Adaptive Lifelong Learning

by

Parul
B.Tech., Guru Nanak Dev University, 2015

A Project Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Adaptive Lifelong Learning

by

Parul
B.Tech., Guru Nanak Dev University, 2015

Supervisory Committee

Nishant Mehta, Supervisor
(Department of Computer Science)

Kwang Moo Yi, Departmental Member
(Department of Computer Science)

Supervisory Committee

Nishant Mehta, Supervisor
(Department of Computer Science)

Kwang Moo Yi, Departmental Member
(Department of Computer Science)

ABSTRACT

Lifelong learning is an emerging field in machine learning that still requires a lot of research. In lifelong learning, the tasks are presented sequentially, the system learns knowledge at each task and the goal is to retain the learned knowledge and utilize it when learning a new task. Exponentially Weighted Aggregation for Lifelong Learning (EWA-LL) is a meta-algorithm used in lifelong learning setting. It transfers information from previous tasks to the next. A prior distribution is maintained on the set of representations, which is updated after the encounter of each new task using the exponentially weighted aggregation (EWA) procedure. This project tries to relax the problem and explores the case of an easy scenario where we have some more information about the data. It implements adaptive learning in lifelong learning setting. It utilizes the adaptive learning algorithm Follow The Leader with Dropout Perturbations (FTL-DP) used in Online Prediction with Expert Advice. FTL-DP sets the losses of the experts to 0 or 1 at each task based on the dropout probability before selecting the leader. This project transports FTL-DP to lifelong learning setting. The goal is to prove that adaptive algorithm in lifelong learning is a better approach than EWA-LL as it gives smaller regret for certain easy problems while still maintaining the regret bounds similar to EWA-LL for the harder problems.

Contents

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

my supervisor, Nishant Mehta for his patient guidance and continuous support.

my parents Priya Stoney and Tejinder Singh for being my pillars of strength.

DEDICATION

This work is dedicated to my teachers, family and friends.

Chapter 1

Introduction

The current dominant model of machine learning follows *isolated learning*. It runs a machine learning algorithm on a given dataset and generates a model. Given a task, a machine learning algorithm does not store the knowledge obtained after running the task. Also, it does not utilize any related information gained from the previous tasks. This type of learning is not very similar to human learning as we tend to retain the knowledge learned in past and use that in future learning and to solve new problems [20]. Lifelong learning aims to imitate human learning. It considers systems that can learn tasks sequentially over a lifetime from one or more domains. They retain the knowledge they have learned and use that knowledge to learn new tasks.

This project works in lifelong learning setting in which the tasks are presented sequentially. Furthermore, each task dataset is itself revealed sequentially. At the beginning of each task t, the algorithm selects a representation and within each task, the learner can use an online learning algorithm to solve task t. The representations are the feature maps which transform the input data. All the tasks share a representation parameter which is a weight vector maintained on the set of representations. It helps in transferring information from past tasks to future tasks. We wish to control the regret of the algorithm which is defined as the difference between the cumulative loss suffered by the algorithm and the loss of the oracle. The goal of this paper is to provide an adaptive algorithm that performs better than Exponentially Weighted Aggregation for Lifelong Learning (EWA-LL) algorithm when the data is easy by following the leader (FTL), that is by selecting the representation with the smallest cumulative loss. For the worst-case data, the adaptive algorithm should have the same regret bound as EWA-LL.

1.1 Motivation

The goal of lifelong learning is to learn in the same way as humans do. It can be well-utilized in applications such as intelligent assistants, web agents, chatbots and physical robots. To make a system truly intelligent, it needs to accumulate the learned knowledge. For example, consider the case of a greeting robot for hotels, its job is to call the name of the guests and chat with them. When the robot has learned the data about the existing hotel guests, it calls them every time by their name and have conversation with them. Now on seeing a new guest, it should ask for their name and take their pictures to recognize them and in future when it sees them, it should be able to call their name and chat with them. This requires self-motivated learning by the system (robot in this case) and the need to retain the learned knowledge [5].

Another area where lifelong learning is extensively useful is Sentiment Analysis (SA) as a lot of information is shared across the domains and projects. The sharing of information or knowledge is generally true in many fields. It is important in Natural Language Processing (NLP) for various reasons [5]:

1. The meanings, expressions and syntax used for a language remains the same across domains and tasks. So, the knowledge learned from one task can be well utilized in another task. For example, when we as humans start learning a new subject e.g. Philosophy, we use our past knowledge about the syntax of a sentence to understand the topics, in this way we share knowledge across domains. Now, after studying the basic concepts and definitions in the subject, we use those definitions in the advanced topics, thus sharing knowledge across tasks.

