaimer:** This report is submitted as part requirement for the MEng Computer Science at UCL. It is substantially the result of my own work except where explicitly indicated in the text. *Either:* The report may be freely copied and distributed provided the source is explicitly acknowledged
Or:
The report will be distributed to the internal and external examiners, but thereafter may not be copied or distributed except with permission from the author.

Abstract

Online learning models are trained incrementally; at each time step, these models accept data and generate a prediction for the current outcome. Such algorithms are ideal for use on temporal data (e.g. stock market data or climate data) and often provide advice for educated decision making - such as advising investments or environmental policies. In this paper, a new online multi-task learning algorithm utilising expert advice is applied to temperature anomaly prediction - a key indicator of climate change.

Expert advice algorithms accept predictions from a set of established experts - human or algorithm. This advice is used alongside the observed expert accuracy to generate predictions. Existing expert advice algorithms predict almost as accurately as the best predictors and require no assumptions about the underlying data. Building upon these algorithms, the multi-task learner employs global, long-term memory which allows previous expert performance to be 'remembered' and communicated amongst tasks. Empirical evidence is provided to demonstrate the superior performance of the multi-task learner, via a series of experiments on synthetic data. In these experiments, the multi-task learner routinely outperforms existing algorithms.

These experiments are then conducted on real-world data for the prediction of temperature anomalies. Currently, temperature anomaly predictions are taken from General Circulation Models (GCMs) which are mathematical models designed to simulate the Earth's climate. However, no consensus exists on the best model and there exists a high variance between GCM predictions. Using advice from a GCM expert set, the multi-task learning algorithm is tasked with generating predictions for different locations over decade-long climate simulations. The results show the developed multi-task learner significantly improves upon the current state-of-the-art for predicting temperature anomalies.

Acknowledgements

I would like to thank Mark Herbster, for his dedicated support throughout the year, and Lydia for being my continued inspiration.

Contents

1 Introduction **2**

1.1 Online Learning with Expert Advice . . . 2
1.2 Switching . . . 3
1.3 Long-Term Memory . . . 3
1.4 Multi-Task Learning . . . 3
1.5 Climate Prediction . . . 4
1.6 Thesis Contributions . . . 4
1.7 Thesis Structure . . . 5

2 Literature Review **6**

2.1 Notation . . . 6
2.2 Online Learning in Stationary Environments . . . 7
2.2.1 Static Expert Learners . . . 7
2.3 Learning in Switching Environments . . . 9
2.3.1 Fixed-Share Learner . . . 9
2.3.2 Variable–Share Learner . . . 10
2.4 Learning with Long-Term Memory . . . 11
2.5 Multi-task Learning . . . 12
2.6 Climate Modelling . . . 14

3 Synthetic Experiments **18**

3.1 Motivation . . . 18
3.2 Experimental Design . . . 19
3.2.1 Synthetic Data Generation . . . 19
3.2.2 Implementation . . . 20
3.2.3 Parameter Selection . . . 21
3.3 Results . . . 22
3.3.1 Single Task, No Switching . . . 22
3.3.2 Single Task, Switching . . . 22
3.3.3 Multi-Task, No Switching . . . 24
3.3.4 Multi-Task, Switching . . . 25
3.3.5 Observations . . . 28

4 Climate Data Experiments **29**

4.1 Motivation . . . 29
4.2 Experimental Design . . . 30
4.2.1 Data Preparation . . . 30

4.3 Experiments . . . 30
4.3.1 Single-Task Zonal Anomalies . . . 31
4.3.2 Multi-Task Zonal Anomalies . . . 33
4.3.3 Final Model . . . 36
4.4 Results . . . 38

5 Discussion **42**
5.1 Conclusion . . . 42
5.2 Evaluation . . . 42
5.3 Future Work . . . 43

Bibliography **43**

A Project Plan **47**
A.1 Aims . . . 47
A.2 Objectives . . . 47

B Table of Used GCMs **48**

C Final Climate Experiments Results **49**

D Infiite Hypothesis Space **51**
D.1 Experimental Design . . . 52
D.1.1 Synthetic Data Generation . . . 52
D.1.2 Implementation . . . 52
D.1.3 Results . . . 52

Chapter 1

Introduction

This chapter provides a brief overview of concepts discussed throughout the paper, primarily online learning and climate prediction. This overview is followed by a short summary of the thesis structure and contributions.

1.1 Online Learning with Expert Advice

In online environments, algorithms learn incrementally over an ordered sequence of T trials, labelled $t = 1, 2, ..., T$. At each trial, the algorithm observes the environment and generates a prediction for the trial outcome. Algorithms can utilise existing expert knowledge by receiving the predictions from a set of n experts. The expert set is a group of human experts or sub-algorithms, labelled $\epsilon_1, \epsilon_2, ..., \epsilon_n$. At trial t, a learner receives a vector of expert advice $\mathbf{x}_t$ where $x_{t,i}$ is the prediction submitted by ϵ_i. Using this input alongside any prior knowledge, the learner generates its own prediction, $\hat{y}_t$. After the learner submits its prediction, the real-world outcome, y_t, is revealed.

For each trial, a learner incurs loss on its prediction $L(\hat{y}_t, y_t)$. L is a predefined loss function $L : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ which returns a real value determined by the difference between a prediction and true outcome. A learner's aim is to minimising the overall loss incurred, $\sum_{t=1}^{T} L(\hat{y}_t, y_t)$, known as the cumulative loss.

Loss is minimised by generating accurate predictions. Under expert advice, this entails identifying which experts to listen to. Learners use the expert loss, $\ell_{t,i} = L(x_{t,i}, y_t)$ for $i = 1, ..., n$, to guide their predictions on future trials. This is done by maintaining weights which are updated at each trial and indicate the 'confidence' in a given expert.

As environments exist in which even the best expert may receive high absolute loss (e.g. adversarial environments), minimising absolute loss is infeasible. Therefore, loss bounds are often measured with respect to the loss of the experts. Regret is the term given to the difference between a learner's loss and the loss of the hindsight-optimal choice. In a stationary environment, the hindsight-optimal choice is the expert incurring minimal loss over a series of trials. Existing algorithms have proven tight worst-case regret bounds, which are linear in the number of best experts. A learner is guaranteed to incur, at most, the loss of the best expert plus a small margin of error. These bounds are proven without making any assumptions about the underlying distribution or expert performance.

1.2 Switching

Shifts in the underlying distribution can cause the best performing expert to vary over time. The hindsight-optimal choice is now the sequence of experts which incurs minimal loss. Best experts are now determined by their performance over a segment of the trials, rather than over the entire sequence. A task involving k changes in the best-performing expert is divided into $k+1$ segments, each with a corresponding best-expert. Each segment is noted by the label of the trial in which the segment begins and the label of the best expert for that segment. The partition of a task consists of a pair of vectors, representing the start times and best experts for each segment.

Learners must now determine the best performing expert for the current segment, and minimise regret accordingly. At the time of prediction, the task partition is unknown to the learner, and a learner must make educated guesses about the current best expert. In the static case, where a single best expert is identified over the whole sequence, this is done simply by observing cumulative loss. This is insufficient in a switching case, as it is possible for the best expert for a given segment to have high cumulative loss. A learner predicting solely based on cumulative loss will not adapt quickly, if at all, to new best experts.

It is possible for a learner to track all potential partitions and predict according to the partition with the lowest loss. However, the space required by this method grows exponentially as trials progress. This is impractical and more efficient solutions are required. Algorithms explored in this paper provide regret bounds that are logarithmic in the number of best experts and linear in the number of switches. Though these algorithms incur slightly more loss than the best partition, the space and time complexity required is linear in the number of experts.

1.3 Long-Term Memory

The ability to identify the best expert may be considered short-term memory. Long-term memory involves remembering experts which have previously performed well. For circumstances in which successful historical performance is indicative of future success, long-term memory allows reduces a learner's regret. In a segment in which an expert is repeated, the learner quickly regains confidence in the repeated expert and incurs lower segment loss than an algorithm only employing short-term memory. A simple implementation of long-term memory may include storing every observation. This is clearly impractical in an online setting where the number of observed trials is continuously increasing and algorithms may run over a considerable number of trials. In implementing long-term memory, it is crucial that algorithms do not require inappropriate amounts of storage or computation time.

1.4 Multi-Task Learning

Multi-task settings involve a set of tasks, where all tasks' domains are related via some underlying commonality. A multi-task learner has s tasks, each receiving predictions per task from a set of n experts. Algorithms now learn the sequence of best experts for each task. The best expert may vary between tasks, and task switches are asynchronous. Transfer learning is a method in which features learned in some source domain are applied to a foreign domain. In the expert advice case, transfer learning is seen via the communication of observed expert performance between tasks. For experts which perform well over multiple tasks, inter-task communication increases confidence in the expert. A given task can quickly adapt to a previous best-expert from any other task, even

if there has been no in-task exposure to that best expert. This is advantageous when a subset of best experts repeatedly performs well across tasks.

1.5 Climate Prediction

Climate models are complex mathematical models which simulate the Earth's climate. Temperature predictions can be extracted from these General Circulation Models (GCMs). GCMs are not data-driven models and do not observe historical weather patterns to generate predictions. Instead, GCMs are comprised of the mathematical representations of numerous geophysical and meteorological processes. Each GCM takes a multitude of factors into consideration, such as wind dynamics, ice-sheet distribution and rainfall. Each GCM differs due to variations in the modelling of the constituting subprocesses, and differing theories on the effect of external climate drivers (e.g. pollution).

Regional temperature anomalies are the difference in temperature of a location compared to the mean temperature for that location and month over a benchmark period. These anomalies are used in climate change literature, as they provide the most accurate reflections of temperature change. As temperature indeed changes by location and time of year, using global and annual predictions can convey temperature changes to appear smaller than they are. By averaging over locations, which may have wildly different climates even at the same point in time, absolute temperature changes may cancel one another out. In the same manner, averaging over a year may portray no annual change in spite of seasons which may have varied wildly from the mean (e.g. below-average Winter temperatures, and above-average summer temperatures). As monthly temperature anomalies avoid these pitfalls, they provide the best illustration of temperature change.

The current benchmark for any climate prediction is the mean of the GCM predictions. This does not take into account the historical GCM performance, regional disparities or seasonal disparities. It may be possible to improve upon this benchmark by implementing a multi-task algorithm with long-term memory, to generate predictions for locations across the globe. No existing research has been done to examine the regularity or identify patterns in GCM performance across locations.

1.6 Thesis Contributions

In this thesis, a series of experiments are conducted using the algorithm contributed by Herbster, Pasteris and Tse in 'Online Multi-task Learning with Long-Term Memory' [1]. First, experiments are run using synthetic data. These experiments provide empirical evidence proving the improved performance of this learner as compared to existing algorithms and advising experts. Synthetic data is generated to simulate different environments, each designed to test a specific aspect of memory. For example, long-term memory can be tested through the use of repeated best experts over time. The results are then analysed with respect to the multi-task learner's use of global, long-term memory. The insights gained from these experiments help to guide the implementation of the temperature anomaly predictor.

The multi-task learner is then applied to real-world climate data to generate temperature anomaly predictions where each task corresponds to a different location. All tasks receive expert advice from a set of GCMs and predictions are compared against historically observed ground temperature anomalies for a time period of 10-years. Data is extracted from the relevant models and a set of experiments, mirroring those completed on synthetic data, is conducted. This identifies patterns within GCM performance and indicates the best parameter choice for the learner.

By drawing comparisons to the results synthetic experiments, the performance of the learners is assessed. The identified best learner is then tuned, and the simulation is conducted again.

Experimental evidence shows that the tuned learner incurs significantly less loss than the average GCM prediction, which is the state-of-the-art. Additionally, the learner predicts better than any single expert across most locations. The algorithm however does not significantly outperform the existing learners on climate data, and the potential causes as to why are discussed alongside improvements to the model and future work.

The code for these experiments can be found at https://github.com/jess-james/SwarmExperiments.

1.7 Thesis Structure

Chapter 2 encapsulates the core concepts discussed throughout the paper, such as learning with expert advice, multi-task learning and long-term memory. Current applications of online learning models to climate data are also explored to provide context for the climate data experiments. In Chapter 3, the design and implementation of the synthetic data experiments are detailed, followed by an evaluation of the results. These experiments are repeated in Chapter 4, as applied to climate data. A final model is generated and the results are discussed. The project is then concluded with further analysis and speculation on future work in Chapter 5.

Chapter 2

Literature Review

This chapter outlines the fundamentals of online learning models to provide a foundation for the subsequent experiments, in particular algorithms for use in switching, multi-task environments. Current work applying such techniques to climate data is reviewed to establish the current research landscape and provide context for research in later chapters.

2.1 Notation

Here, notation used throughout the paper is defined. For brevity, the notation $[k] := \{1, 2, ..., k\}$ is used.

As before, there exists a task for which the learner aims to generate accurate predictions for the outcome of each trial. The ordered sequence of trials is labelled $1, ..., T$. When utilising expert advice, there exist n experts, labelled $\epsilon_1, ..., \epsilon_n$. A prediction from expert, ϵ_i at time t is written as $x_{t,i} \in \mathcal{X}$, where $\mathcal{X}$ indicates the prediction space. Predictions take the form of either a single-value or a probability distribution over a set of potential outcomes. In this paper, the single-value prediction case is used, $\mathcal{X} \subseteq \mathbb{R}$.

These expert predictions are received by the learner as a vector, $\mathbf{x}_t \in \mathcal{X}^n$. Using this input alongside prior knowledge, a learner generates its own prediction for the current trial, $\hat{y}_t \in \mathcal{X}$. Prior knowledge from previous trials can include the outcome of previous trials, and the loss incurred by the learner and experts. This knowledge is compiled in the form of a weight vector, $\mathbf{w}_t$, which represents the learner's confidence in each expert and is used to generate predictions. A learner's prediction therefore can be written as $\hat{y} = Pred(\mathbf{w}_t, \mathbf{x}_t)$.

At each trial t, after the learner submits its prediction, the trial's true outcome, $y_t \in \mathcal{Y}$, is revealed. For some predefined loss function, $L : \mathcal{Y} \times \mathcal{X} \to \mathcal{L}$, the learner incurs loss, $L_t(y_t, \hat{y}_t)$. Recall that this loss indicates the amount of error made in the prediction. Each expert incurs loss $\ell_{t,i} = L(y_t, x_{t,i})$. The expert losses for a each timestep are often compiled into one loss vector, $\boldsymbol{\ell}_t \in \mathcal{L}^n$.

A learner aims to predict accurately over the set of trials. This is equivalent to minimising its cumulative loss. When learning with expert advice, a learner minimises their loss with respect to the expert set, which is called the regret. Regret for a single trial, compared to the best expert for that trial, can be written as $\mathcal{R} = L_t(y_t, \hat{y}_t) - L(y_t, x_{t,i})$ where $i = \underset{i}{\operatorname{argmin}}\, L(y_t, x_{t,i})$.

2.2 Online Learning in Stationary Environments

2.2.1 Static Expert Learners

Static expert learners aim to identify the single best expert over the sequence of trials. A learner reduces the weight assigned to an expert at each trial according to the expert's incurred loss. Over a sequence of trials, these updates compound and provide an indicator for the best expert. Assuming a uniform prior, the expert incurring the least cumulative loss garners the most confidence from the learner. A learner's regret is then minimised with respect to that best-performing expert. These algorithms are successful in environments in which one expert performs well consistently.

An example of a static learner is the weighted majority algorithm [2]. The algorithm generates a binary output by predicting with the weighted majority (i.e. if q_0 is the sum of weights of all experts predicting 0, and q_1 for experts predicting 1 - the learner then predicts 1 if $q_1 > q_0$). All experts are initially assigned equal weights, signifying a uniform prior. When the learner predicts incorrectly, the weight of each expert in the majority group is reduced by a factor β, $0 \leq \beta < 1$. As the output space is binary, a zero-one loss is used to count the number of mistakes made by the learner. The loss bound for the weighted majority algorithm is proven to be $\mathcal{O}(\min_i L(\epsilon_i, \mathbf{y}_T) + \ln n)$. The halving algorithm is a special case of the weighted majority algorithm. Assuming the existence of a perfect expert, a learner reduces its confidence in a learner to zero if that expert predicts incorrectly. The total number of mistakes made is then less than $\log_2 n$. These algorithms require $O(n)$ space to maintain weights for each expert, and run in time linear to the number of experts.

Algorithm 1: Weighted Majority Algorithm

```
Input 0 ≤ β < 1
Initialisation  w_{1,1}, ..., w_{1,n} = 1/n
for t=1,2,..., T do
    Receive x_t
    Predict 1 if ⟨x_t, w_t⟩ > 0, otherwise predict 0
    Receive y_t
    Update:
        for i={1,...,n} do
            if x_{t,i} ≠ y_t then
                w_{t+1,i} = β w_{t+1,i}
            end
        end
end
```

The aggregating algorithm contributed by Vovk [3] provides a more flexible static learner. The aggregating algorithm as it is not bound to the binary prediction space and can work in conjunction with other loss functions. Starting with a uniform weight vector, each expert's weight is multiplied by $\exp(-\eta \ell_t^i)$ at each trial. The parameter η represents the learning rate. It follows that experts that predict correctly on a trial retain all their weight, while the confidence in incorrect expert weights is reduced proportionally to their loss. After the update, the weight vector is normalised so that for prediction using the dot product prediction function. Bayes' algorithm is a special case of the aggregating algorithm, achieved when $\eta = 1$.

Algorithm 2: Aggregating Algorithm

Parameters: $\eta > 0$
Initialise: $w_{1,1}, ..., w_{1,n} = \frac{1}{n}$
for *t=1,2,..., T* **do**
  Receive $\mathbf{x}_t$
  Predict $\hat{y}_t$.
  Receive $\mathbf{y}_t$ from environment
  Incur loss $L(\hat{y}_t, y_t)$ for the algorithm, and $\ell_{t,i} = L(x_{t,i}, y_t)$ for $i \in [n]$
  Update weights
    $\mathbf{w}_{t+1} \leftarrow \frac{\mathbf{w}_t \odot \exp(-\eta \boldsymbol{\ell}_t)}{\mathbf{w}_t \cdot \exp(-\eta \boldsymbol{\ell}_t)}$
end

The loss bounds for the aggregating algorithm, and algorithms explored later, are dependent on the loss function and prediction function used. This is due to a concept called (c, η)-realisability.

DEFINITION 1 *[4] Given $c, \eta > 0$, a prediction function $Pred(\cdot)$ and a loss function $L(y, \hat{y})$ are (c, η)-realisable if, for any normalised weight vector $\mathbf{w} \in [0, 1]^n$, input $\mathbf{x}$ and outcome y if*

$$L(y, Pred(\mathbf{w}, \mathbf{x})) \leq -c \ln \sum_{i=1}^{n} w_i e^{-\eta L(y, \mathbf{x}_i)}$$

For a (c, η)-realisable choice of prediction function and loss function, Vovk provided worst-case loss bounds of $L(\mathbf{y}, \hat{\mathbf{y}}) \leq c\eta \min_i L(\epsilon_i) + c \ln n$ [3]. This is equal to a regret with respect to the best expert that is logarithmic in the number of experts. The term $c \ln n$ indicates the error incurred while identifying the best expert, and can be reduced with a strategic of c

The two prediction functions which occur throughout this paper are the dot product prediction and the Vovk prediction function. Dot product predictions are simply the dot product of the normalised weight vector with the input. The Vovk prediction function is more sophisticated, and is detailed below.