2. Consider the case where an algorithm classifies reviews about a phone as positive or negative. This algorithm can share its knowledge across domains and use it in making predictions for a new task, where we need to classify a review about a tablet. As most NLP problems relate to each other in some way, so lifelong learning can be used in NLP to learn like humans.

Lifelong learning capability is also essential for self-driving cars to work well in the real-life environment. For example, lifelong learning finds its application in image recognition. A self-driving car should learn cumulatively: recognizing each object that it has been trained on, identifying new objects that it has never seen before, and learning to recognize the new objects incrementally.

1.2 Contribution

The aim of this project is to adapt the Follow the Leader with dropout perturbations [23] for the lifelong learning setting.

The results show that for a finite set of representations, if one of the representation always perform better than the others, FTL performs better than EWA-LL, but the results change for worst-case, where EWA-LL performs better. To fix this, Follow the leader with dropout perturbations is used, wherein binarized dropout perturbations are used. This algorithm performs better than EWA-LL in an easy case scenario and still maintains the regret bound similar to EWA-LL in worst-case scenarios.

1.3 Organization

Chapter 1 gives an overview of lifelong learning and discusses the aim of this project. Chapter 2 talks about the related work done in lifelong learning and online learning. Chapter 3 explains Decision Theoretic Online learning (DTOL) and the adaptive algorithms used in online learning setting, AdaHedge and Follow the Leader with dropout perturbations (FTL-DP). Chapter 4 introduces the lifelong learning problem and explains EWA-LL and FTL-DP in lifelong learning setting. Chapter 5 explains the dataset and experimental setup. It evaluates the regrets suffered by different algorithms in lifelong learning setting. Chapter 6 gives a conclusion for the project and discusses the future work which can be carried forward from this project.

Chapter 2

Related Work

[18] gives a good survey of the work done in the field of lifelong learning.

The concept of lifelong learning was introduced by Thrun and Mitchell in 1995 [21]. It has been researched in mainly the following four areas:

1. Lifelong Supervised Learning:
   Lifelong learning techniques in this area were proposed in the context of memory-based learning and neural networks. [8] introduced cumulative learning, which is another form of lifelong learning which builds a new multi-class classifier by updating the old classifier efficiently every time it is introduced to a new class or concept. ELLA (Efficient Lifelong Learning Algorithm) [17] is used for multi-task learning which considers the learning tasks independent of each other. [16] further considered ELLA in an active task selection setting.

2. Lifelong Unsupervised Learning:
   This area covers lifelong topic modeling and lifelong information extraction. [4] proposes a method to mine the prior knowledge automatically and dynamically from topic already found from large number of domains. It also gives a topic model to use this mined knowledge to guide the model inference. [25] proposed a holistic topic model that can model the problems of aspect extraction, opinion identification, polarity classification, and separation of general and aspect-specific opinions simultaneously under a unified framework. [13] approaches the problem of aspect extraction based on framework of lifelong learning. It implements two forms of recommendations which are based on semantic similarity and aspect associations respectively.

3. Lifelong Semi-Supervised Learning:
   [14] has done work in this area. NELL has been reading the web since January 2010 and has created a knowledge base of millions of beliefs. It uses the relationship between the various beliefs in its knowledge base to infer new beliefs.

4. Lifelong Reinforcement Learning:
   In Lifelong reinforcement learning, the agent can accumulate knowledge over a lifetime of experiences and learns how to behave in a environment by performing actions and seeing the results [2]. [22] tried to capture the invariant knowledge about each individual task and the invariant characteristics of the environment for robot learning. [19] proposed an algorithm in which each environment is considered as a task for lifelong learning.

In online learning, algorithms are generally analysed in the worst case. For instance, the Hedge algorithm which is used in most methods of decision theoretic online learning, takes a parameter called the learning rate. In most previous works, the learning rate is tuned to provide better bounds for the worst-case scenario. However, there may be a situation, when you have an action that performs much better than the rest. There has been work done on proposing algorithm that adapts itself based on the difficulty of the problem, that is, it provides optimal performance guarantee in the worst-case scenario but gives smaller regret on easy instances. This project utilizes the approach of Adaptive learning that is used in on-line learning setting.