DEFINITION 2 *[4] Given input $\mathbf{x}_t$, a weight vector $\mathbf{w}_t$, a loss function $L(y, \hat{y})$ and constants $c, \eta > 0$. Vovk's prediction function is*

$$\hat{y} = \frac{1}{2}[L_0^{-1}(\Delta(0)) + L_1^{-1}(\Delta(1))]$$

where $L^{-1}(y, \hat{y})$ is the inverse of $L(y, \hat{y})$, $L_0^{-1}(z) = L^{-1}(0, z)$, $L_1^{-1}(z) = L^{-1}(1, z)$ and $\Delta(y) = -c \ln(\sum_{i=1}^{n} w_i e^{-\eta L(y, x_i)})$.

The loss function used in later experiments is the square loss, $L(y, \hat{y}) = (y - \hat{y})^2$. The square loss is shown to be (c, η)-realisable when $\eta = 1/c$. [4] When using the square loss, the suggested c values for the dot product prediction function and Vovk prediction function are 2 and 1/2 respectively. [5].

These algorithms perform well successfully in comparison to a single expert, and do not adapt well to changes in best expert. In the aggregating algorithm, the best expert gains high weight (especially if it is the best expert for many trials). In order for a second segment to be adapted to, the best expert must incur the minimum cumulative loss. Waiting for the previous best expert to incur a higher cumulative loss than the current best expert may take many trials, if it occurs at all. This is infeasible for implementation in a switching environment, and adaptive learners are required.

2.3 Learning in Switching Environments

Unlike the previous algorithms which identify the best expert based on total historical performance, the following algorithms identify the best expert for the current segment and minimise their loss accordingly.

The best expert for trial t, can be written as ϵ_t^*, and is equal to ϵ_i where $i = \min_i \mathcal{L}(y_t, x_t^i)$. A 'switch' is said to occur at time t when $\epsilon_t^* \neq \epsilon_{t+1}^*$. For task with k switches, there are $k+1$ segments with a corresponding best expert. Note that an expert can be the best expert for multiple segments, provided they are not adjacent. For any task, its segmentation and the relevant best experts is called its partition. A task's partition is therefore noted by two vectors, $\mathbf{t}$ and $\boldsymbol{\epsilon}*$. The vector $\mathbf{t}$ is a vector representing the labels of trials in which a switch occurs. The k switch times which form the $k+1$ segments as follows; $[t_0, t_1)$, $[t_1, t_2)$, ... , $[t_{k-1}, t_k)$,$[t_k, T)$. The best expert vector is then comprised of the best experts for each of these segments, $\epsilon_0^*, \epsilon_1^*, ..., \epsilon_k^*$. Now, the aim is to minimise the loss relative to this best-expert partition.

A brute force solution is to apply the aggregating algorithm, with weights for each possible partition. Over the sequence of trials, there exist $\binom{T-1}{k} n(n-1)^k$ possible sequences of best experts. At each of the k switches, there is a choice of $n-1$ possible experts, as the next best expert cannot be the same as the current one. These k trials are spread across the $T-1$ trials, as the first trial is excluded. There are n possible best experts at the first trial. The regret for the aggregating algorithm applied to each partition is seen in Equation 2.1.

$$R \leq c\left(\ln n + k \ln\left(\frac{T-1}{K}\right) + \ln(n-1) + 1\right) \tag{2.1}$$

This shows that the regret increases with the number of trials. Additionally, this is an extremely inefficient method, in regards to space and time complexity, as the number of partitions grows exponentially with time. Algorithms such as the Fixed-Share and Variable-Share algorithms address the issue of efficiency, and provide modest regret bounds.

2.3.1 Fixed-Share Learner

The Fixed-Share algorithm is presented by Herbster and Warmuth [5]. Parameters α and η are given to the expert, representing the switching likelihood and learning rate. Weights are initially apportioned equally, and at each trial, a loss update is conducted as in the aggregating algorithm. After this update, each expert now donates a portion, α, of its weight to a pool. This pool is redistributed equally amongst the experts.

This learner models the sequence of best experts as a Markov chain, with each state as an expert. A Markov chain models an entity's state over time, using a graphical representation of all possible states as nodes. Each node is connected by an edge representing the probability of transitioning from one state to the next. In Markovian processes, these probabilities are independent of time and the entity's next state is only dependent on its current state. Given the best expert at time t, the likelihood of the best expert remaining the same at time $t+1$ is $1-\alpha$ (2.2). There is equal likelihood of switching to any of the other $n-1$ states from the current state, with probability $\frac{\alpha}{n-1}$ (2.3). As long-term memory is not employed in a Markov model, the probabilities (2.3) hold for all timesteps t.

$$P(\epsilon_{t+1}^* = \epsilon_i | \epsilon_t^* = \epsilon_i) = 1 - \alpha \tag{2.2}$$

$$P(\epsilon^*_{t+1} = \epsilon_i | \epsilon^*_t = \epsilon_j) = \frac{\alpha}{n-1}, \forall j \neq i \tag{2.3}$$

Via the redistribution of weight, the Fixed-Share algorithm is able to adapt to new experts quickly. When an expert with high weight begins to perform badly, its weight is quickly redistributed. All other experts gain a fraction of the weight in the pool. This form of short-term memory allows the expert to identify the current best expert. As weights are maintained for each expert, as opposed to each partition-expert pair, storage requirements are reduced. Both the space and time complexity are bounded by $O(n)$ [5].

Algorithm 3: Fixed–Share Algorithm

Parameters: $c, \eta > 0$, $0 \leq \alpha \leq 1$
Initialise: $w_{1,1}, ..., w_{1,n} = \frac{1}{n}$
for $t = 1, 2, ..., T$ **do**
  Receive $\mathbf{x}_t$
  Predict with any $\hat{y}$ that satisfies $L(\hat{y}, y) \leq \Delta(y)$ for y=0 and y=1
    where $\Delta(y) = -c \ln \sum_{i=1}^{n} v_{t,i} e^{-\eta L(x_{t,i}, y)}$ and $v_{t,i} = w^s_{t,i} / \sum_{i=1}^{n} w^s_{t,i}$
  Receive outcome y_t
  Incur loss $L(y_t, \hat{y}_t)$ for learner and $\ell_{t,i} \leftarrow L(y_t, x_{t,i})$ for all experts.
  Loss update
    $\mathbf{w}^m_t \leftarrow \frac{\mathbf{w}_t \odot \exp(-\eta \boldsymbol{\ell}_t)}{\mathbf{w}_t \cdot \exp(-\eta \boldsymbol{\ell}_t)}$
  Share update:
    $pool = \sum_{i=1}^{n} w^m_{t,i}$
    $\mathbf{w}_{t+1,i} \leftarrow (1-\alpha)\mathbf{w}^m_t + \frac{1}{n-1}(pool - \alpha \mathbf{w}^m_t)$
end

Loss is expressed relative to partitions with k switches, noted k-partitions. The loss of some k-partition on a sequence S is written as $L(\mathcal{P}_{T,n,k,\mathbf{t},\mathbf{e}}(S))$. T, n and k represent the number of trials, experts and switches. As before, the vectors $\mathbf{t}$ and $\mathbf{e}$ denote the partition's switch times and best experts. The loss of the algorithm can be compared to the best partition, and can be minimised by the optimal selection of α, such that $\alpha^* = \frac{k}{T-1}$. This optimal selection of α will be used in later experiments.

Modest worst-case regret bounds are provided which are logarithmic with respect to the number of experts but linear with regards to time[5]. A drawback of the Fixed-Share algorithm is that it allows perfect experts to share weight with other experts, even over long segments for which it is the best expert, as it donates weight to the pool at each trial. This makes the best-case regret bound unnecessarily high. This can be avoided through the use of the Variable-Share algorithm.

2.3.2 Variable–Share Learner

Variable-Share only requires experts to share weight when incurring large loss. This allows perfect experts to accumulate all the weight over segments in which they perform well, and still rapidly redistribute the weight once they begin to incur loss. The switching parameter, α, approximates the shifting rate per unit of loss (i.e. the number of anticipated switches divided by the anticipated loss).

Algorithm 4: Variable–Share Algorithm

Parameters: $c, \eta > 0$, $0 \leq \alpha \leq 1$
Initialise: $w_{1,1}, ..., w_{1,n} = \frac{1}{n}$
for *t=1,2,..., T* **do**
  Receive $\mathbf{x}_t$
  Predict with any $\hat{y}$ that satisfies $L(\hat{y}, y) \leq \Delta(y)$ for y=0 and y=1
    where $\Delta(y) = -c \ln \sum_{i=1}^{n} v_{t,i} e^{-\eta L(x_{t,i}, y)}$ and $v_{t,i} = w_{t,i}^{s} / \sum_{i=1}^{n} w_{t,i}^{s}$
  Receive outcome y_t
  Incur loss $L(y_t, \hat{y}_t)$ for learner and $\ell_{t,i} \leftarrow L(y_t, x_{t,i})$ for all experts.
  Loss update
    $\mathbf{w}_t^m \leftarrow \frac{\mathbf{w}_t \odot \exp(-\eta \boldsymbol{\ell}_t)}{\mathbf{w}_t \cdot \exp(-\eta \boldsymbol{\ell}_t)}$
  Share update:
    $pool = \sum_{i=1}^{n} (1 - (1-\alpha)^{\ell_{t,i}}) w_{t,i}^{m}$
    $w_{t+1,i} \leftarrow (1-\alpha)^{\ell_{t,i}} w_{t,i}^{m} + \frac{1}{n-1} [pool - (1 - (1-\alpha)^{\ell_{t,i}}) w_{t,i}^{m}]$
end

An estimate on the number of switches, $\hat{k}$ and the upper bound on k-partition loss $\hat{L}$ are used for tuning α. The loss is minimised with an optimal setting of $\alpha^* = \frac{\hat{k}}{2\hat{k}+\hat{L}}$ [5].

Efficient implementation can be done by grouping updates. Instead of calculating the weights for each expert individually, the 'pool' variable can be created and used to update all weights in a single update. Using these updates (2.4, 2.5), the computation time is reduced from $O(n)$ to $O(1)$, which provides noticeable improvement when repeated over many trials. This method of implementation is used in later chapters to improve the experiment efficiency.

$$\textbf{Fixed-Share Update:} \quad w_{t,i}^{s} = (1-\alpha) w_{t,i}^{m} + \sum_{j \neq i}^{n} \frac{\alpha}{n-1} w_{t,j}^{m} \tag{2.4}$$

$$\textbf{Variable-Share Update:} \quad w_{t,i}^{s} = (1-\alpha)^{L(y_t, x_{t,i})} w_{t,i}^{m} + \sum_{j \neq i}^{n} \frac{\left(1 - (1-\alpha)^{L(y_t, x_{t,j})}\right)}{n-1} w_{t,j}^{m} \tag{2.5}$$

While the Variable-Share algorithm can adapt to experts in a dynamic setting relatively well, it retains no memory of previous expert performance. Experts which have performed well previously are treated equally to all other experts. More sophisticated algorithms can utilise previous experiences by employing long-term memory to 'recall' previous expert performance.

2.4 Learning with Long-Term Memory

When a subset of experts consistently perform well, it is desirable to track which experts have previously performed well. Bosquet and Warmuth contribute algorithms which utilise long-term memory in Tracking a Small Set of Experts by Mixing Past Posteriors [6]. Similar to previous algorithms, a loss update takes place at each trial, now with the addition of a mixing update. The mixing update combines the weight vector as observed after the loss update, $\mathbf{w}_t^m$, with any previous final weight vectors $\mathbf{w}_{i<t}$. The degree to which previous weight vectors are 'mixed' is defined by a mixing scheme. When no previous weight vectors are mixed in, the mixing algorithm remains as the aggregating algorithm. Other modes involve a Fixed-Share learner with some form of memory. Fixed-Share to uniform past combines $1-\alpha$ of the current weight vector, and equal proportions $\frac{\alpha}{t}$ of all previous weights. Fixed-Share to decaying past combines the $1-\alpha$ of the current weight,

with 'decayed' previous weight vectors. Previous weight vectors become increasingly small the further in the past they occurred, due to the decay parameter γ. These algorithms both require the space-time complexity $\mathcal{O}(nt)$. Decay is useful when performance from many trials before is not indicative of current expert performance.

	Coefficients
Static Expert	$\beta_{t+1}(t) = 1$
	$\beta_{t+1}(q) = 0$ for $0 \leq q < t$
Fixed-Share to Start Vector	$\beta_{t+1}(t) = 1 - \alpha$
	$\beta_{t+1}(0) = \alpha$
Fixed-Share to Uniform Past	$\beta_{t+1}(t) = 1 - \alpha$
	$\beta_{t+1}(q) = \alpha \frac{1}{t}$ for $0 \leq q < t$
Fixed-Share Decaying Past	$\beta_{t+1}(t) = 1 - \alpha$ and
	$\beta_{t+1}(q) = \alpha \frac{1}{(t-q)^\gamma} \frac{1}{Z_t}$ for $0 \leq q < t$
	where $Z_t = \sum_{q=0}^{t-1} \frac{1}{(t-q)^\gamma}$ and $\gamma \geq 0$

Table 2.1: Mixing Scheme Coefficients

Algorithm 5: Mixing Algorithm

Parameters: $\eta > 0$, $0 \leq \alpha \leq 1$
Initialise: $w_{1,1}, ..., w_{1,n} = \frac{1}{n}$
for t=1,...,T **do**
  Receive $\mathbf{x}_t$
  Predict $\hat{y} \leftarrow Pred(\mathbf{w}_t, \mathbf{x}_t)$
  Receive outcome y_t
  Incur loss $L(y_t, \hat{y}_t)$ for learner and $\ell_{t,i} \leftarrow L(y_t, x_{t,i})$ for all experts.
  Loss update:
    $w_{t,i}^m = \frac{w_{t,i} e^{-\eta L(y_t, x_{t,i})}}{\sum_{j=1}^n w_{t,j} e^{-\eta L(y_t, x_{t,j})}}$
  Mixing update:
    Set mixture coefficients $\boldsymbol{\beta}$ according to Table 2.1.
    $\mathbf{w}_{t+1} = \sum_{q=0}^t \beta_{t+1}(q) \mathbf{w}_q^m$
end

The Mixing algorithm is useful in that the long-term memory provides tighter loss bounds, and is beneficial when experts are repeated. However, as the complexity is dependent on time, it is unsuitable for situations in which a learner must predict over many trials.

2.5 Multi-task Learning

A novel algorithm is presented in Online Multi-task Learning with Long-Term Memory [1]. This algorithm utilises expert advice, with each of the n experts providing predictions across s tasks which are labelled $1, ..., S$. Each task i consists of T^i local trials, and k^i switches. Therefore, there are $T = \sum_{i=1}^s T^i$ global trials, and $k = \sum_{i=1}^s k^i$ global switches. In this setting, global refers to the interaction with all tasks.

Each global trial addresses a single task which is chosen according to a task vector $\mathbf{v}$. At global time τ, the local time of the task is then noted as $t = \boldsymbol{\sigma}(\tau) = \sum_{l=1}^\tau [\mathbf{v}^\tau = \mathbf{v}^l]$ indicating the number of previous global trials that targeted the current task. For each global trial τ, experts provide

predictions for the current task $\mathbf{v}^\tau$. A learner receives $\mathbf{v}^\tau$ and the set of expert predictions. Using this input, the learner generates its own prediction.

A set of s task partitions form the s-task-partition, where i represents the partition for task i. For each task i, $\boldsymbol{\epsilon}^i$ represents the order of best experts and $\mathbf{t}^i$ is comprised of the k^i local switch times. As before, they present the partition $[t_1^i, t_2^i), ..., [t_{k^i}^i, t_{k^i+1}^i)$ where the corresponding expert for each segment k_m^i is $\boldsymbol{\epsilon}_m^i$. A learner should minimise its loss with respect to this global partition. Assuming the tasks are interrelated and that best experts may be common across tasks, the use of global memory helps to reduce the learner's loss by communicating observed expert performance between tasks. A task can quickly adapt to a best expert it has no prior exposure to, given that it has performed well in other tasks. Inter-task communication increases the confidence in experts that perform well across tasks.

Each best expert can be called a mode, with the parameter m representing the number of modes. It follows that $m = |\cup_{i\in[s]} \cup_{j\in[k+1]} \boldsymbol{\epsilon}_m^i|$. If a subset of experts consistently outperform the rest, this subset can be identified and prioritised over other experts. This is especially beneficial when $m << k$.

Algorithm 6: Online Learner for Predicting $\mathcal{H}_{\text{fin}}$

Parameters: $\mathcal{H}_{fin} \subseteq \{-1,1\}^{\mathcal{X}}; s, m, k, T \in \mathbb{N}$

Initialise:

$n := |\mathcal{H}_{\text{fin}}|; \boldsymbol{\pi}^1 \leftarrow \frac{1}{n}; \mu := \frac{1}{m}; \mathbf{w}_1^1 = ... = \mathbf{w}_1^s \leftarrow \mu\mathbf{1}; \theta := 1 - \frac{k}{T-s}; \phi := \frac{k}{(m-1)(T-s)}$ and

$\eta := \sqrt{\left(m\log(\frac{n}{m} + smH\frac{1}{m} + (T-s)H(\frac{k}{T-s}) + (m-1)(T-s)H\left(\frac{k}{(m-1)(T-s)}\right)\right)\frac{2}{T}}$

for $\tau = 1, ..., T$ **do**

Receive task, $i \leftarrow \ell^\tau \in [s]$

Receive $x^\tau \in \mathcal{X}$

$i \leftarrow \ell^t; t \leftarrow \boldsymbol{\sigma}(\tau)$

Predict:

$\mathbf{v}^\tau \leftarrow \frac{\boldsymbol{\pi}^\tau \odot \mathbf{w}_t^i}{\boldsymbol{\pi}^\tau \cdot \mathbf{w}_t^i}$, $\hat{h}^\tau \sim \mathbf{v}^\tau$, $\hat{y} \leftarrow \hat{h}^\tau(x^\tau)$

Receive $y^\tau \in \{-1, 1\}$

Update:

i) $c_i^t = L(\mathbf{X}_t^i, y_t)$ ii) $\boldsymbol{\delta} = w_t^i \odot \exp(\eta \mathbf{c}^t)$

iii)$\beta = (\boldsymbol{\pi}^t \cdot \mathbf{w}_t^i)/(\boldsymbol{\pi}^t \cdot \boldsymbol{\delta})$ iv)$\boldsymbol{\epsilon} = \mathbf{1} - \mathbf{w}_t^i + \beta\boldsymbol{\delta}$

v) $\boldsymbol{\pi}^{t+1} = \boldsymbol{\pi}^t \odot \boldsymbol{\epsilon}$ vi) $\mathbf{w}_{t+1}^i = (\phi(\mathbf{1} - \mathbf{w}_t^i) + \theta\beta\delta) \odot \boldsymbol{\epsilon}^{-1}$

The aggregating algorithm could be applied to each global partition, but this is again infeasible and requires exponential space over trials. Instead, this multi-task learner requires space-time complexity which is linear in the number of experts. By setting the optimal parameters, $\theta_0^*(w) = \frac{1}{m}, \alpha^* = \frac{k}{T-s}, \beta^* = \frac{k}{(m-1)(T-s)}$, the regret bound becomes is logarithmic in respect to time T, support its suitability for use in real-world applications which occur over many trials.

This algorithm employs global, long-term memory which is examined thoroughly in Chapter 3. Global memory involves inter-task communication, where confidence in experts is shared across tasks. This is beneficial for situations in which the same expert performs well simultaneously across trials. Long-term memory is employed, with the improvement upon the mixing algorithm, in that previous weights need not be stored. Thus, the storage requirements are significantly reduced and are linear in the number of experts. Historical expert performance is additionally communicated across tasks via global long-term memory. The learner can adapt to the expert in a specific task, even if the expert has not yet performed well in the given task, providing it has performed well

globally. For situations in which the distributions of best experts per task are related, global long-term memory should lead to a learner incurring less loss than its single-task or short-term memory counterparts.

2.6 Climate Modelling

In Chapter 4, online learning algorithms are applied to real-world climate data. This section outlines existing literature to provide the research landscape and review existing techniques inform the later experiments.

The current climate crisis has made understanding the global climate an international priority. Offline machine learning is often utilised within climate research; many of these research efforts employ data mining or neural nets [7] to analyse both the causes[8] and effects [9, 10] of climate change.

One technique to predict the extent of the rising global temperature, entails simulating the climate via mathematical models of various geophysical and meteorological processes (e.g. rainfall, ocean circulation, wind dynamics). Multiple research institutions generate these sophisticated General Circulation Models (GCMs), but there exists no consensus on a single best model or how to generate one. There is a high variance amongst these models' predictions, due to the differing implementations of the constituting physical processes and the differing emphases placed on external drivers. The mean prediction (over all models) predicts more accurately than any single model [11]. This phenomenon is unexplained. The rule of large numbers need not apply as the models are not randomly sampled over the set of all possible models. Models contain many similarities in the internal modelling and may share biases, meaning the errors between models may not implicitly cancel out. Despite this, the average of the models is currently used as the benchmark estimator.

GCMs have many drawbacks. Each model has biases caused by implementation decisions that differ from real climate behaviour. The simulations are not synchronized with observed data and are based solely on informed assumptions. This is in contrast to weather forecasting algorithms which accept input regarding the current climate. Furthermore, these models are chaotic, meaning miniscule value changes (e.g. rounding) cause great change to the rest of the model. Each model also has a varying sensitivity to external climate drivers. Even when models are run using the same data (e.g. regarding expected pollution), the models may not converge to the same result. Knowledge of these climate drivers is also uncertain which provides further difficulty in predicting future climate.

Although each model has flaws, the set of GCMs seem to predict well. Applying online learning methods to an expert set of GCMs can help to provide even more accurate predictions. In 'Tracking Climate Models'[12], Monteleoni presents a meta-learner Learn-α to predict temperature anomalies. A temperature anomaly is the difference between a 'standard' temperature (taken in a specific location, over a benchmark period) and the observed temperature. Temperature anomalies are often used to forecast climate change, by demonstrating the temperature increase per location. As anomalies are calculated in comparison to a benchmark period, anomalies can be easily compared across locations. This cannot be done using raw ground temperatures, as ground temperatures vary greatly across different regions.

Each sub-learner of Learn-α is an implementation of the Fixed-Share algorithm, using a set of GCMs as expert advice. The sub-learners have varying α values. Monteleoni's meta-learner tracks the optimum learner and thus optimum hyperparameters for the Fixed-Share algorithm.

The parameters of the sub-algorithms are noted as $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_j]$. The meta-learner is also an implementation of Fixed-Share with $\alpha = 0$. Monteleoni's model successfully outperforms the average prediction of the GCMs in multiple experiments, which was the current state-of-the-art.

Algorithm 7: Learn-α Algorithm

Input: Set of models, M_i, $i \in [n]$, which predict M_i^t at time t $\boldsymbol{\alpha} \in [0,1]^m$
Init : $p_1(j) \leftarrow \frac{1}{m}, \forall j \in [m]$, $p_{1,j}(i) \leftarrow \frac{1}{n}, \forall i \in [n], \forall j \in [m]$
for $t = 1, ..., T$ **do**
  for $i = 1, ..., n$ **do**
    $Loss_i = (y_t - M_i^t)^2$
  end
  for $j = 1, ..., m$ **do**
    LossPerAlpha[j] $\leftarrow -\log \sum_{i=1}^{n} p_{t,j}(i) e^{-Loss[i]}$
    $p_{t+1}(j) \leftarrow p_t(j) e^{-LossPerAlpha[j]}$
    for $i = 1, ..., n$ **do**
      $p_{t+1,j}(i) \leftarrow \sum_{k=1}^{n} p_{t,j}(k) e^{-Loss[k]} P(i|k; \alpha_j)$
      $P_{t+1,j} \leftarrow P_{t+1,j} / \sum_{i=1}^{n} p_{t+1,i}(j)$
      PredictionPerAlpha[j] $\leftarrow \sum_{i=1}^{n} p_{t+1,j}(i) M_i^{t+1}$
    end
  end
  $P_{t+1} \leftarrow P_{t+1} / \sum_{j=1}^{m} P_{t+1}(j)$
  Prediction $\leftarrow \sum_{j=1}^{m} p_{t+1}(j)$ PredictionPerAlpha[j]
end

Using models taken from the CMIP3 Database and true ground temperature values from the NASA GISTEMP database, the Learn-α algorithm is applied to predicting annual temperature anomalies. The model achieves higher accuracy when predicting temperature anomalies rather than raw ground temperatures. By predicting anomalies, a learner need not learn the underlying weather patterns for the area.

Montelioni's application of the Learn-α algorithm outperformed the average of the models, in both hindcast and forecast scenarios. In annual experiments, Learn-α performed better than the best model in all but one experiment, in which it performed only slightly worse. Learn-α also suffered less loss than Fixed-Share(α) learners (for all α values used in Learn-α)

Another way to improve predictions includes smoothing over areas, as shown in 'Global Climate Model Tracking using Geospatial Neighborhoods' [13]. A smoothing parameter β between 0 and 1, is used to define the level of influence each region has on its neighbours. Setting β to 0 indicates no location smoothing (i.e. original Learn-α predictions) whereas a value of 1 indicates an area's prediction is generated solely through the models' predictions for neighbouring cells. Here, cells define geographical regions of varying sizes.

Algorithm 8: Neighbourhood-Augmented Model Tracking Algorithm

Input: Set of geographic sub-regions, indexed by $r \in \{1, ..., q\}$, that collectively span the globe and the region weights, w_r Set of models, M_i, $i \in \{1, ..., n\}$, which predict $M^t_{i,r}$ at time t for region r. $\boldsymbol{\alpha} \in [0,1]^m$ $\beta \in [0,1]$

Init : $p_{1,r}(j) \leftarrow \frac{1}{m}, \forall j \in \{1, ..., m\}$, $p_{1,r,j}(i) \leftarrow \frac{1}{n}, \forall i \in \{1, ..., n\}, \forall j \in \{1, ..., m\}$

```
for t=1,..., T do
    for r=1,..., q do
        Predict according to equation 1.
        for i=1,..., n do
            Loss_i = (y^t_r − M^t_{i,r})^2
        end
        for j=1,...,m do
            LossPerAlpha[j] ← −log Σ_{i=1}^n p_{t,j,r}(i) e^{−Loss[i]}
            p_{t+1,r}(j) ← p_{t,r}(j) e^{−LossPerAlpha[j]} for i=1,...,n do
                p_{t+1,j,r}(i) ← Σ_{k=1}^n p_{t,j,r}(k) e^{−Loss[k]} P(i|k; α_j, β, r, t)
            end
            P_{t+1,j,r} ← P_{t+1,j} / Σ_{i=1}^n p_{t+1,j,r}(i) PredictionPerAlpha[j,r]
                ← Σ_{i=1}^n p_{t+1,j,r}(i) M^{t+1}_{i,r}
        end
        P_{t+1,r} ← P_{t+1,r} / Σ_{j=1}^m p_{t+1,r}(j)
        Prediction[r] ← Σ_{j=1}^m p_{t+1}(j) PredictionPerAlpha[j,r]
    end
    GlobalPrediction ← Σ_{r=1}^q w_r Prediction[r]
end
```

$$Z = \frac{1}{\alpha} \sum_{i \in \{1,...,n\} \setminus \{k\}} \left[(1-\beta) + \beta \frac{1}{|S(r)|} \sum_{s \in S(r)} P_{expert}(i,t,s) \right] \tag{2.6}$$

$$P(i|k; \alpha_j, \beta, r, t) = \begin{cases} (1-\alpha) & \text{if } i = k \\ \frac{1}{Z}[(1-\beta) + \beta \frac{1}{|S(r)|} \sum_{s \in S(r)} P_{expert}(i,t,s)] & \text{if } i \neq k \end{cases} \tag{2.7}$$

Using neighbourhood influence proved successful, as all runs where $\beta = 1$ outperformed those where $\beta = 0$. Learn-α was constantly outperformed by the neighbourhood-augmented model. This indicates that global memory could prove beneficial and inspires the application of a multi-task learner to predicting temperature anomalies across multiple locations. As the benefits of adaptive algorithms were proven in these papers (conducted using the same experimental design and data), this design is mirrored in the subsequent experiments in this paper. While both these models achieved modest success, neither account for historical GCM performance or performance across regions. Through the application of the multi-task learner, global, long-term memory may help to improve the predictions of these models.

Regions in these experiments were simply selected by coordinates, meaning each region does not accurately reflect the Earth's subclimates. For example, the results for regions overlapping the North or South pole are often skewed. This caveat is avoided in this paper, via selecting smaller

zones based within decided regions.

Chapter 3

Synthetic Experiments

3.1 Motivation

This chapter details experiments conducted using the multi-task learning algorithm (Algorithm 12)[1], which will now be referred to as the Swarm algorithm. The goal of these experiments is to empirically prove the benefits of long-term global memory, as used by the Swarm. This will be done by comparing the loss of the Swarm learner to that of Fixed- and Variable-Share learners (Algorithms 4 and 5). Learner performance are further compared to the best expert, best partition and average expert.

The aim of these experiments is to highlight the benefits of long-term memory and global memory as well as their combined effect. For each experiment, artificial data is generated to simulate a specific type of environment. Each environment will require a certain facet of memory to be used in order for the learner to perform well, therefore the use of model memory is assessed. Over the series of experiments, all environment variations will be tested. All the algorithms utilise short-term memory and so their basic functionality of best expert identification is confirmed via an introductory experiment. In the following experiments, the advantages of global and long-term memory are highlighted via the comparison of the Swarm to a Variable-Share learner. The Variable-Share learner is chosen as a baseline for comparison as it is a dynamic, expert advice algorithm with similar short-term memory to the Swarm. As the algorithms short-term single-task memory is algorithmically similar, the Variable-Share is an appropriate comparison to highlight the Swarm's memory benefits - beyond short-term memory.

In the most basic setting, there is a single task with no switching and a single best expert. A learner will prove its use of short-term memory by identifying the current best expert. Performing well over this single segment entails identifying the best expert and maintaining its confidence. Any learner employing short-term memory successfully will have a loss which converges as the best expert is learned. This setting can be used as a 'sanity check' that verifies the basic functioning of the short-term memory of all algorithms.

Switching environments test both the dynamic, short-term memory and long-term memory. In a switching setting with a sequence of distinct best experts, the ability of a learner to adapt to new experts is observed which enabled by dynamic, short-term memory. We can measure the learner's adaptability through the number of trials taken to converge after a switch and the loss incurred during that period.

By simulating a switching environment in which best experts occur repeatedly, the use of long-term memory is tested. Long-term memory allows a learner to remember the previous best experts.

The use of long-term memory allows a learner to 'recall' the historical performance of the expert, and quickly adapt to those experts with positive observed performance. This can be further proved by using a small pool of best experts as the confidence in a best expert is compounded over multiple segments for which it performs well, leading to an even smaller learner loss.

A multi-task setting will test the benefits of global memory. Global memory allows communication and memory transfer between tasks. It follows that a best expert observed in one task is communicated to the other tasks meaning tasks can quickly adapt to best experts without prior in-task exposure to that expert. Additionally, confidence is compounded when multiple tasks have the same best expert. The functionality of these learners is verified in the simplest multi-task case, by using a distinct expert for each task, and observing each task identify the best learner. By simulating a stationary multi-task environment with repeated experts (i.e. at least two tasks have the same best expert), inter-task communication is tested. By communicated the best experts between tasks, learners can build confidence in the given expert across appropriate tasks.

In a switching multi-task setting, global long-term memory provides an advantage when a subset of experts perform repeatedly well across tasks. This is because the task's history is shared. By simulating environments with repeating expert subsets across tasks, the effect global long-term memory is observed.

The observed behaviour in these experiments can provide insight into the behaviour of GCMs in the following chapter. As environment types are exhausted, the multi-task learner's performance on the real data can be used to infer information about the GCM distribution by identifying the environment the Swarm performed most similarly in.

3.2 Experimental Design

3.2.1 Synthetic Data Generation

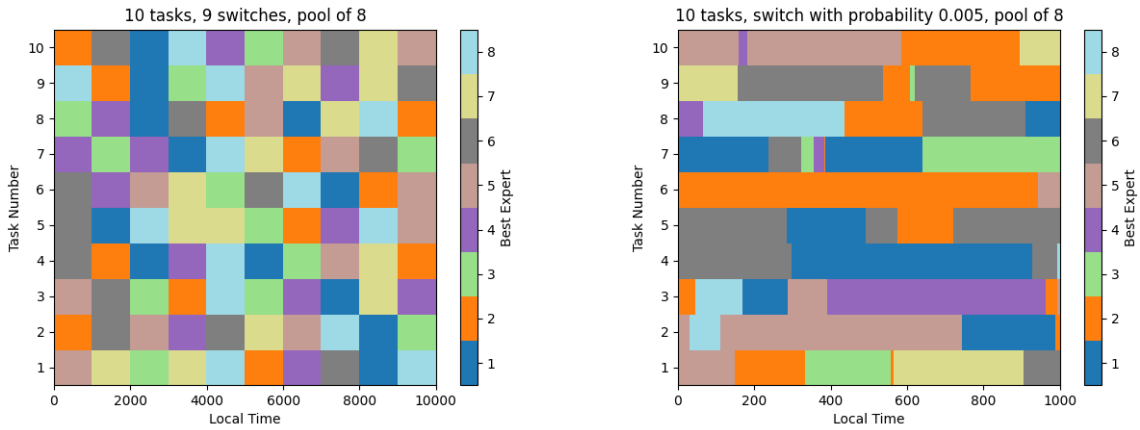
Each task is of the same length, $\frac{T}{s}$. Following the experimental design used in 'Mixing Past Posteriors'[6], the true outcomes, Y_t, are sampled randomly such that the outcome at each time step is a binary random variable. After generating Y, all non-best-expert predictions are sampled from the appropriate prediction space (i.e. (0,0.5) for $y_t = 0$, and (0.5,1) otherwise), in order for all learners to predict relatively well. To form the best expert, predictions are sampled from a smaller range concentrated toward the true outcome. These predictions are also uniform random variables over the space (0,0.15) when $y_t = 0$, and over the space (0.85, 1) otherwise. Generating expert predictions using this method, allows the best expert to outperform all other experts over designated segments. All experts still predict correctly, albeit with higher loss on average. In total, 100 experts provide advice during the following experiments. The data generation guidelines are outlined below.

$$Y \in \{0,1\}^T \text{ such that } P(Y_t = 1) = 0.5, \forall t = \{1, ..., T\}$$

$$X_{i,t} \sim \begin{cases} U(0.85, 1), & \text{if } \epsilon_t = i \text{ and } Y_t = 1 \\ U(0, 0.15), & \text{if } \epsilon_t = i \text{ and } Y_t = 0 \\ U(0.5, 1), & \text{if } \epsilon_t \neq i \text{ and } Y_t = 1 \\ U(0, 1), & \text{if } \epsilon_t \neq i \text{ and } Y_t = 0 \end{cases}$$

As seen above, there is a set of modes $[\exists i \exists t \epsilon^*_{i,T}]$. To create the order of modes (and thus

best experts) for each task, the task must be split into segments. To segment the task, a switch parameter must be specified representing either the number of trials after which to switch or the probability of switching modes on any given trial. For the former, the order of experts is simply generated by sampling the modes uniformly with replacement for k+1 partitions. For the probabilistic switching (as shown in Figure 2b), to generate the best expert for each trial, a binary random variable is sampled, with probability p of returning 1. Upon the variable being sampled as 1, the next expert is randomly chosen from the set of remaining modes (excluding the current mode). In Figure 3.1a, each task switches every 1000 local time steps to the next best expert. In Figure 3.1b, each task switches best expert with probability 0.005, over 1000 local trials.



(a) Example of uniform switching across tasks.

(b) Example of switching with probability p.

Figure 3.1: Types of switching

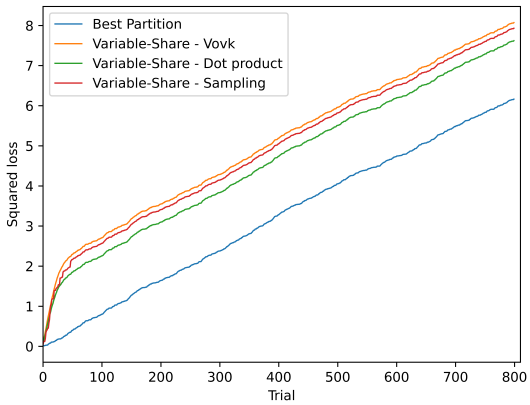
3.2.2 Implementation

Each experiment consists of n experts, m modes, s tasks and T trials. A task vector, $\mathbf{v}$, is used to define the order in which each task is updated. The task vector is simply created by via shuffling a list of task labels. Each task label, $i \in [s]$, occurs in the list with frequency equal to the number of local trials in i.

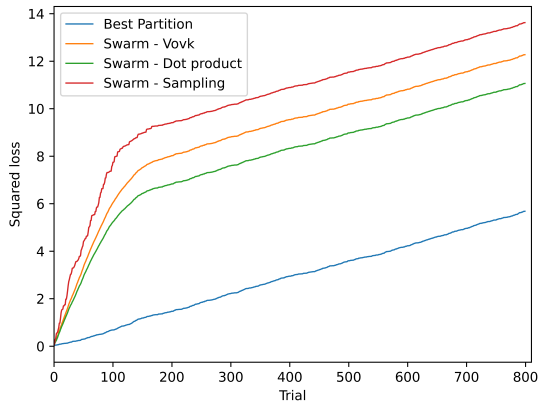
At each global trial τ, the current task, i, is set to $\mathbf{v}_\tau$. All multi-task learners receive the current task i, and the expert predictions $\mathbf{X}_t$. After learners output predictions, y^t is revealed. The loss for each learner is recorded, both total and per-task. The square loss is used throughout the experiments (multiple experiments were repeated using the log loss for comparison).

For the Fixed-Share algorithm, α is set to $\frac{m}{T-1}$. For the Variable-Share algorithm, α is set to $\frac{\hat{k}}{2\hat{k}+\hat{L}}$.

Preliminary experiments were conducted to compare the performance of various prediction functions; predicting with $\mathbf{v}^t \cdot \mathbf{x}^t$, sampling from $\mathbf{x}^t$ using a probability distribution defined by $\mathbf{v}^t$ or the Vovk prediction function. These experiments used synthetic data (generated according to the above scheme), and the parameters $s = 1, k = 0, m = 1, n = 64, T = 800$, which represented a sequence of 4 distinct experts. The Vovk prediction function is chosen for use in the experiments as it approximates the average performance of all three functions (as seen in Figure 3.2).



(a) Variable-Share Algorithm.



(b) Swarm Algorithm.