[7] proposes a new way of setting the learning rate, which adapts to the difficulty of the learning problem. The AdaHedge algorithm guarantees optimal performance in worst-case but achieves much smaller regret on easy instances. The probability produced by Hedge are interpreted as a generalization of Bayesian probabilities. The difficulty of problem is measured in terms of the speed at which the posterior probability of the best action converges to one. [23] describes that it is difficult to get optimal performance for both easy and worst-case scenarios by tuning the learning rate. It introduces the FlipFlop algorithm, which combines the best of FTL and Hedging strategies. It achieves regret within a constant factor of the FTL regret without sacrificing AdaHedge's worst-case guarantees.

Another adaptive algorithm considered for Online Prediction with Expert Advice is Follow the Perturbed Leader (FPL). It perturbs the loss of each expert by independent additive noise drawn from a fixed distribution, and then selects the expert with minimum perturbed loss to make prediction. The expert is selected by breaking the

ties uniformly at random. It needs to tune the noise magnitude as a function of L^*, loss of the best expert in hindsight. In [23] the losses of the experts are randomly set to 0 or 1 before selecting the leader. They prove that this simple, tuning-free version of the FPL algorithm achieves two features: regret bound of $O(\sqrt{L^* \ln K} + \ln K)$ as a function of L^* for the worst-case data, and for the easy case when there is a gap between the expected loss of the best expert and all others it achieves an optimal regret of $O(\ln K)$.

Chapter 3

Adaptive Online Learning

In Decision Theoretic Online Learning(DTOL), an agent has access to K actions, and assigns probability to each action and makes a decision. Next, each action incurs a loss from the range [0,1] and the agent suffers the expected loss under the probability distribution $\mathbf{w_t}$ it produced [7]. Regret is defined as the difference between the cumulative loss suffered by the agent in T rounds and the cumulative loss of the best action which suffers the minimum loss:

$$R(T) = \sum_{t=1}^{T} \mathbf{w}_t . \boldsymbol{\ell}_t - L_T^* \tag{3.1}$$

where $L_T^* = \min_k \sum_{t=1}^{T} \ell_t^k$ is the loss of the best action in hindsight. The algorithms are generally analyzed in the worst case to obtain low regret regardless of the sequence of data.

Hedge algorithm is the most widely used algorithm in this kind of problems. This algorithm is described in 1. Consider the problem with i experts, Hedge keeps weights $w_{i,t}$ over the different experts at each time step and update these weights according to the observed losses. The performance of the Hedge depends on the learning rate η. Also, this algorithm is largely studied and analyzed for worst-cases , in which we get a regret bound of $\sqrt{\frac{T \ln K}{2}}$ if T is known.

Consider a situation, when you have an expert that performs much better than the rest. FTL would be a better approach in this case as it selects the expert with the smallest cumulative loss. But there is a drawback for FTL. Consider the case of antagonistic data for which two experts suffer opposite losses under absolute loss. The loss suffered by first expert is (0,1,0,1,0...) and the loss suffered by second expert

Algorithm 1: Hedge

Initialize $w_{i0} = 1$ for all i.

Loop for t=1 to T:

1. Choose expert $\hat{k}_t$ from categorical distribution p_t where $p_t(i) = w_{i,t} / \sum_j w_{jt}$.
2. Select $x_t = x(\hat{k}_t, t)$, the prediction of $\hat{k}_t$
3. For each i, set $w_{i,t+1} = w_{it}(1-\eta)^{\ell_t(x(k_i,t))}$

is (1/2,0,1,0,1..). Now after the first round, the FTL algorithm selects first expert. After the second round, the cumulative loss of both experts is 1 and 1/2 respectively. So, FTL selects the second expert which performs poorly in round 3. After round 3, the cumulative losses of the experts are 1 and 3/2, so FTL selects the first expert which again suffers a loss of 1 in the next round. FTL would not be a good choice for this setting as it will suffer regret that grows linearly in T.