Figure 3.2: Algorithm Loss per Prediction Function

3.2.3 Parameter Selection

The α values for the Fixed-Share and Variable-Share algorithms are chosen optimally, to reduce loss. For Fixed-Share, $\alpha = \frac{k}{T-1}$. For Variable-Share, the number of switches and the anticipated loss are specified. The anticipated loss is defined by $\alpha = \frac{k}{2k\tilde{L}}$. Parameters for the Swarm algorithm are set as follows:

$$\alpha = \frac{k}{T-s}; \beta = \frac{k}{(T-s)(m-1)}$$

$$\boldsymbol{\pi}^1 = \frac{1}{n}, \mu = \frac{1}{m}, \mathbf{w}_1^i = ... = \mathbf{w}_1^s = \mu\mathbf{1}$$

$$\theta = 1 - \frac{k}{T-s}, \phi = \frac{k}{\max\{1,(m-1)\}(T-s)}$$

As the square loss and Vovk prediction function are used, the parameters c and η are initially set as 1/2 and 2 respectively, as suggested by the literature[5]. Yet, smaller values of c can be used for lower incurred loss.

DEFINITION 3 *For any observed S and expert i, the loss of algorithm A is bound by*

$$L(S, A) \leq -c \ln w_{\ell+1,i}^s \tag{3.1}$$

when A is the Static Expert, Fixed-Share or Variable-Share algorithm and both the loss function L and prediction function are (c, η)-realizable.

Experiments were conducted to verify the suitability of η, and therefore c, in this artificial setting and ensure a fair comparison to the other algorithms. Clearly, utilising a low c value will provide the Swarm with a baseline advantage. To maintain (c, η)-realizability, $c = \frac{1}{\eta}$ throughout the experiments. In the following experiments, a sequence of 4 distinct experts ($s = 1, k = 3, m = 4, n = 64, T = 800$) is simulated to determine a suitable value of η. η values were tested over the range [0,100]. Detailed results of these experiment can be found in the appendix. The suggested value of $\eta = 2$ matched the performance of the Fixed-Share algorithm, falling slightly behind the Variable-Share algorithm. In the following synthetic experiments, $\eta = 2$ is used, to provide a fair comparison.

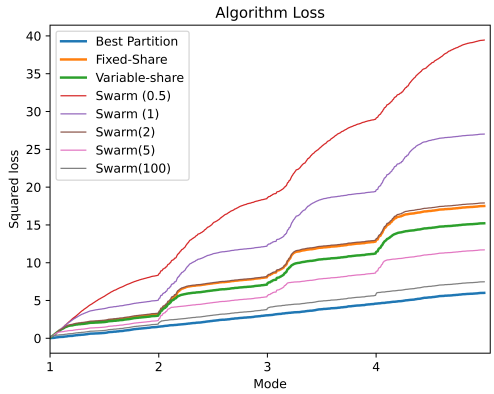


Figure 3.3: Single Task, No-Switching Loss

3.3 Results

The following section provides the results of the experiments. Each experiments parameters are listed, along with the loss of each expert over the T trials. For multi-task experiments, the loss provided is the tasks' average over time.

3.3.1 Single Task, No Switching

In the most basic case, one best expert is simulated in a single task setting. All learners incur almost exactly the same loss, as shown by the green line in Figure 3.3. The learners successfully adapt to the best learner. This is shown by the learner loss increasing at the same rate as the best expert after a short learning period. The learner loss is slightly higher than that of the best expert, but significantly lower than the average expert. The Variable Share algorithms and the Swarm adapt to the best expert identically in terms of time and loss. The basic short-term memory of the learners is verified, as both learners successfully identified the expert.

i) Parameter set: $(n = 100, T = 200, s = 1, k = 0, m = 1)$.

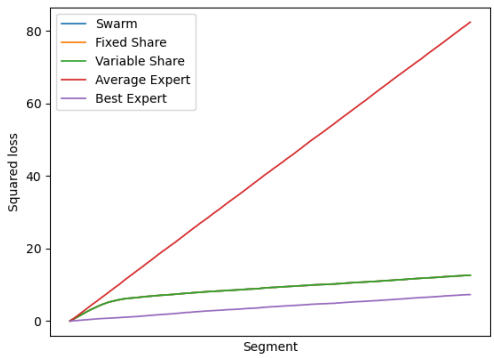


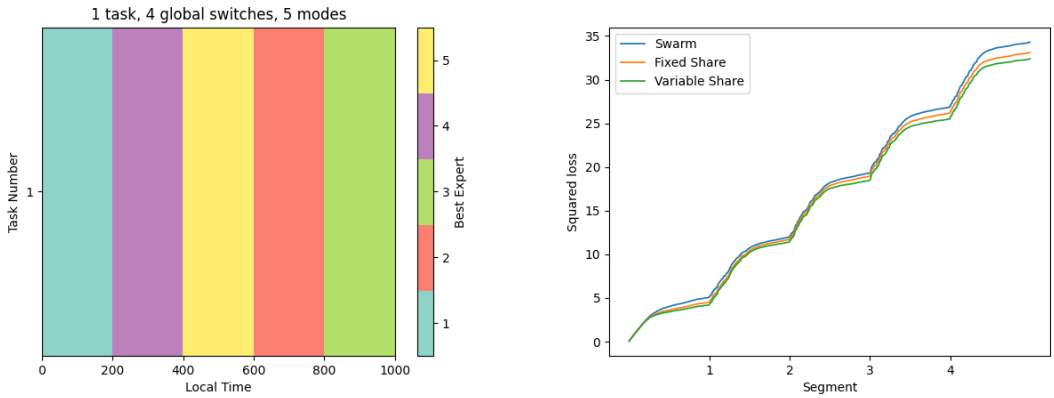
Figure 3.4: Single Task, No-Switching Loss

3.3.2 Single Task, Switching

Distinct Experts

The algorithms are now run on a sequence of 5 distinct experts in a single task. A visualisation on the expert assignment is shown in Figure 3.4a. The experiment results are shown in Figure

3.4b. Each learner incurs loss at the beginning of each segment. The learner loss then plateaus once the algorithm has gained sufficient confidence in the best expert for the current segment. The loss incurred during each segment is roughly equal, as the experts are modelled as a Markov chain. The loss incurred in each segment indicated the learner's readjustment after each switch. All three algorithms perform similarly, with Variable-Share incurring the least loss and the Swarm algorithm incurring slightly more. This experiment confirms the dynamic, short-term memory of all algorithms in the switching scenario. ii) Parameter set: $(n = 100, T = 1000, s = 1, k = 4, m = 5)$.



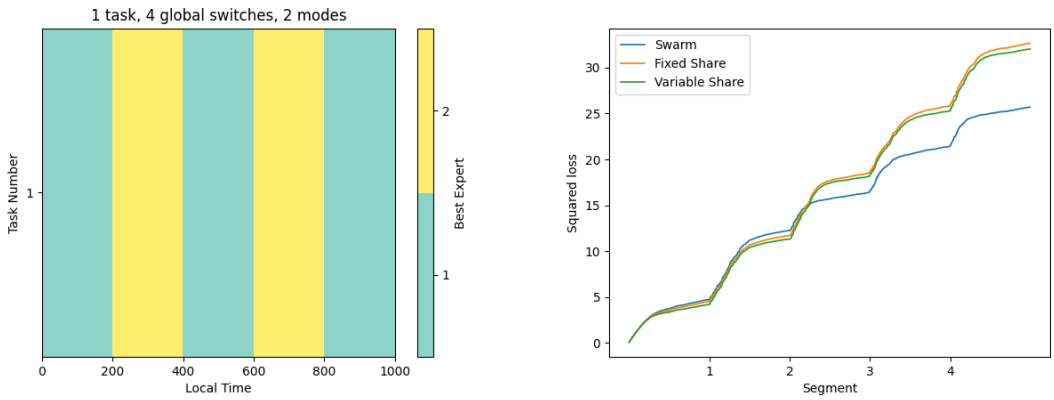
(a) Expert Assignment

(b) Loss

Figure 3.5: Single Task Switching with Distinct Experts

Repeated Experts

The pool size is reduced to 2, simulating switching environments in which a subset of the experts perform well repeatedly. Figure 3.5b shows that the Swarm is slightly inferior to its counterparts over the first two segments, reflecting its performance in the previous experiment. At the second switch, there is the first instance of expert repetition. After the second switch, the Swarm incurs about half the per-segment loss of the Variable-Share learner. This per-segment loss is significantly lower than the Swarm's own loss in the first 2 segments. This is due to the use of long-term memory. As the confidence in expert 1 gained from the first segment is retained, despite relatively poor performance in the second segment. This effect is mirrored at the third switch, as Expert 2 is repeated. At the final switch, the Swarm learner incurs the least loss of any segment. By this time, Expert 1 has outperformed the other experts for two segments, and the Swarm learner has higher confidence in expert 1 compared to the second switch. The Swarm significantly outperforms the Variable-Share learner by the end of the experiment. This confirms that long-term memory reduces loss when best experts are repeated. iii) Parameter set: $(n = 100, T = 1000, s = 1, k = 5, m = 2)$.



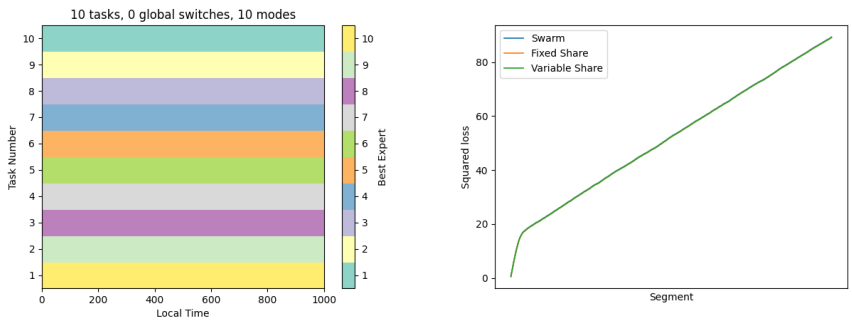
(a) Expert Assignment (b) Loss

Figure 3.6: Single Task Switching with Repeated Experts

3.3.3 Multi-Task, No Switching

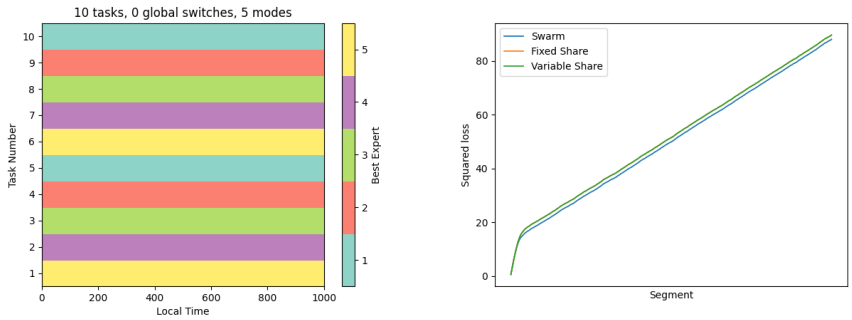
A stationary multi-task setting is now simulated. Each task has a single best expert across all trials. In the first experiment, operating with a distinct expert per task, each expert performs nearly identically, as shown by the green line in Figure 3.7b. Each algorithm learns the best expert per task, after which the loss incurred per trial decreases. This confirms that the multi-task implementation of the Variable-Share learner performs comparably to the Swarm in a multi-task scenario where inter-task communication provides no additional information.

In the second part of this experiment, the pool size is reduced to 5, as seen in Figure 3.8a. The Swarm learner incurs slightly less loss than in the previous experiment, while Variable-Share's performance remains the same. The Variable-Share algorithm does not communicate between tasks. Therefore, reducing the pool size has no impact on the performance as no additional information is gained, despite best experts being observed across multiple tasks. Each expert is shared across 2 tasks and inter-task communication allows the best expert confidence to be shared with other tasks, leading to the slight reduction in loss. As the number of modes is reduced further, the gap in learner performance widens as seen in Figure 3.9b. This is due to the confidence in each expert being shared across 5 tasks. As the learners converge predictions converge to that of the best expert, the regret can be seen by the initial loss gain before the gained loss matches that of the best expert. The regret can be seen to be approximately half that of the Variable-Share learner. iv) Parameter sets: $(n = 100, T = 1000, s = 10, k = 0, m = [10, 5, 2])$.



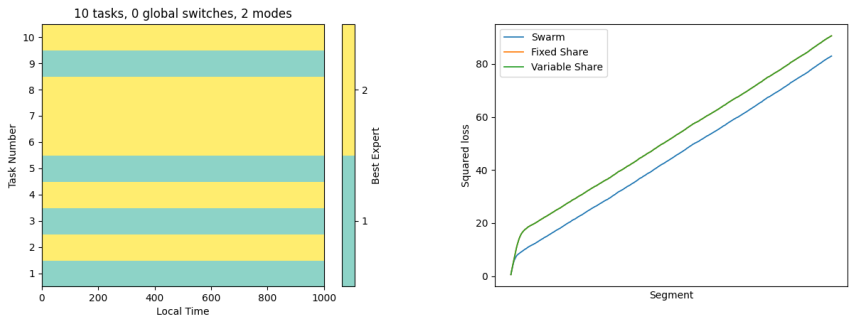
(a) Expert Assignment (b) Loss

Figure 3.7: Multi-Task No Switching with $m = 10$



(a) Expert Assignment (b) Loss

Figure 3.8: Multi-Task No Switching with $m = 5$



(a) Expert Assignment (b) Loss

Figure 3.9: Multi-Task No Switching with $m = 2$

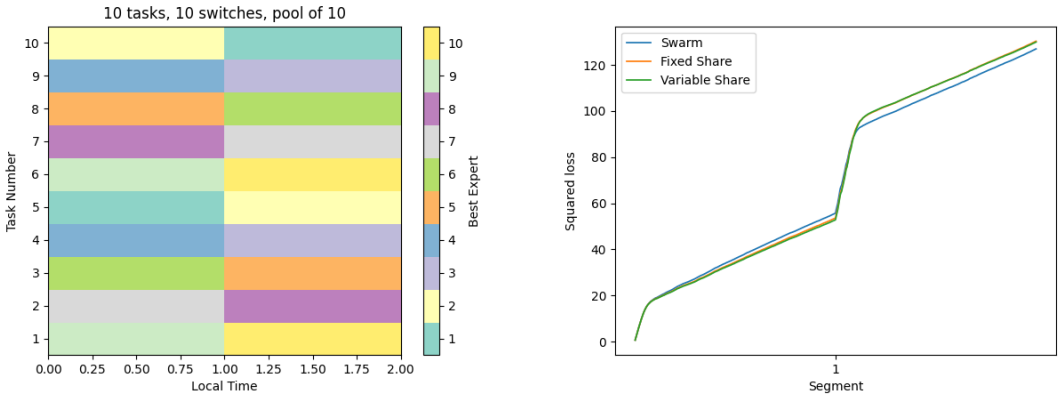
3.3.4 Multi-Task, Switching

In this set of experiments, each setting consists of 10 tasks, and 10 switches. This mimics an environment in which each task has one switch. In each experiment, the Swarm outperforms the Variable-Share algorithm. The repetition of each expert once, as seen in Figure 3.10b, provides a small advantage for the Swarm against the Variable Share algorithm. In the first segment, the Variable-Share algorithm performs better. In the second segment, under the observation of a repeated best expert albeit from a different task, the Swarm incurs less loss than the Variable-Share learner.

Reducing the number of modes to 5, the Swarm closes the loss difference in the first segment to a negligible difference. As some experts are repeated across tasks, the global memory proves beneficial. The greater difference in loss is seen after the switch, in which the Swarm incurs far less loss as previously seen experts are repeated. The memory transfer in the Swarm algorithm allows the confidence in the pool to be shared across tasks, which also leads to the Swarm algorithm loss plateauing earlier.

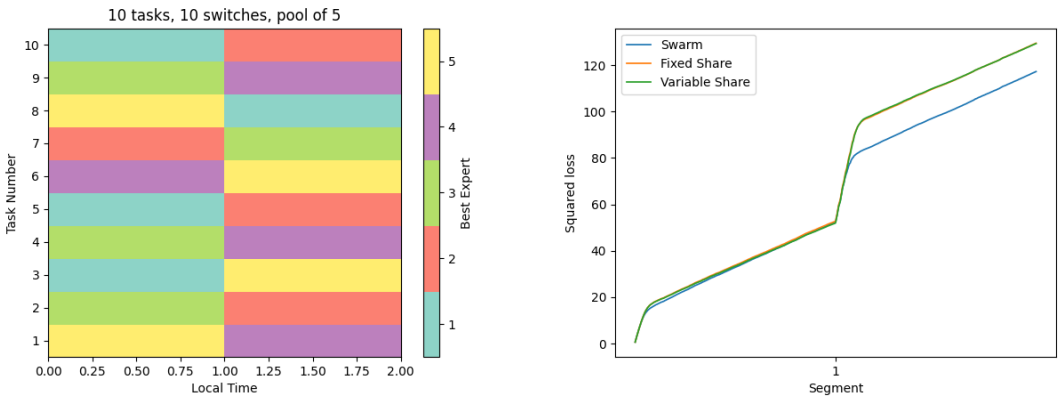
The effect is emphasised further with $m = 2$. The Swarm incurs less loss than the Variable-Share learner in both segments. The regret observed in the first segment is about half that of the Variable-Share learner, and is even smaller in proportion in the second segment. This demonstrates the benefits of global memory.

v) Parameter sets: $(n = 100, T = 1000, s = 10, k = 10, m = [10, 5, 2])$.