We are interested in relaxing the problem and see if we can get better bounds if we have some more information about our problem while still maintaining worst-case guarantees. For example, if the cumulative loss of the best action L_T^* is known beforehand, it is possible to achieve a regret bound of $\sqrt{2L_T^* \ln(K) + K}$ by setting the learning rate appropriately [3]. Similarly, if the cumulative empirical variance $VAR_T^{\max} = \max_{t \leq T} VAR_t(\ell_t)$, (where $VAR_t(i)$ is the variation in losses of expert i up to the t^{th} round, and l_t is the best expert till the t^{th} round) of the best expert is known, the regret is bounded by $8\sqrt{VAR_T^{\max} \ln(K)} + 10 \ln(K)$ [10]. There has been work done on proposing algorithm that adapts itself based on the difficulty of the problem, that is, it provides optimal performance guarantee in the worst-case scenario but gives smaller regret on easy instances.

[7] proposes a new way of setting the learning rate, which adapts to the difficulty of the learning problem. The AdaHedge algorithm guarantees optimal performance in worst-case but achieves much smaller regret on easy instances. In AdaHedge, the probabilities produced by Hedge are interpreted as a generalization of Bayesian probabilities. The difficulty of problem is measured in terms of the speed at which the posterior probability of the best action converges to one. For a hard case scenario in AdaHedge, the posterior probability does not converge. And for these instances, it suffers a regret of the optimal order $O(\sqrt{L_T^* \ln(K)})$ where L_T^* is the cumulative loss of the best action. For easy case scenarios, in which the posterior probability converges

sufficiently fast, the regret of AdaHedge is bounded by a constant [23].

Another example for adaptive algorithm in online learning is Follow the Leader with dropout perturbations (FTL-DP) (2). In FTL-DP, the loss of each expert is perturbed by independent additive noise drawn from a fixed distribution, and then the predictions are made with the leader (with the minimum perturbed loss where ties are broken uniformly at random).

In [23], the losses of the experts are randomly set to 0 or 1 before selecting the leader instead of perturbing them with additive noise. This technique has advantage over other adaptive algorithms as it does not involve any complex tuning regime.

For worst-case scenario, the regret for FTL-DP is bounded by $O(\sqrt{L_T^* \ln(K)} + \ln K)$. For the easier problem, when there is a fixed gap between the expected loss of the best expert and all others, the regret nound for FTL-DP improves to $O(\ln K)$ The three types of perturbation technique used in the algorithms are:

1. Follow the Perturbed Leader (FPL)
   In the general Follow the Perturbed Leader (FPL) algorithms [12] [9], the leader $\hat{k_t}$ is selected by

   $$\hat{k_t} = \text{argmin}_k \, L_{t-1,k} + \epsilon_{t-1,k}$$

   where $\epsilon_{t-1,k}$ is the random noise, chosen independently for every expert k, added to the cumulative losses. Experiments have been done with various distributions for $\epsilon_{t-1,k}$.[12] considers exponential and uniform distributions and [6] considers binomial distribution.

2. Binarized Dropout Perturbations (FTL-DP)
   [23] implements FPL based on Binarized Dropout Perturbations (BDP). For any dropout probability $\alpha \in (0, 1)$, the binarized dropout perturbation of loss $\ell_{t,k}$ is defined as:

   $$\hat{\ell}_{t,k} = \begin{cases} 1 \text{ with probability } (1-\alpha)\ell_{t,k} \\ 0 \text{ otherwise} \end{cases} \tag{3.2}$$

   Let $\hat{L}_{T,k} = \sum_{t=1}^{T} \hat{\ell}_{t,k}$ be the cumulative BDP loss for expert k. Then the BDP algorithm chooses

$$\hat{k}_t = \text{argmin}_k \, \tilde{L}_{t-1,k},$$

with ties broken uniformly at random.[23]

This technique is computationally efficient as it does sparse updates. It involves a parameter α but that only affects the constants. BDP may be viewed as an instance of FPL with the additive data-dependent perturbations

$$\epsilon_{t-1,k} = \tilde{L}_{t-1,k} - L_{t-1,k}[23]$$

3. Standard Dropout Perturbations

   The standard definition of dropout is omitting hidden units or features in a neural network while training it's parameters on batch data using Gradient Descent [24][26][11]. [23] explains the single neuron case, in which each expert may be recognized with a feature and the *standard dropout* perturbations (without binarization) become:

$$\hat{\ell}^s_{t,k} = \begin{cases} \ell_{t,k} \text{ with probability (1-}\alpha\text{)} \\ 0 \text{ otherwise} \end{cases}, \tag{3.3}$$