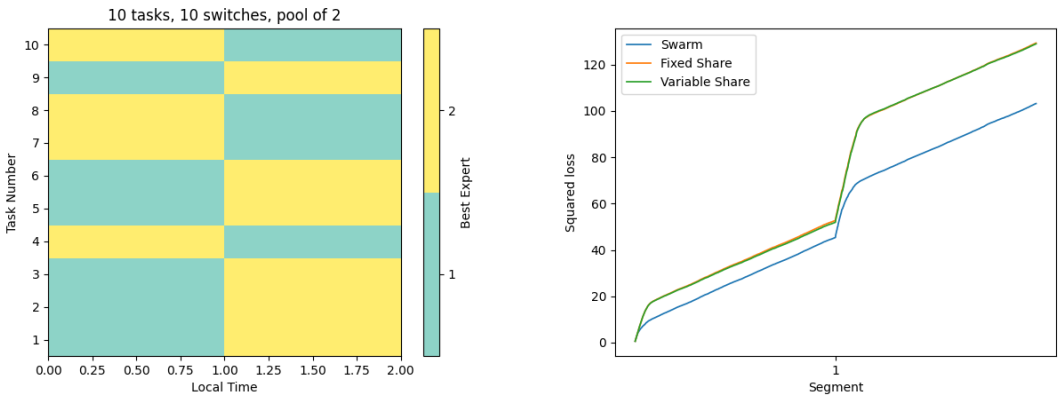
(a) Expert Assignment (b) Loss

Figure 3.10: Multi-Task Switching with $m = 10$



(a) Expert Assignment (b) Loss

Figure 3.11: Multi-Task Switching with $m = 5$



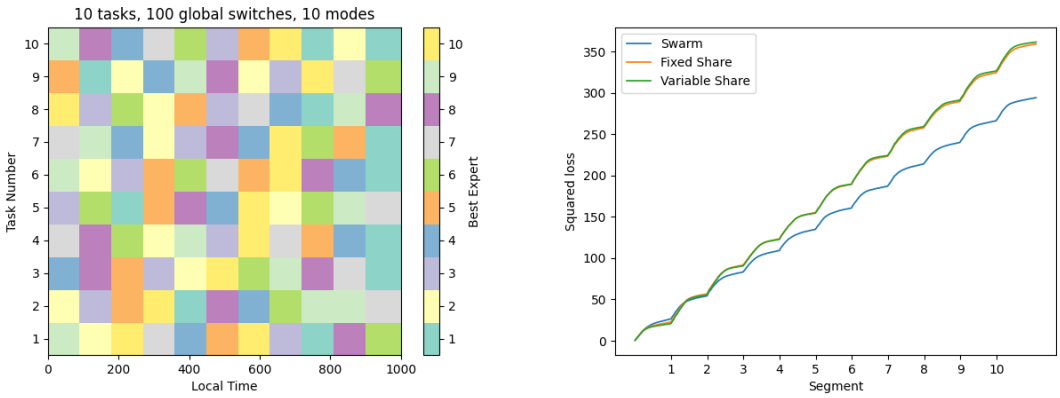
(a) Expert Assignment (b) Loss

Figure 3.12: Multi-Task Switching with $m = 2$

A highly switching environment, $m << k$, is tested next. For all values of m, the Swarm outperforms the Variable-Share learner. The benefits of inter-task communication are highlighted

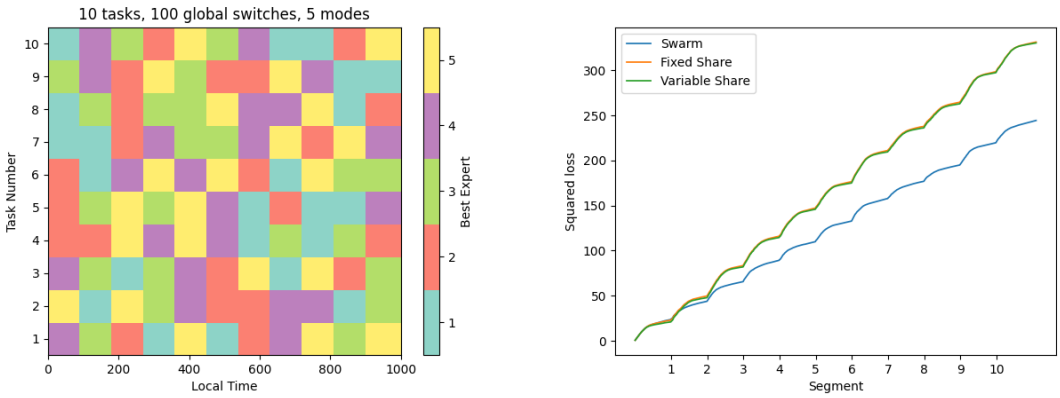
as m decreases, as more tasks contribute confidence toward the best expert set. This is seen by the loss reduction is the first segment. When the pool size is small, the Swarm learner incurs minimal loss at each segment by adapting rapidly to the switch. This is best demonstrated in Figure 3.15b, where at each switch, the Swarm converges to the best expert almost immediately.

v) Parameter sets: $(n = 100, T = 1000, s = 10, k = 90, m = [10, 5, 2])$.



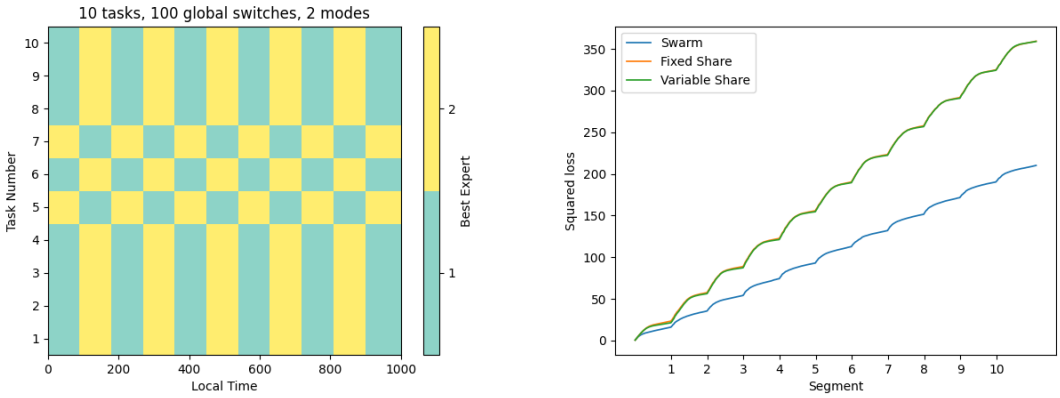
(a) Expert Assignment (b) Loss

Figure 3.13: Multi-Task Switching with $m = 10$



(a) Expert Assignment (b) Loss

Figure 3.14: Multi-Task Switching with $m = 5$



(a) Expert Assignment

(b) Loss

Figure 3.15: Multi-Task Switching with $m = 2$

3.3.5 Observations

In this section, experiments run on synthetic data have thoroughly demonstrated the consistent superior performance of the Swarm algorithm in comparison to the Variable-Share algorithm in a range of settings. In settings with no expert repetition, both algorithms perform similarly. In fact, the Swarm often falls just slightly behind, due to the choice of learning rate in these experiments. In contexts involving repeated experts, either within a single task or across tasks, the Swarm quickly outperforms the Variable-Share algorithm. The more the subset of experts is repeated, the greater the gap in performance. Using the patterns observed in these experiments, parallels are drawn in Chapter 4 between these experiments and those conducted on real-world data.

Chapter 4

Climate Data Experiments

This chapter details the application of the Swarm learner to the task of predicting temperature anomalies. The aim of these experiments is to test whether the strong performance of the Swarm, as witnessed in synthetic experiments, is replicated when applied to real-world data. The success and limitations of the algorithm in this setting are then discussed.

4.1 Motivation

Currently, the standard best predictor in this area of research is the mean of the GCM's predictions. This naive method does not factor in historical performance of the GCM. The experiments in the previous chapter demonstrated that the Swarm outpredicts the average expert, thus should outpredict the current best predictor in this domain also.

While various zones each have distinct climates, all climates are connected by global weather patterns. The application of a multi-task model can allow the best predictor for the current climate to be communicated across tasks. The multi-task model is applied to zonal temperature anomaly prediction via each task corresponding to a section of the globe (as defined by latitude and longitude). For a given zone, the Swarm predicts the variation from the monthly mean temperature using GCM expert advice and receives true temperature anomaly data as the outcome. In each experiment, the Swarm loss is compared to the loss of the each GCM, the average model prediction, and a Variable-Share learner. The Variable-Share learner is extended in the same manner as in Chapter 3, and is used to highlight the effect of global long-term memory in the Swarm.

In this chapter, the experiments mirror the style of the synthetic experiments. This is done to provide insight into the distribution of best performing GCMs and ultimately the best Swarm predictor in this setting. Using the findings from this set of experiments, the best model is implemented and tuned to provide the most accurate predictor.

Ideally, the aim is to generate an expert advice model successful enough for use in future anomaly prediction and forecasting. This would be proven via an improvement in performance compared to the best current predictor - the average GCM prediction. Regardless of performance, this application will provide insight into the underlying distribution of the best GCMs. Through varying experiment parameters (i.e. number of tasks and switches), the best method of modelling the GCM best expert is identified. The change in parameters signifies a differing assumption about the GCM performances. By identifying the best-performing algorithm, the most accurate assumptions about GCM performance can be inferred. Identifying which GCMs perform well, and in which locations, could aid in future understanding of why certain models perform better.

4.2 Experimental Design

The models applied in the experiments from Chapter 3 are applied to the temperature anomaly prediction setting as follows. The expert set comprises of 15 GCMs. There are 10 tasks with each task corresponding to a different region of the globe. To provide fair comparison against the Variable-Share learner, the η value is set to 2 for both algorithms. All other parameters are set to the optimal values, as in Chapter 3.

4.2.1 Data Preparation

A set of General Circulation Models taken from the World Climate Research Programme's (WCRP's) Coupled Model Intercomparison Project Phase 3 (CMIP3) Multi-Model Dataset [14]. The specific model details can be found in Appendix B. As each GCM outputs monthly temperature per region, data is first translated into regional monthly temperature anomalies. All available models contain data from the years 1980-2000. As such, the period 1980-1990 was provided as a benchmark region, in order to reduce bias caused by a small benchmark period. Zonal monthly averages are calculated by taking the mean temperature across the same month in every year of the benchmark period for each location. Zonal anomalies for the years 1990-2000 are simply calculated by subtracting the applicable zonal, monthly means from the 1990-2000 predictions. These results are standardised, with mean 0 and standard deviation 1. The temperature anomaly predictions at this stage have small regions, gridded by latitude and longitude with width 0.1. By not grouping results until the final stage data generation, experiments can be conducted over various region sizes.

The true outcomes are recorded temperature anomalies taken from the NASA GISTEMP Database[15]. NASA GISTEMP provides historical recorded temperature anomaly data on both a monthly and annual basis. The experiments are conducted on monthly predictions in order to provide sufficient data points for examination over the 10-year period. This data is already in the correct format and requires no preparation.

NASA grids the globe into ten sections, one point from each section was sampled such that regions of size $(\pm 10, \pm 10)$ around each point suffered no overlap. This equates to 3 points in Asia, 2 in Africa and North America, and 1 in each of Europe, South America and Oceania. Due to the small size of these regions, the predictions are be representative of a single subclimate. These 10 points are used throughout this set of experiments. This follows the experimental design of Monteleoni's experiments, with the alteration of smaller zone sizes.

4.3 Experiments

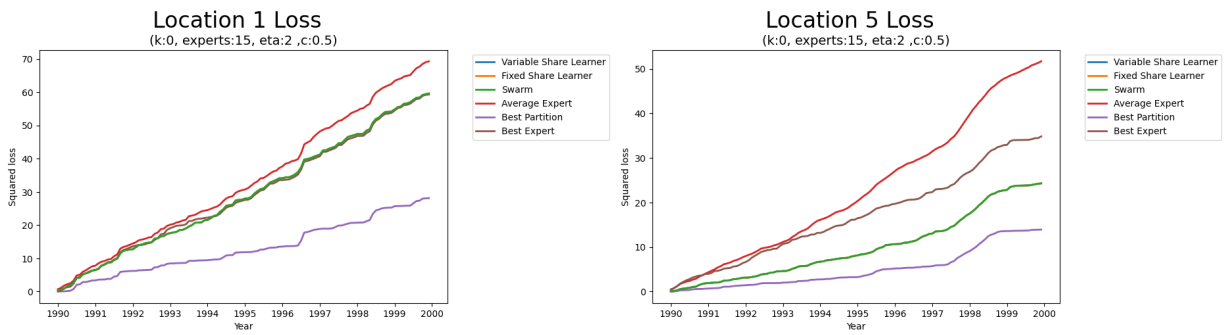
By first operating in the single-task case, experiments are run in static and switching settings. This will first confirm the suitability of the algorithms to this setting, in the simple static case. Any benefit of long-term memory as applied to temperature anomaly prediction will be seen via the difference in loss incurred by the Swarm and Variable-Share learner. The degree to which the Swarm outperforms the Variable-Share learner indicates the degree to which best experts are repeated. In the multi-task case, the relation between GCM performance across location will be demonstrated.

4.3.1 Single-Task Zonal Anomalies

First, the simulation of 1990-2000 is run for each location individually, emulating a single-task environment. Both the static and dynamic cases are tested by using switch values of $k = 0, 10$.

Static

For each location, both the Swarm and Variable-Share algorithms performed identically and incurred significantly less loss than the average expert. These algorithms consistently incurred no more loss than the best expert by the end of the trials, and in many cases far outperformed the best expert. This confirms that the best GCM weighting can be learned by the learners, and provides a baseline accuracy for the subsequent models. Figure 4.3 displays the best and worse performance locations of the learners. In location 1, the learners matched the performance of the best expert. In Figure 4.4, the fluctuation of weights in Location 5 indicates that the best-performing GCM does change meaning a switching algorithm would be more appropriate.



(a) Zone 1 (b) Zone 5

Figure 4.1: Zonal Losses in Single-Task Static Setting

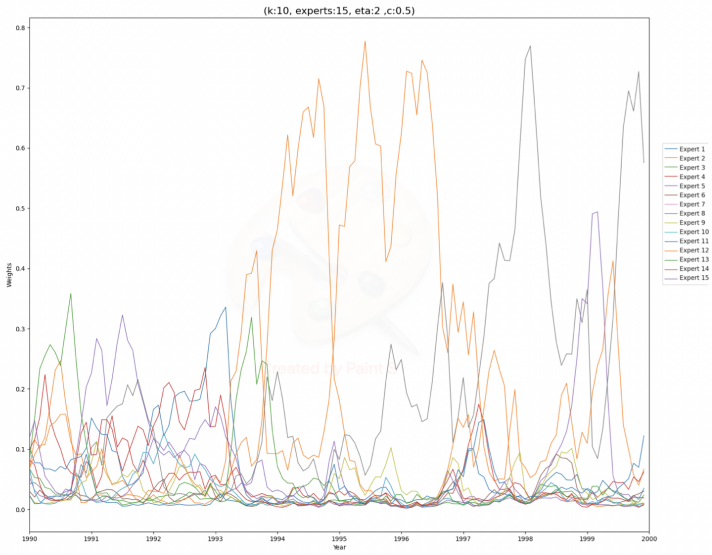
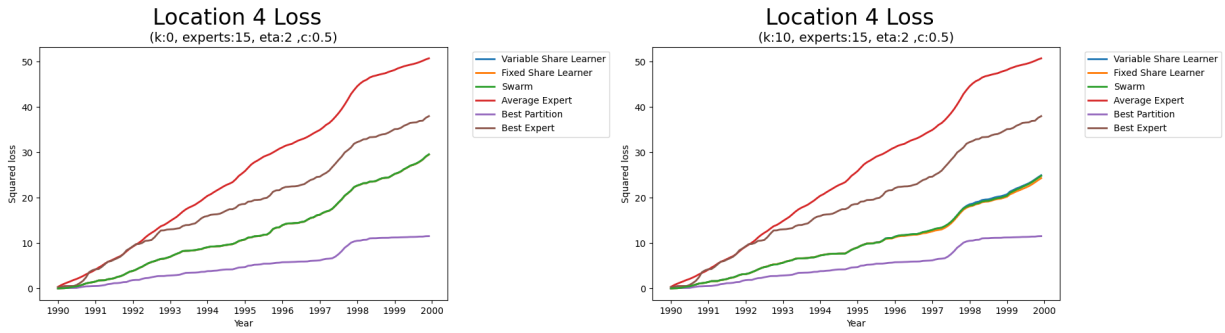


Figure 4.2: Zonal Weights in Single-Task Static Setting

Dynamic

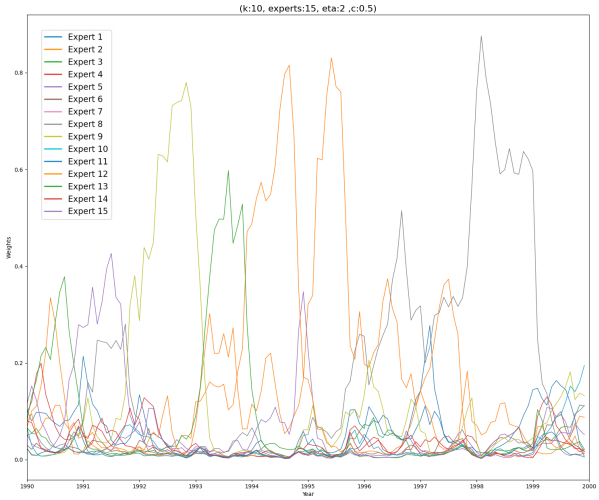
The learners are again implemented on each location separately, with 10 switches per location. Both learners suffered less loss for each location, due to the learners' ability to quickly adapt to GCMs that perform well. Figure 4.5 provides a side by side comparison of the loss at location 4. Location 4 demonstrated the largest decrease in learner loss from the static to dynamic setting. The decrease in loss confirms that a switching environment is suitable across all locations, and provides further improvment upon the current benchmark values (as shown by the red line).



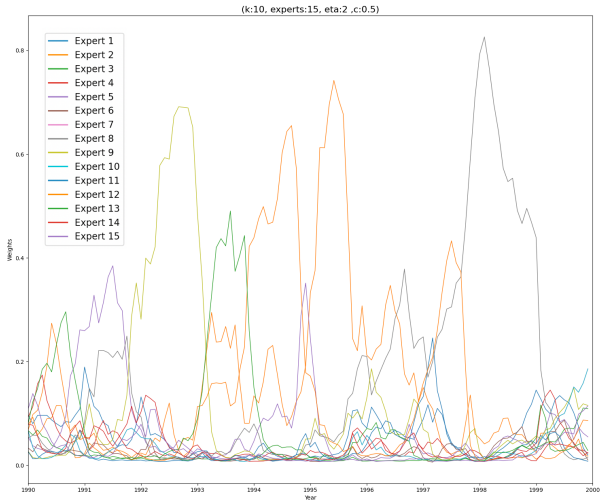
(a) Static (b) Dynamic

Figure 4.3: Location 4 Loss Comparison

The Swarm does not outperform the Variable-Share learner in any of these experiments. Figure 4.5b shows nearly identical performance between the two algorithms. This indicates the the long-term memory of the Swarm is not particularly beneficial in this setting. It is not the case that a GCM that predicts well over one period is no more likely to predict well again. In fact, in can be seen in the weights of the learners that certain experts do repeatedly perform well. However, in this case the ability of the Variable-Share learner to gain confidence in an expert quickly cancels out the benefit of long-term memory. The Variable-Share learner is not hampered by previous performance therefore incurs loss in segments where a new expert performs well for only a short period of time. In location 4, despite an expert performing well repeatedly (as seen by the orange line in Figure 4.6), the Swarm incurs loss identical to that of the Variable-Share algorithm.



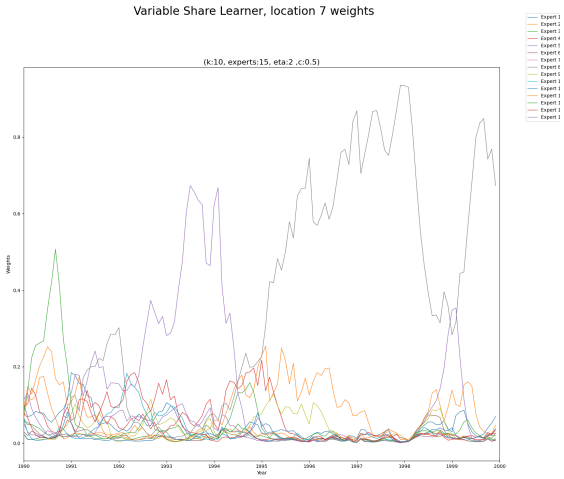
(a) Variable-Share Weights



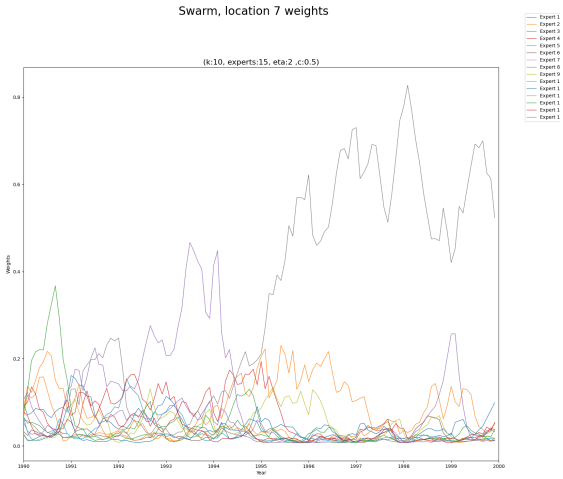
(b) Swarm Weights

Figure 4.4: Location 4 Weight Comparison

The Swarm does incur slightly less loss than the Variable-Share learner over the initial period in which an expert is repeated. There are specific locations which benefit from the Swarm's long-term memory. For example, in location 7, the Swarm's ability to retain weight in an expert that has performed well over a long period leads to the Swarm incurring less loss than the Swarm in this trial. By comparison, circa 1999, the Variable-Share learner gains confidence in a learner which only briefly performs well. This is seen in Figure 4.7.