Algorithm 2: Follow the Leader with binarized dropout perturbations

---

Parameter: Dropout probability $\alpha \in (0, 1)$
Initialization: $\tilde{L}_{0,k} = 0$ for all $k = 1, ..., K$
Loop for t=1 to T:

1. Pick expert $\hat{k}_t = \text{argmin}_k \, \tilde{L}_{t-1,k}$ (with ties broken uniformly at random).
2. Observe loss vector $\ell_{\mathbf{t}}$ and suffer loss $\ell_{t,\hat{k}_t}$.
3. Draw $\tilde{\ell}_{t,k}$ according to 3.2, independently for all k.
4. Update $\tilde{L}_{t,k} = \tilde{L}_{t-1,k} + \tilde{\ell}_{t,k}$ for all k.

Chapter 4

Adaptive Lifelong Learning

Lifelong machine learning (or lifelong learning) is an advanced form of machine learning that learns continuously, retains the knowledge learned in previous tasks, and uses it in learning future tasks and predicting new tasks. [5]. Machine learning algorithms work in isolation, they get a training dataset and make predictions based on that but do not retain or accumulate the knowledge they gained, so given the same dataset twice, a machine learning algorithm would start learning from scratch instead of utilizing the knowledge that it already gained. So, to make the machine learning systems truly intelligent, lifelong machine learning is used.

This project implements the lifelong learning problem in an online setting [1]. The problem under consideration is to implement EWA-LL, Follow the Leader and Follow the Leader with dropout perturbations [23] in this setting and analyze their performance for easy case and worst-case scenarios. The tasks are presented sequentially [1]. We let Z be a set and prescribe a set G which has K representations $g : X \to Z$ and a set H of predictors $h : Z \to \mathbb{R}$. For each task t, the learner is challenged with

$$S_t = ((x_{t,1}, y_{t,1}), ...(x_{t,N}, y_{t,N})) \in (X, Y)^N \tag{4.1}$$

where N is the within-task sample size. Next, the dataset S_t is also presented sequentially. The object $x_{t,i}$ is revealed, learner predicts $\hat{y}_{t,i}$, label $y_{t,i}$ is revealed and the learner incurs the loss $\hat{\ell}_{t,i} = \ell(\hat{y}_{t,i}, y_{t,i})$.

The prediction error for task t is

$$\frac{1}{N} \sum_{i=1}^{N} \hat{\ell}_{t,i} \tag{4.2}$$

At the end of all the tasks, average error is

$$\frac{1}{T}\sum_{t=1}^{T}\frac{1}{N}\sum_{i=1}^{N}\hat{\ell}_{t,i} \tag{4.3}$$

Now, say for a given representation g, the best predictor h_t for task t is known in advance, then an ideal learner would get the error

$$\inf_{h_t\in H}\frac{1}{N}\sum_{i=1}^{N}\ell(h_t\circ g(x_{t,i}),y_{t,i}) \tag{4.4}$$

Hence the within task regret of representation g is

$$R_t(g)=\frac{1}{N}\sum_{i=1}^{N}\hat{\ell}_{t,i}-\inf_{h_t\in H}\frac{1}{N}\sum_{i=1}^{N}\ell(h_t\circ g(x_{t,i}),y_{t,i}) \tag{4.5}$$

The normalization in 4.5 gives equal weights to different tasks. An oracle who knows the best representation g for all tasks in advance, would have suffered the error

$$\inf_{g\in G}\frac{1}{T}\sum_{t=1}^{T}\inf_{h_t\in H}\frac{1}{N}\sum_{i=1}^{N}\ell(h_t\circ g(x_{t,i}),yt,i) \tag{4.6}$$

The performance of the algorithms is compared based on the regret suffered by the algorithm, which is defined as:

$$R=\frac{1}{T}\sum_{t=1}^{T}\frac{1}{N}\sum_{i=1}^{N}\hat{\ell}_{t,i}-\inf_{g\in G}\frac{1}{T}\sum_{t=1}^{T}\inf_{h_t\in H}\frac{1}{N}\sum_{i=1}^{N}\ell(h_t\circ g(x_{t,i}),y_{t,i}), \tag{4.7}$$

4.1 Algorithm

There are three algorithms that are considered in this project. EWA-LL [1] algorithm, FTL and FTL-DP. The goal of the project is to implement FTL and FTL-DP [23] in lifelong learning setting and to show that how well does these algorithms perform when compared to EWA-LL. The project explores the setting of finite subset of relevant predictors.