(a) Variable-Share Weights



(b) Swarm Weights

Figure 4.5: Location 7 Weight Comparison

4.3.2 Multi-Task Zonal Anomalies

Static

In this experiment, all locations are run together. The values of s, k and m are set to 10, 0 and 15. First, the static case is used to identify any benefit of global memory in this setting. In all locations, the Variable-Share and Swarm learner performed identically. Figure 4.6 shows the

identical loss of the Swarm and Variable Share learners, alongside the best expert in location 1. The weights for both learners in this location are shown in Figure 4.7. While nearly identical, the weights for the Swarm learner are slightly lower (approximately 0.05, 5% at maximum confidence). This is due to the shared confidence in experts from the other tasks. Though there is some overlap in the best GCM across tasks, there is enough variation such that global memory does not provide a significant advantage.

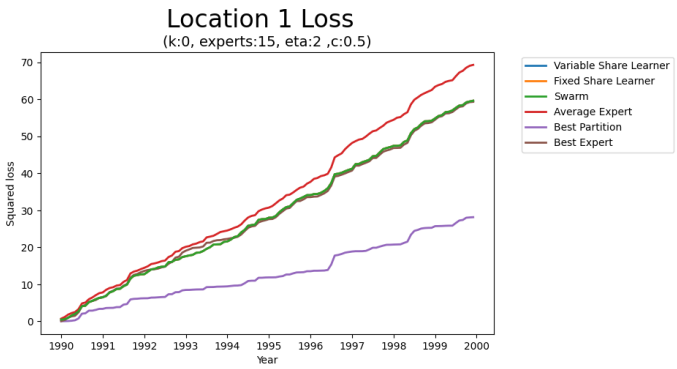
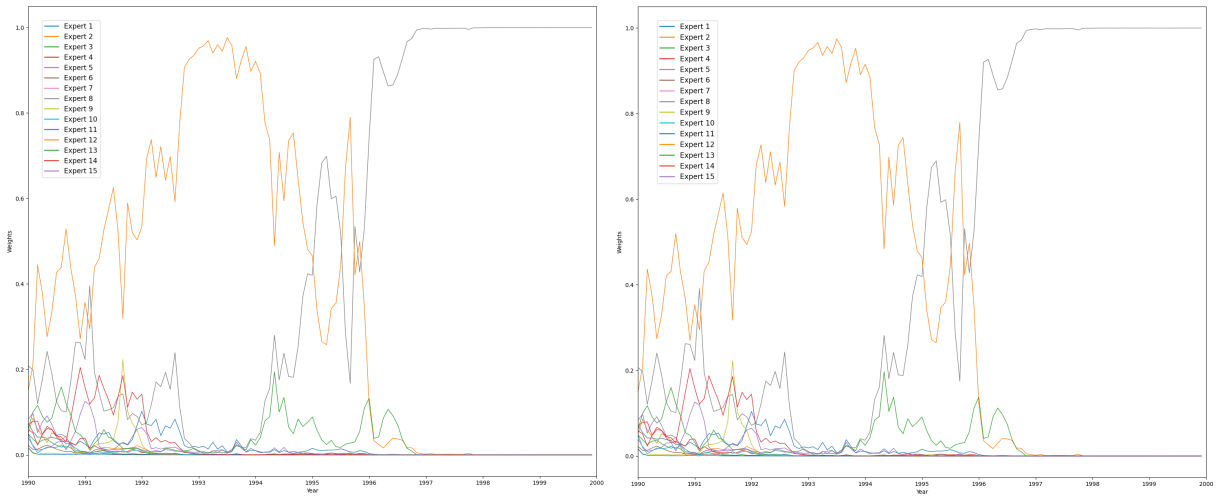


Figure 4.6: Location 1 Loss



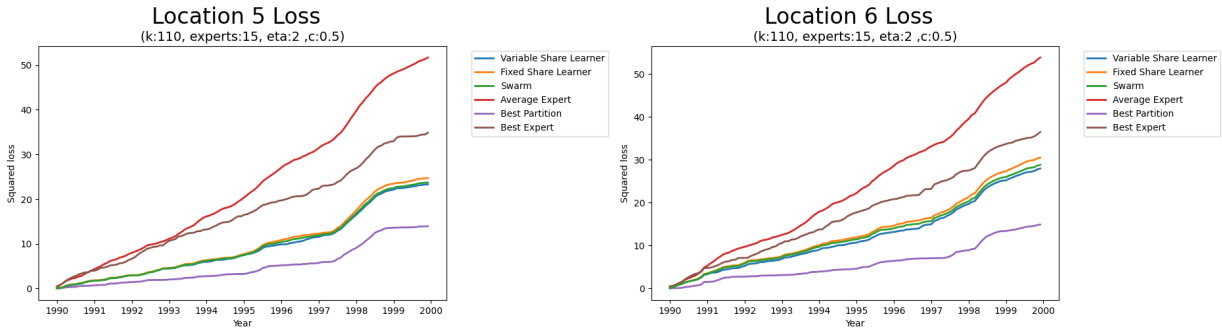
(a) Variable-Share Weights

(b) Swarm Weights

Figure 4.7: Location 1 Weight Comparison

Dynamic

The switch value is set to 100, indicating an approximate yearly change in best expert. In the majority of locations, the learners perform virutally identically as in the static case. However, in a handful of locations (such as Locations 5 and 6) the Swarm incurs higher loss than the Variable-Share learner (Figure 4.8).



(a) Location 5

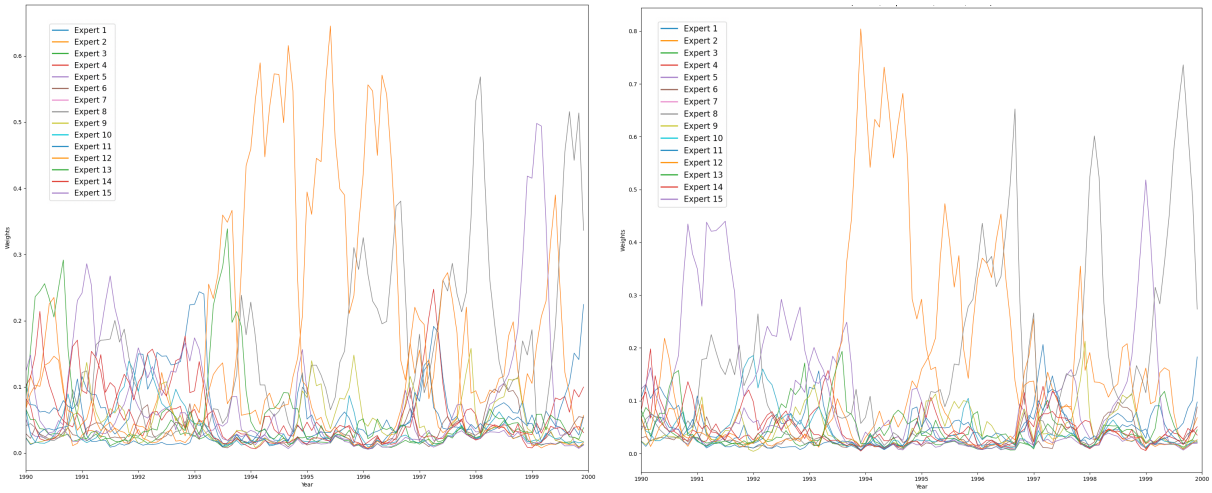
(b) Location 6

Figure 4.8: Location 5 and 6 Loss

Locations 5 and 6 share a common best expert (as shown by the orange line in Figure 4.9). As the expert performs well in both tasks, this should lead to a reduction in the loss incurred by the Swarm learner due to global memory. Inspection of the weights shows that the use of global memory stops the Swarm learner from gaining increased confidence in any one learner, as compared to the Variable-Share learner. The Variable-Share learner achieves a maximum confidence of 0.6-0.8 for the best expert (Figure 4.9). The maximum weight for any expert in the Swarm learner is less than 0.16 (Figure 4.10), far lower than the Variable-Share learner. This is due to the sharing of weights amongst tasks.

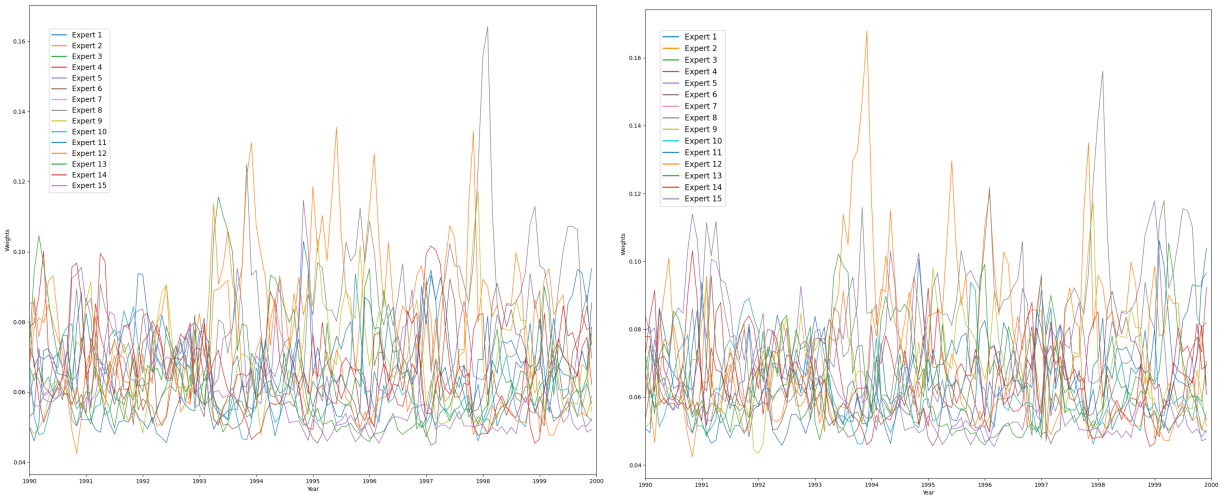
This increase in loss for the Swarm algorithm only occurs in locations which specific experts far outpredict the others. Most locations do not have a primary best predictor meaning that the increased confidence in multiple good experts allows the Swarm to still predict accurately. Of the locations used in these experiments, only 2 have primary best predictors. Even in these locations, the additional loss to the Swarm is small. Overall, the total loss across all regions is only slightly higher than the Variable-Share learner.

For instances where there is a significant best expert, the shared confidence leads to an increase in loss for the Swarm learner. In these locations, global memory is not necessarily beneficial. Locations 5 and 6 correspond to the most inland locations. These express potentially the most stable climate, due to the lack of effect of sea air. These climates experience the lowest temperature anomalies of all locations (<1C). The predictable behaviour of these locations lead to stability in the best predictor.



(a) Location 5 (b) Location 6

Figure 4.9: Variable-Share Weight Comparison



(a) Location 5 (b) Location 6

Figure 4.10: Swarm Weight Comparison

The same experiments were run with lower m values, to simulate a smaller expert set. These Swarm learners performed even worse in comparison to the Variable-Share learner, as the best expert subset was then decided upon globally. Confidence could therefore be gained in this chosen subset, which may not be applicable to all regions due to climate variations.

4.3.3 Final Model

The multi-task switching model with $m = 15$ is used as the final model. The model is first tuned, and then the simulations are re-run and results are discussed.

Parameter Tuning

The previous experiments demonstrated the improved accuracy in a switching setting, with an arbitrary setting of k. In contrast to the synthetic experiments, the best given expert at any given

time is not known in advance, nor is the number of switches. The switching parameter is tuned via a grid search over a range of potential values. Experiments were repeated using 10 different randomly sampled points as above, and the average loss for each k value was calculated. The results for $k = 10, 20, 50, 100, 1200$ are seen in Figure 4.2. These values correspond to $1, 2, 5, 10, 120$ local switches over the ten-year period. Varying the expected switch value in the range $k < 100$ caused little change to the loss incurred by the Swarm learner. Increasing the number of switches, led to a slight decrease in the loss of the Swarm for these values. Beyond this, higher values for the number of expected switches led to an increased loss, as seen by the purple line in Figure 4.2. A suitable value of 100 was chosen, representing one switch per zone per year. As shown by the red line in 4.2, a local switch value of 10 provides the lowest loss.

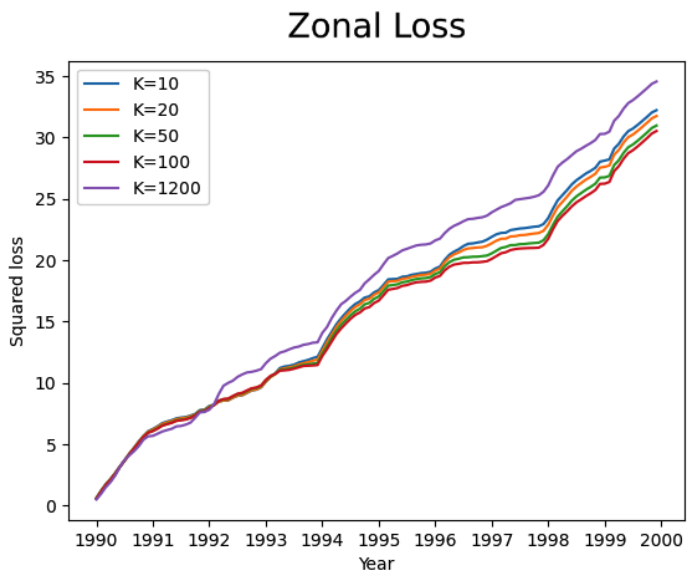


Figure 4.11: Average Zonal Loss over varying k values

To determine a suitable learning rate, a test was first run on a set of randomly sampled points over a range of η values. Another 10 locations ($\pm 10, \pm 10$ coordinates) were randomly sampled over the globe. The simulation was run in a dynamic, multi-task setting ($s = 10, k = 100$). Figure 4.1 displays the global loss over the test locations. The average zonal loss did not change drastically for all tested values of $\eta = [0.1, 0.2, ..., 10]$. As expected, higher values of η provided lower loss. Learners with η values of $5, 10$ performed similarly to the Variable Share learner. As seen in the previous experiments, a learning rate of 2 for the Swarm algorithm has higher global loss. As the environment fluctuates wildly and quick adaptation is required, an aggresive value of 10 is used for the learning rate in the final model to provide minimal loss.

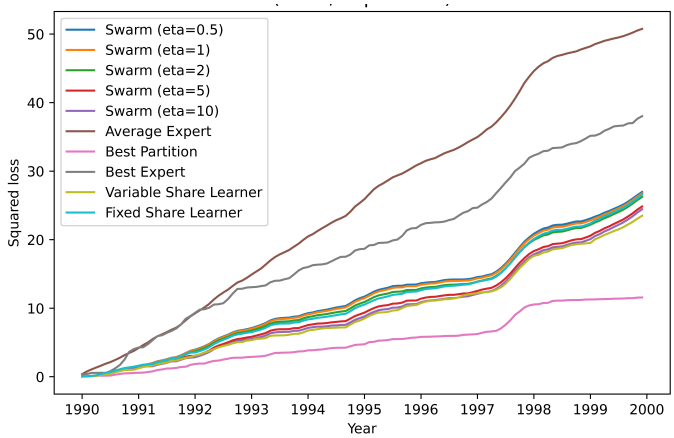


Figure 4.12: Average Zonal Loss over varying η values

4.4 Results

Using the tuned model, a final simulation is run against a new set of 10 locations with the same constraints as before. The total loss incurred over all regions by the Variable-Share and Swarm algorithms is significantly less than the existing benchmark (Figure 4.13). Both learners also incur less loss than the best expert for each location, when assessed globally.

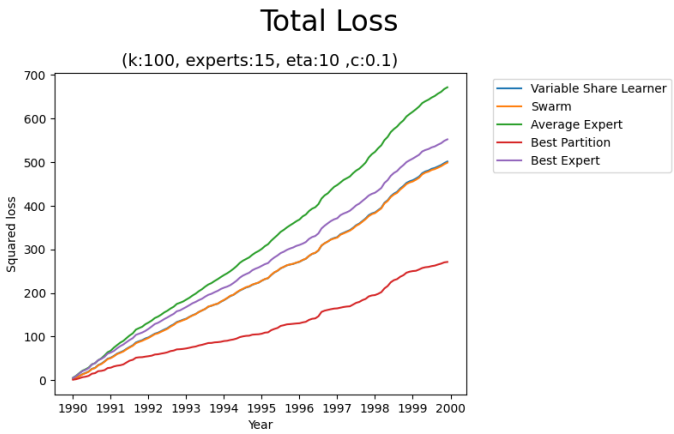
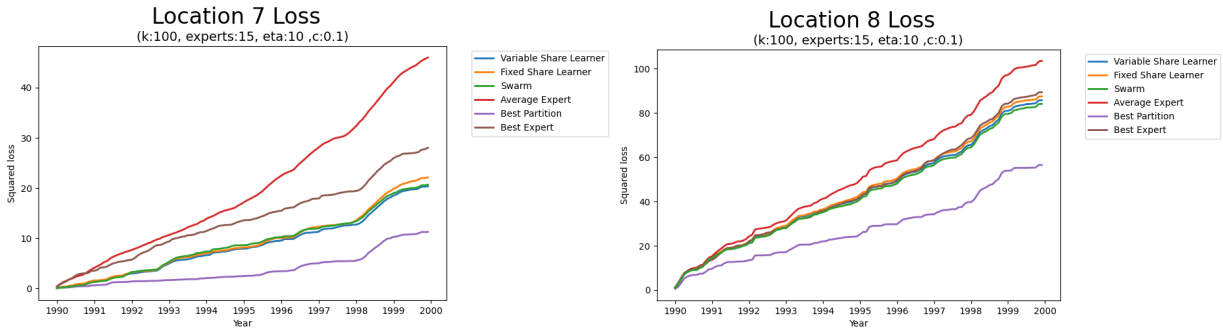


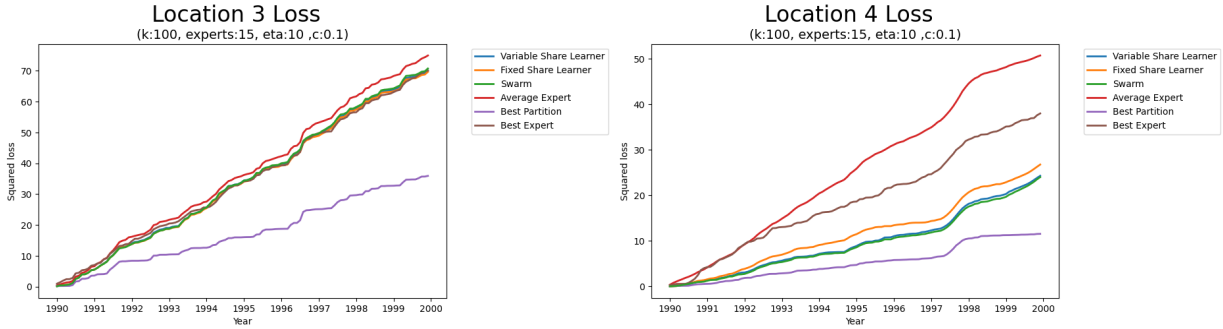
Figure 4.13: Global Loss in Final Simulation

In all but one location, the tuned Swarm algorithm outperforms the Variable-Share learner. However, differences between the learners are insignificant relative to the total loss. When loss is compiled over all locations, the loss of the learners is virtually identical. The locations with the highest and lowest difference between learners is shown in Figure 4.14. Between the best and worst Swarm:Variable-Share performances, the range of differences between learners is 5.3 which reaffirms the similarity of the learners. Figure 4.15 shows the locations in which the learners incurred the most and least loss relative to the best expert. The maximum and average difference between the Swarm and best learner is 15, this further proves the Swarm as an improvement upon the average predictor.