4.1.1 EWA-LL

The EWA-LL algorithm [1] for fixed number of representations is outlined in Algorithm 3. It considers *online-within-online* lifelong learning setting. The tasks are presented sequentially and then the N samples within a task also reveal sequentially. For each task t, the algorithm selects a representation $\hat{g}_t$ based on the probability vector w_t. For the online learning problem within task t, we reveal the N samples sequentially and make a prediction using Online Gradient Descent algorithm (OGA) with representation $\hat{g}_t$. The within task algorithm suffers loss using representation $\hat{g}_t$. It also observes the loss suffered by all other representations. In step iii), the weight vector is updated using exponentially weighted aggregation procedure. Any representation g which does not perform well on task t, is less likely to be reused on the next task. For the finite predictors case, [1] gives a bound of $O(1/m)$ which is an improvement from $O(1/\sqrt{m})$ bound derived by [15]. When the within-task algorithm has a regret bound, EWA-LL inherits this good property [1].

For the worst-case, [1] gets a bound on the expected regret but it does not adapt to the problem, that is given some additional information about the data, such as the best representation, it would still run the algorithm in the same way and would not change its strategy or tune its learning rate.

4.1.2 Follow The Leader (FTL)

FTL guarantees constant regret in stochastic setting, but has poor performance for worst-case data. FTL selects the representation in round t that suffers minimum loss over all the past rounds $(1, ..., t-1)$ as outlined in Algorithm 4.

FTL improves the performance for the case when we have a better representation that performs better than all other representations.

4.1.3 Follow the Leader with dropout perturbations

The two main algorithm considered in online prediction with expert advice are Hedge/ Weighted Majority and Follow the Perturbed Leader. In FTPL, the loss of each expert is perturbed by independent additive noise drawn from a fixed distribution, and then the predictions are made with the leader (with the minimum perturbed loss where ties are broken uniformly at random).

In [23], the losses of the experts are randomly set to 0 or 1 before selecting the

Algorithm 3: EWA-LL

Data: A sequence of datasets
$S_t = ((x_{t,1}, y_{t,1}), ..., (x_{t,m_t}, y_{t,m_t}))$, $1 \leq t \leq T$. associated with different learning tasks; the points within each dataset are also given sequentially.

Input: A weight vector $\mathbf{w}$ on the representations, a learning parameter $\eta \geq 0$,K number of representations and a learning algorithm for each task t which, for any representation g_i $i = 1, , , K$ where K is the number of representations returns a sequence of predictions $y^{\hat{g}}_{t,i}$ and suffers a loss

$$\hat{L}_t(g) := \frac{1}{m_t} \sum_{i=1}^{m_t} \ell \left(y^{\hat{g}}_{t,i}, y_{t,i} \right).$$

Loop For $t = 1, ..., T$

1. A representation $\hat{g}_t$ is selected with probability w_t
2. Run the within-task learning algorithm on S_t and suffer loss $\hat{L}_t(\hat{g}_t)$.
3. Update $w_{t+1} = \frac{\exp(-\eta \hat{L}_t(\hat{g}_t)) w_t(\hat{g}_t)}{\sum_{i=1}^{K} \exp(-\eta \hat{L}_t(g_i)) w_t(g_i)}$

leader instead of perturbing them with additive noise. This technique has advantage over other adaptive algorithms as it does not involve any complex tuning regime.

This project implements FTL-DP based on Binarized Dropout Perturbations (BDP) as described in Algorithm 5. For any dropout probability $\alpha \in (0, 1)$, the binarized dropout perturbation of loss l_{t,g_i} is defined as:

$$\hat{\ell}_{t,g_i} = \begin{cases} 1 \text{ with probability } (1\text{-}\alpha) \ell_{t,g_i} \\ 0 \text{ otherwise} \end{cases} \tag{4.8}$$

Let $\tilde{L}_{T,g_i} = \sum_{t=1}^{T} \hat{\ell}_{t,g_i}$ be the cumulative BDP loss for representation g. Then the BDP algorithm chooses

$$\hat{g}_t = \text{argmin}_i \, \tilde{L}_{t-1,g_i},$$

with ties broken uniformly at random [23]. Next, the within-task algorithm is run using the representation $\hat{g}_t$. The loss suffered is observed and then $\tilde{L}_{T,g}$ is updated.