(a) Location 7 (b) Location 8

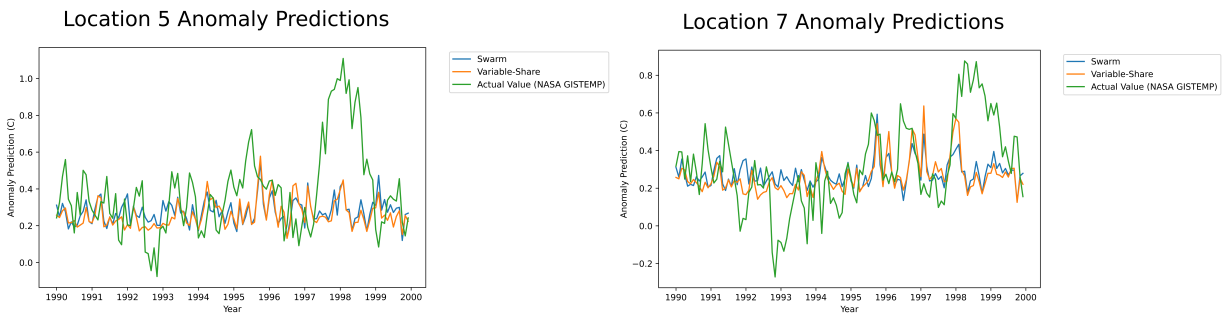
Figure 4.14: Location Loss Comparison



(a) Location 3 (b) Location 4

Figure 4.15: Location Loss Comparison

By examining the predictions of the model against the true ground anomalies, it can be seen that the algorithms' predictions are far more conservative. The Swarm predictions seem to mimic the yearly average zonal anomaly with small annual increases at the beginning of each year. The locations in which the learners incurred the least loss are shown in Figure 4.16, compared to the observed temperature anomalies (represented by the green line). In both locations, the Swarm leaner slightly adapts to a greater anomaly during 1995. However, in the years of 1993, 1996 and 1998 the greater anomalies were not successfully adapted to. The Swarm utilises the GCM set as input so its predictions are bounded by the input and none of the GCMs came close to predicting the 1998-1999 temperature increase.



(a) Location 5 (b) Location 7

Figure 4.16: Temperature Anomaly Prediction Comparison

Herein lies the significant drawback of this prediction method. All the GCMs seem to underestimate temperature anomalies. This is best demonstrated in Figure 4.17. The cluster of expert predictions lie around the mean prediction of +0.1 and remain in the range ±1, yet the true anomalies reach maximums of 2.95. A main use of temperature anomaly predictions is to forecast the effect of climate change. As the Swarm learner generates its prediction based upon GCMs advice, if these predictions severely underpredict the extent of temperature anomalies, so too will the predictions of the Swarm learner.

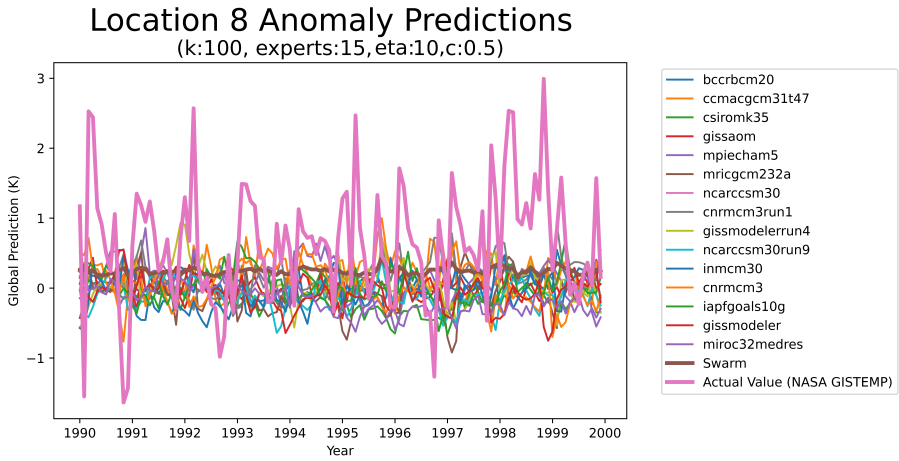
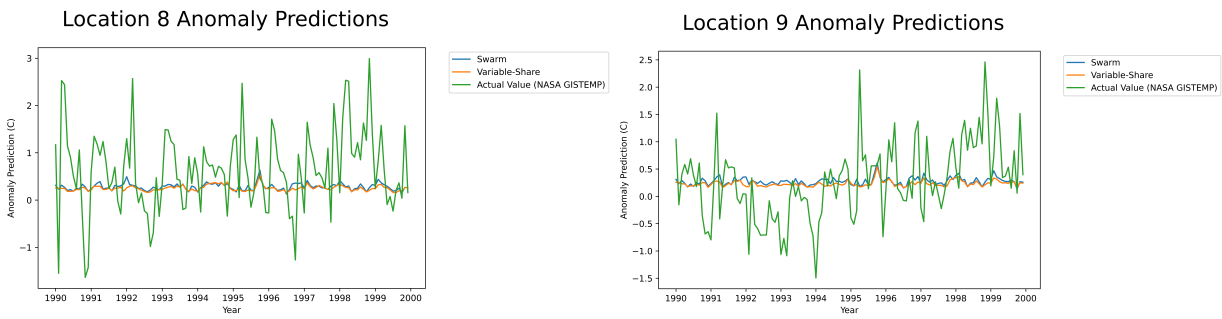


Figure 4.17: Location 8 Temperature Anomaly Predictions

In locations 8 and 9, the learners receive the highest loss. This is due to overly conservative expert predictions, and high variance in oberserved anomalies (Figure 4.18).



(a) Location 8

(b) Location 9

Figure 4.18: Temperature Anomaly Prediction Comparison

Summary

The final model significantly outperformed the existing benchmark in each location. In the majority of locations, the best individual GCM is also improved upon. As in the first set of climate experiments, the Swarm incurred loss comparable to that of the Variable-Share learner. The tuned model slightly improves upon the Variable-Share learner overall, and predicts more accurately than the Variable-Share in the majority of locations. Both learners provide more accurate predictions than current methods, with the best predictor being the tuned model.

Long-term memory is proven to be useful, as the Swarm model incurs less loss than the Variable-Share in the single-task, switching environment. Global memory however seems to lead to slightly

poorer performance in comparison to the single-task learners, as locations remain influenced by one another and the best expert may vary across locations. The multi-task switching model still provides improvements upon the existing methods. Via drawing comparisons to the experiments in the previous chapter, the multi-task learner should perform far better than the single-task learners provided there is a common best expert set. The lack of improvement from the single- to multi-task learner signals that the best experts between locations are not as common.

The predictions of the learners are limited by the expert advice. When the anomalies are far outside the expert prediction range, all models and learners incur more loss. The aim of climate change prediction is therefore made difficult, as more accurate GCMs would first be required in order to build a more accurate learner.

Chapter 5

Discussion

5.1 Conclusion

In this paper, a new online multi-task learner utilising expert advice was applied to a series of experiments using both synthetic and real-world data and proved that this novel algorithm outperforms existing techniques.

First, systematic testing of the Swarm on synthetic data highlighted the benefits of global, long-term memory via demonstrating that the new algorithm incurs less loss than the existing algorithms (i.e. Fixed-Share, Variable-Share.).

The new task of temperature anomaly prediction in the multi-task setting was introduced. The set of experiments was then repeated on real-world climate data to gain insight into the performance of the advising GCMs. Using this information, a final mutli-task model was implemented alongside existing methods for comparison. Performance of the Swarm and Variable-Share algorithms in this multi-task environment provide promising results by generating better predictions than the average expert which is the current best predictor. Furthermore, in the majority of experiments, these algorithms also outperform every GCM.

5.2 Evaluation

Through providing empirical evidence of the algorithms' performances in a synthetic setting, a baseline was provided by which to compare the performance on real-world data. These synthetic experiments were completed systematically and each experiment successfully demonstrated the necessary facet of memory.

While the results on real-world data are an improvement upon existing methods, work could be done to obtain a more accurate predictor. Currently, generated predictions tend to be overly conservative. As the predictions of these learners are affected by the performance of the GCMs, a main issue arises from the GCMs inability to predict anomalous temperature anomalies. The performance of the Swarm is comparable to that of the Variable-Share learner which does not utilise global memory. It is observed that certain GCMs perform well across regions, however the additional confidence does not necessarily lead to improved accuracy in such models as both GCM performance and ground truths vary wildly. These algorithms are more applicable in environments with a higher degree of regularity.

5.3 Future Work

This paper provided the foundation for future experiments applying the Swarm to climate data, and many variations of the conducted experiments would be suitable future research. For example, updated GCMs have started to be published in the newer CMIP6 dataset. These models could be the basis for ample further experiments, first by reproducing the results of this experiment set and then by conducting experiments over longer time periods. With sufficient computing power, running these experiments over larger sets of locations would potentially provide more accurate models globally, or by clustering locations together for regional accuracy Additional experiments could also be conducted in the forecast setting, by using a single model forecast predictions as the target values.

The use of more complex techniques, such as smoothing using neighbourhood-influence [12] could also applied to either dataset, to observe the combined effect of the Swarm and enhanced climate predictors. By utilising the neighbourhood smoothing algorithm, the set of expert advice to the Swarm would be enhanced which in turn would improve the Swarm predictions. In addition, the Swarm could be used to identify the optimal regional weights for the neighbourhood-smoothing algorithm, dependent on climate. As the GCMs are chaotic, meaning small changes propagate to larger changes throughout the model, the Swarm could identify these regions of large impact.

In the style of the Learn-α algorithm, a meta-algorithm could be applied to multiple instantiations of the Swarm algorithm to learn its optimal parameters. Alternatively, the Swarm itself could be used as a meta-learner in an adaptation of Learn-α. Predictions generated by the Learn-α could also be directly compared to the Swarm.

The Swarm learner could be extended to the infinite hypothesis space via use of kernels, as a second algorithm is provided in [1]. Preliminary work on this learner was done as an extension to this project, and its implementation for the single-task is detailed in Appendix D. Future work could extend this learner to the multi-task setting.

Bibliography

[1] M. Herbster, S. Pasteris, and L. Tse, "Online Multitask Learning with Long-Term Memory," *arXiv:2008.07055*, Aug. 2020.

[2] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," 1994.

[3] V. G. Vovk, "Aggregating strategies," pp. 371–386, July 1990.

[4] G. Vovk, "A game of prediction with expert advice," p. 10, 1998.

[5] M. Herbster and M. K. Warmuth, "Tracking the Best Expert," p. 28, 1998.

[6] O. Bousquet and M. K. Warmuth, "Tracking a Small Set of Experts by Mixing Past Posteriors," *Journal of Machine Learning Research 3 (2002), pp. 363–396.*, p. 34, 2002.

[7] V. M. Krasnopolsky and M. S. Fox-Rabinovitz, "Complex hybrid models combining deterministic and machine learning components for numerical climate modeling and weather prediction," *Neural Networks*, vol. 19, pp. 122–134, Mar. 2006.

[8] E. Hawkins and R. Sutton, "The Potential to Narrow Uncertainty in Regional Climate Predictions," *Bull. Amer. Meteor. Soc.*, vol. 90, pp. 1095–1108, Aug. 2009.

[9] P. A. O'Gorman and J. G. Dwyer, "Using Machine Learning to Parameterize Moist Convection: Potential for Modeling of Climate, Climate Change, and Extreme Events," *Journal of Advances in Modeling Earth Systems*, vol. 10, no. 10, pp. 2548–2563, 2018. _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2018MS001351.

[10] S. J. Doherty, S. Bojinski, A. Henderson-Sellers, K. Noone, D. Goodrich, N. L. Bindoff, J. A. Church, K. A. Hibbard, T. R. Karl, L. Kajfez-Bogataj, A. H. Lynch, D. E. Parker, I. C. Prentice, V. Ramaswamy, R. W. Saunders, M. S. Smith, K. Steffen, T. F. Stocker, P. W. Thorne, K. E. Trenberth, M. M. Verstraete, and F. W. Zwiers, "Lessons Learned from IPCC AR4: Scientific Developments Needed to Understand, Predict, and Respond to Climate Change," *Bull. Amer. Meteor. Soc.*, vol. 90, pp. 497–514, Apr. 2009.

[11] T. Reichler and J. Kim, "How Well Do Coupled Models Simulate Today's Climate?," *Bull. Amer. Meteor. Soc.*, vol. 89, pp. 303–312, Mar. 2008.

[12] C. Monteleoni, G. A. Schmidt, S. Saroha, and E. Asplund, "Tracking climate models," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 4, no. 4, pp. 372–392, 2011. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sam.10126.

[13] S. McQuade and C. Monteleoni, "Global Climate Model Tracking using Geospatial Neighborhoods," p. 7, 2012.

[14] G. A. Meehl, C. Covey, T. Delworth, M. Latif, B. McAvaney, J. F. B. Mitchell, R. J. Stouffer, and K. E. Taylor, "The wcrp cmip3 multi-model dataset: A new era in climate change research," 2007.

[15] G. Team, "Giss surface temperature analysis (gistemp), version 4," 2021. NASA Goddard Institute for Space Studies. Dataset accessed 2020-11-02 at data.giss.nasa.gov/gistemp/.

[16] F. Rosenblatt, *The Perceptron: A Probabilistic Model for Information Storage and Organization.*

[17] Y. Freund and R. E. Schapire, "Adaptive Game Playing Using Multiplicative Weights," *Games and Economic Behavior*, vol. 29, pp. 79–103, Oct. 1999.

[18] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, pp. 119–139, Aug. 1997.

[19] A. Chernov and V. Vovk, "Prediction with expert evaluators' advice," *arXiv:0902.4127 [cs]*, Mar. 2009. arXiv: 0902.4127.

[20] D. Adamskiy, M. K. K. Warmuth, and W. M. Koolen, "Putting Bayes to sleep," p. 9.

[21] S. Arora, E. Hazan, and S. Kale, "The Multiplicative Weights Update Method: a Meta Algorithm and Applications," p. 31.

[22] T. Reichler and J. Kim, "How Well Do Coupled Models Simulate Today's Climate?," *Bull. Amer. Meteor. Soc.*, vol. 89, pp. 303–312, Mar. 2008.

[23] C. Monteleoni and T. S. Jaakkola, "Online Learning of Non-stationary Sequences," p. 8.

[24] V. G. Vovk, "A game of prediction with expert advice," in *Proceedings of the eighth annual conference on Computational learning theory*, COLT '95, (New York, NY, USA), pp. 51–60, Association for Computing Machinery, July 1995.

[25] D. Haussler, J. Kivinen, and M. K. Warmuth, "Sequential prediction of individual sequences under general loss functions," *IEEE Transactions on Information Theory*, vol. 44, pp. 1906–1925, Sept. 1998. Conference Name: IEEE Transactions on Information Theory.

[26] M. Hosseini, A. Bigtashi, and B. Lee, "Generating future weather files under climate change scenarios to support building energy simulation – A machine learning approach," *Energy and Buildings*, vol. 230, p. 110543, Jan. 2021.

[27] J. P. Evans, F. Ji, C. Lee, P. Smith, D. Argüeso, and L. Fita, "Design of a regional climate modelling projection ensemble experiment – NARCliM," *Geoscientific Model Development*, vol. 7, pp. 621–629, Apr. 2014. Publisher: Copernicus GmbH.

[28] S. Shackley, P. Young, S. Parkinson, and B. Wynne, "Uncertainty, Complexity and Concepts of Good Science in Climate Change Modelling: Are GCMs the Best Tools?," *Climatic Change*, vol. 38, pp. 159–205, Feb. 1998.

[29] B. Wang, L. Zheng, D. L. Liu, F. Ji, A. Clark, and Q. Yu, "Using multi-model ensembles of CMIP5 global climate models to reproduce observed monthly

rainfall and temperature with machine learning methods in Australia," *International Journal of Climatology*, vol. 38, no. 13, pp. 4891–4902, 2018. _eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/joc.5705.

[30] K. Ahmed, D. A. Sachindra, S. Shahid, Z. Iqbal, N. Nawaz, and N. Khan, "Multi-model ensemble predictions of precipitation and temperature using machine learning algorithms," *Atmospheric Research*, vol. 236, p. 104806, May 2020.

[31] M. Herbster, "Learning additive models online with fast evaluating kernels," p. 17.

[32] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online Learning with Kernels," *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, vol. 52, no. 8, p. 12, 2004.

[33] T. M. Press, "Learning with Kernels | The MIT Press." Publisher: The MIT Press.

[34] M. Herbster, "Learning Additive Models Online with Fast Evaluating Kernels," in *Computational Learning Theory* (G. Goos, J. Hartmanis, J. van Leeuwen, D. Helmbold, and B. Williamson, eds.), vol. 2111, pp. 444–460, Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. Series Title: Lecture Notes in Computer Science.

[35] C. Gentile, "A New Approximate Maximal Margin Classification Algorithm," p. 30.

[36] P. Auer and M. Warmuth, "Tracking the best disjunction," in *Proceedings of IEEE 36th Annual Foundations of Computer Science*, (Milwaukee, WI, USA), pp. 312–321, IEEE Comput. Soc. Press, 1995.

[37] N. Littlestone, "Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm," p. 34.

[38] N. J. L. Lenssen, G. A. Schmidt, J. E. Hansen, M. J. Menne, A. Persin, R. Ruedy, and D. Zyss, "Improvements in the GISTEMP Uncertainty Model," *J. Geophys. Res. Atmos.*, vol. 124, pp. 6307–6326, June 2019.

Appendix A

Project Plan

A.1 Aims

Explore the performance of a novel online multi-task learning algorithm by providing experimental evidence, including tests on real-world climate data. Examine the performance of the algorithm and assess its suitability for the climate prediction setting.

A.2 Objectives

1. Review current online learning algorithms, both single- and multi-task.
2. Implement online learning algorithms for comparison.
3. Implement experiments on synthetic to provide evidence of a functional implementation of the multi-task algorithm. Compare the performance to that of other techniques.
4. Examine current literature regarding online learning as applied to climate data.
5. Gather and prepare climate data for input into the online multi-task algorithm.
6. Tune the learner to improve performance on test data and run experiments to compare the algorithm's performance to existing methods.
7. Evaluate the application of the algorithm in this setting.

Appendix B

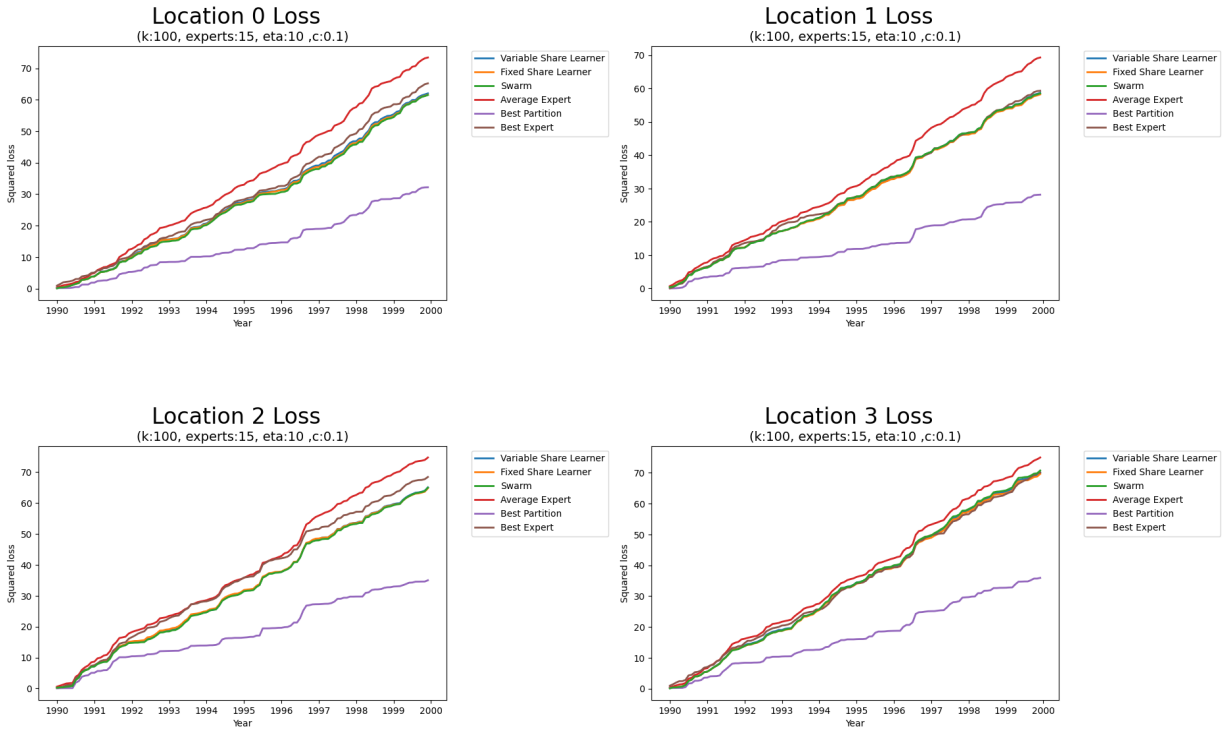
Table of Used GCMs

Model Code	Institute	Model
bccrbcm20	Bjerknes Centre for Climate Research	BCM2.0
cccmacgcm31t47	Canadian Centre for Climate Modelling and Analysis	CGCM3.1 Model T47
cnrmcm3	Meteo-France, Centre National de Recherches Meteorologiques	CM3 Model
cnrmcm3run1	Meteo-France, Centre National de Recherches Meteorologiques	CM3 Model
csiromk35	CSIRO Atmospheric Research, Australia	Mk3.5 Model
gissaom	NASA Goddard Institute for Space Studies	C4x3
gissmodelerrun4	NASA Goddard Institute for Space Studies	C4x3
gissmodeler	NASA Goddard Institute for Space Studies	C4x3
iapfgoals10g	LASG, Institute of Atmospheric Physics, China	FGOALS1.0_g
inmcm30	Institute for Numerical Mathematics, Russia	INMCM3.0 Mode
miroc32medres	CCSR/NIES/FRCGC	MIROC Model V3.2 High Resolution
mpiecham5	Max Planck Institute for Meteorology, Germany	ECHAM5 MPI OM
mricgcm232a	Meteorological Research Institute, Japan	CGCM2.3.2a
ncarccsm30	Community Climate System Model	CCSM3.0
ncarccsm30run9	Community Climate System Model	CCSM3.0

All models contributed to the 'Climate of the 20th Century Experiment' (20C3M) for IPCC and can be accessed at https://esgf-node.llnl.gov/search/cmip3/.

Appendix C

Final Climate Experiments Results



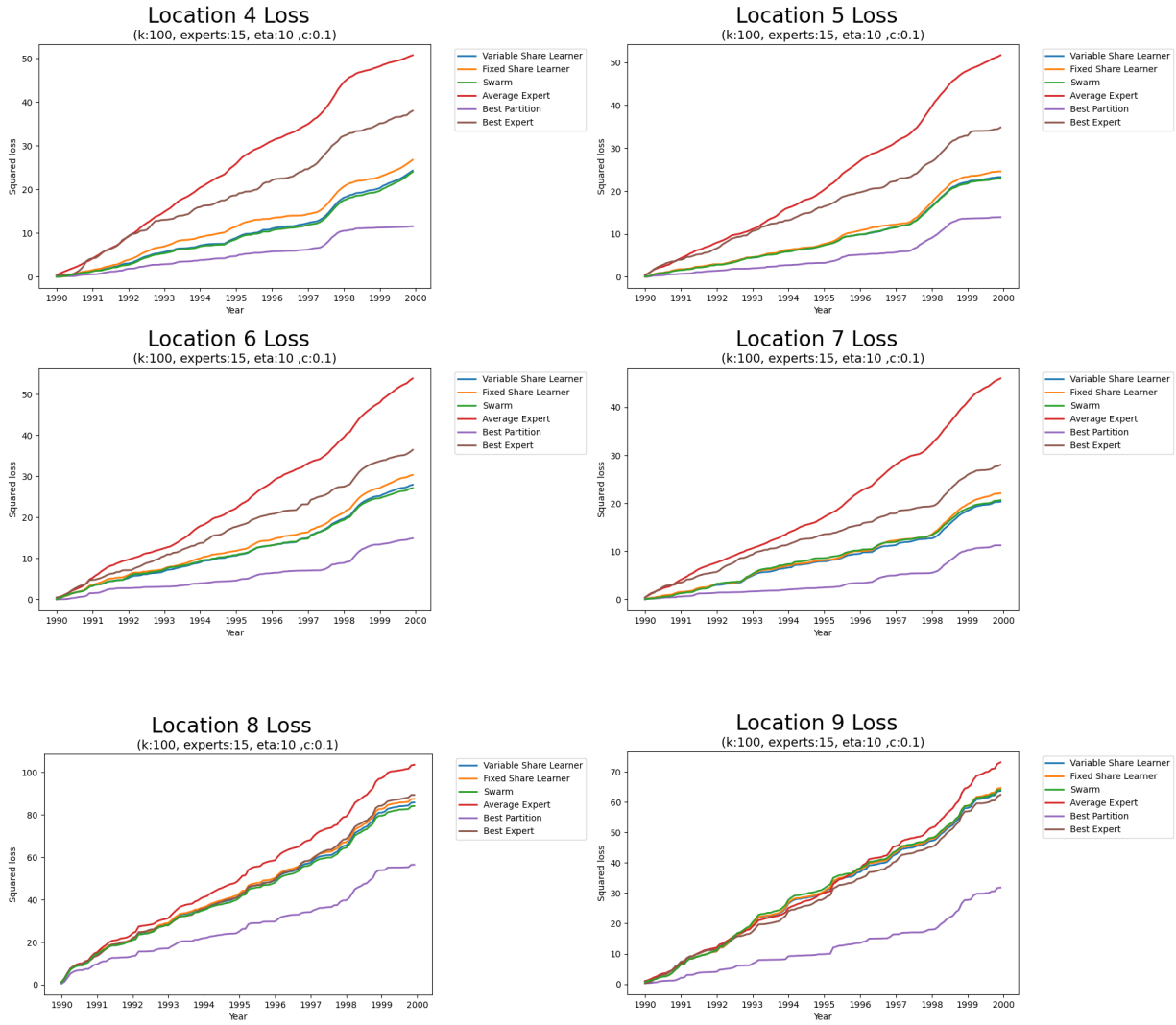


Figure C.4: Zonal Losses

Appendix D

Infiite Hypothesis Space

A set of best experts is a finite hypothesis space. The previous multi-task learner can be expanded to find the best functional approximation from an infinite hypothesis space via the use of a Reproducing Kernel Hilbert Space. Expert advice is clearly no longer required.

DEFINITION 4 *For $\mathcal{X} \subset \mathbb{R}^n$, a Reproducing Kernel Hilbert Space (RKHS) is a Hilbert space , $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathbb{R}$, if $\exists l : \mathcal{X} \to \mathbb{R}$ such that:*

1. *k has the reproducing property, $f(X) = \langle f(\cdot), k(\cdot, x)\rangle$*
2. *$\mathcal{H}$ is the closure of the span of $k(x, \cdot)$, for $\forall x \in \mathcal{X}$.*

An algorithm for predicting in a RKHS space in a multi-task setting is contributed by Herbster et al. [1].

Algorithm 9: Multi-task Online Learner for Predicting $\mathcal{H}_K^x$

Parameters: Tasks $s \in \mathbb{N}$, task lengths $T^1, ..., T^s \in \mathbb{N}$, $T = \sum_{i=1}^{s} T^i$, learning rate $\eta > 0$, complexity estimate $\hat{C} > 0$, modes: $m \in [T]$, strictly positive definite Kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $\tilde{P} = \tilde{P}^{\ell, T^1, ..., T^s} : [T] \times [T] \to \mathbb{R}$, with $\max_{\tau \in [T]} K(x^\tau, x^\tau) \leq \hat{X}_K^2$, $\hat{X}_P^2 = 2 \log_2 T$

Initialise: $\mathbb{U} \leftarrow \emptyset, \mathcal{X}^1 \leftarrow \emptyset, \mathcal{T}^1 \leftarrow \emptyset$

for $\tau = 1, ..., T$ **do**

- Receive task, $i \leftarrow \ell^\tau \in [s]$
- Receive $x^\tau \in \mathcal{X}$
- $t \leftarrow \boldsymbol{\sigma}(\tau)$ **Define:**
  - $\mathbf{K}^\tau := (K(x, z))_{x,z \in \mathcal{X} \cup \{x^\tau\}}$
  - $\mathbf{P}^\tau := (\tilde{P}(\tau, v))_{\tau, v \in \tau^\tau \cup \{\tau\}}$
  - $\tilde{\mathbf{X}}^\tau(v) := \left[\frac{\sqrt{\mathbf{K}^\tau}\mathbf{e}^{x^v}}{\sqrt{2\hat{X}_K^2}}; \frac{\sqrt{\mathbf{P}^\tau}\mathbf{e}^v}{\sqrt{2\hat{X}_P^2}}\right] \left[\frac{\sqrt{\mathbf{K}^\tau}\mathbf{e}^{x^v}}{\sqrt{2\hat{X}_K^2}}; \frac{\sqrt{\mathbf{P}^\tau}\mathbf{e}^v}{\sqrt{2\hat{X}_P^2}}\right]^\top$
  - $\tilde{\mathbf{W}}^\tau \leftarrow \exp\left(\log(\frac{\hat{C}}{2Tm})\mathbf{I}^{|\mathcal{X}^\tau|+|\mathcal{T}^\tau|+2} + \sum_{v \in \mathbb{U}} \eta y_v \tilde{\mathbf{X}}^\tau(v)\right)$
- **Predict**:
  - $Y^\tau \sim \mathrm{U}(-\gamma, \gamma)$; $\bar{y}^\tau \leftarrow \mathrm{tr}(\tilde{\mathbf{W}}^\tau(v)\tilde{\mathbf{X}}^\tau(v)) - 1$; $\hat{y}_t^i := \mathrm{sign}(\bar{y}^\tau - Y^\tau)$
- **Receive** $y_t^i := y^\tau \in \{-1, 1\}$ **if** $y^\tau \bar{y}^\tau \leq \frac{1}{\sqrt{m}}$ **then**
  - $\mathbb{U} \leftarrow \mathbb{U} \cup \{t\}, \mathcal{X}^{\tau+1} \leftarrow \mathcal{X}^\tau \cup \{x^\tau\}$ and $\mathcal{T}^{\tau+1} \leftarrow \mathcal{T}^\tau \cup \{\tau\}$
- **else**
  - $\mathcal{X}^{\tau+1} \leftarrow \mathcal{X}^\tau \cup \{x^\tau\}$ and $\mathcal{T}^{\tau+1} \leftarrow \mathcal{T}^\tau \cup \{\tau\}$

D.1 Experimental Design

Using a 1x1 grid, points are chosen within the grid and designated a positive or negative classification. The learner must now learn the boundary within this grid that divides the positive from negative points.

D.1.1 Synthetic Data Generation

Data was generated such that the decision boundaries were sufficiently complex to learn. As the goal is now classification, the output space is $\mathcal{Y} = \{-1, +1\}$. A two-dimensional input space was used, $\mathcal{X} = [0, 1]^2$. Given a radius, r, 6 adequately-spaced points, $\mathbf{v}_i$, were sampled (i.e. $|\mathbf{v}_{i,0} - \mathbf{v}_{j,0}| > 6r, |\mathbf{v}_{i,1} - \mathbf{v}_{j,1}| > 6r, \forall i, j \in [0, ..., 6]$) from $\mathcal{X}$. The points are divided into two groups; the first three points representing a positive classification, and the rest a negative classification. For t in $\{1, ..., T\}$, i is sampled from a discrete uniform distribution, $\{0, ..., 5\}$. The point for time t is then generated as below:

$$x_t \sim N(\mathbf{v}_i, r\mathbf{I}_n) \qquad y_t = \begin{cases} 1 & \text{if } i < 3 \\ -1 & \text{otherwise} \end{cases}$$

Around each point, v_i, 100 points are sampled from a multivariate Gaussian distribution with mean, $\mu = v_i$, and standard deviation $\mathbf{\Sigma} = r\mathbf{I}_n$. As the points are distanced from one another, the data is ensured to be linearly separable.

In the switching case, $k + 1$ sets of 6 points are sampled to define the decision boundary. For each segment, the points are generated as before.

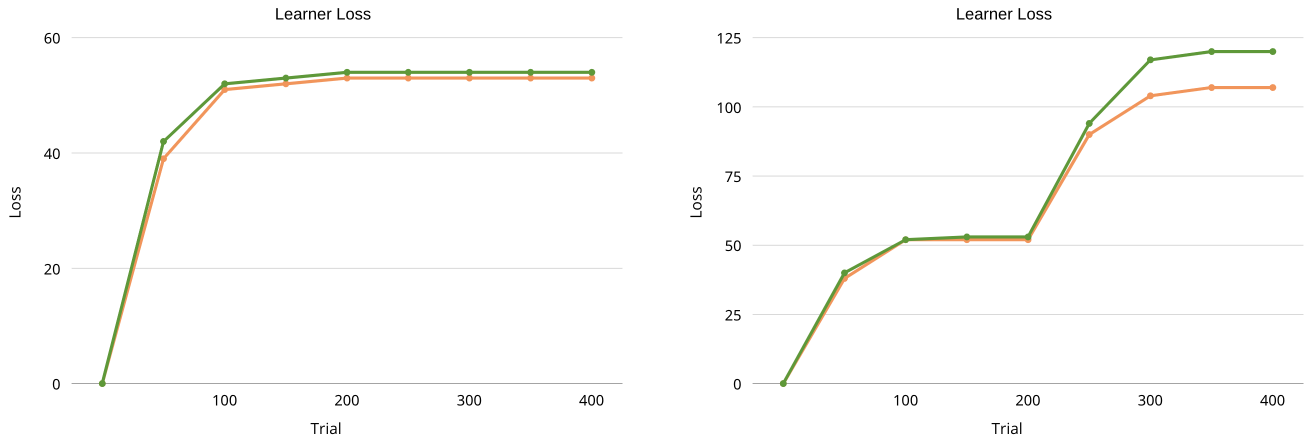
D.1.2 Implementation

A Gaussian kernel was trained using Online Gradient Descent, for comparison against the Swarm. Preliminary experiments were then run to determine the width of the Gaussian kernel. The accuracy per step is assessed via an unseen dataset of size 100, sampled from the same distribution as (x_t, y_t), every 50 trials. Due to the randomised test dataset, the learning curve is not smooth. Testing every 50 steps provides the general learning curve and reduces computation time. After a grid search over potential values, the width of the Gaussian was set to 0.1 and the degree of the polynomial k was set to 14. Figure 5.1 shows a sample of k values and their respective accuracies.

As implemented for a single task, experiments are run over 400 trials. First, the static case is represented by one set of points being chosen within the grid. The switching case is then tested by the sampling X values from a different set of points, after the first 200 trials.

D.1.3 Results

Both learners successfully learn the best boundary in the single-task case. The loss incurred in the static case is very similar, with a slight increase for the Swarm learner, most likely due to rounding errors. In the switching case, the Swarm performs slightly worse than the Gaussian Kernel. The below figures show learner losses, where orange signifies the Gaussian Kernel and green indicates the Swarm learner. The boundary discovered by the learners in the static case is also shown below. Triangles indicate the training set of positive points, and diamonds represent the negative training set points. The blue shaded area marks areas recognised as positive within the grid, and red marks the negative area.



(a) Static

(b) Switching

Figure D.1: Loss Comparison

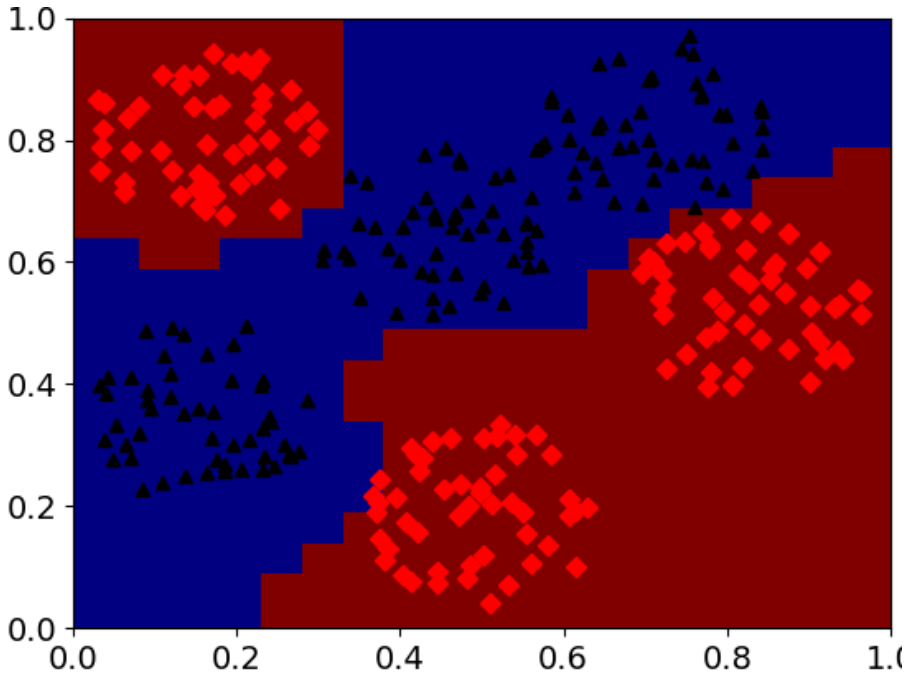


Figure D.2: Discovered Boundary