This technique is computationally efficient as it does sparse updates. It involves

Algorithm 4: Follow the Leader

Data: A sequence of datasets
$S_t = ((x_{t,1}, y_{t,1}), ..., (x_{t,m_t}, y_{t,m_t}))$, $1 \leq t \leq T$. associated with different learning tasks; the points within each dataset are also given sequentially.
$\hat{L}_t(g) := \frac{1}{m_t} \sum_{i=1}^{m_t} \ell\left(\hat{y}_{t,i}^g, y_{t,i}\right)$. is the loss suffered in task t by representation g
$L_t(g_k) = \sum_{i=1}^{t} (\ell_{i,g_k})$ where ℓ_{i,g_k} is the loss suffered by representation g_k in round i.

Loop t=1 to T:

1. Select representation $\hat{g}_t = \text{argmin}_i \, L_t(g_i)$
2. Run the within-task learning algorithm on S_t and suffer loss $\hat{L}_t(\hat{g}_t)$.

a parameter α but that only affects the constants.

Algorithm 5: Follow the Leader with dropout perturbations

Input: $0 < \alpha < 1$
$\tilde{L}_{0,g_i} = 0$ for all $i = 1, , K$
Loop for t=1 to T:

1. Select representation $\hat{g}_t = \text{argmin}_i \, \tilde{L}_{t-1,g_i}$
2. Run the within-task learning algorithm on S_t and and observe the loss ℓ_{t,g_i} for all the representations (i=1,,K) and suffer loss $\ell_{t,\hat{g}_t}$
3. for i=1,..,K:
$\tilde{\ell}_{t,g_i} = 1$ with probability $(1-\alpha)\ell_{t,g_i}$,
0 otherwise
$\tilde{L}_{t,g_i} = \sum_{j=1}^{t} \tilde{\ell}_{j,g_i}$

4.1.4 Within-Task Algorithm

Online Gradient Descent

For the within task algorithm, we implement OGA (Online Gradient Descent Algorithm). For a given task t the samples (X_N, Y_N) are revealed one by one. The prediction is made using all the representations and the loss vector is observed. The

Algorithm 6: Online Gradient Descent

Data: A task $S_t = ((x_{t,1}y_{t,1}), ..., (x_{t,N}, y_{t,N}))$
Input: Stepsize $\chi > 0$ and $\theta_{\mathbf{1}} = \mathbf{0}$
Loop For $i = 1, ..., N$
i. Predict $\hat{y}^g_{t,i} = h_{\theta_i} \circ g(x_{t,i})$,
ii. $y_{t,i}$ is revealed, update

$$\theta_{\mathbf{i+1}} = \theta_{\mathbf{i}} - \chi \Delta_\theta \ell(h_\theta \circ g(x_{t,i}), y_{t,i})|_{\theta=\theta_i}$$

loss suffered by EWA-LL, FTL and FTL-DP is also noted. The weight vector initialized as zero, is updated after every sample based on the loss observed using gradient descent algorithm.

Chapter 5

Evaluation and Results

5.1 Data

The dataset used has T=1000 tasks with N=10000 within-task samples. The project explores the setting of finite subset of relevant predictors in which we assume we have K representations. For task t, where $t = 1, ..., T$, the dataset has d-dimensional input samples $\mathbf{X_N}$ generated using random normal distribution with 0 mean and standard deviation of 0.1. Each data point is normalized to have unit Euclidean norm. To get the labels, a weight vector $\mathbf{w}$ is generated using random normal distribution (mean=0, standard deviation=0.1) and normalized to have unit Euclidean norm. Next the labels for the input samples in task t are generated using $\mathbf{y_t} = \mathbf{X_N} \cdot \mathbf{w}$. An independent, zero-mean Gaussian noise is added to the N labels $(y_1, ..., y_N)$.

5.2 Experimental Setup

In this section we introduce our notation. We consider the problem of regression, as described in [1]. Let X and Y be some sets. A predictor is a function $f : X \to Y$ where $Y = \mathbb{R}$ The loss is denoted by $\ell(f(x), y)$. In lifelong learning , the aim is to transfer the information (a common data representation) obtained from the previous tasks to a new one. We let Z be a set and prescribe a set G which has K representations $g : X \to Z$ and a set H of predictors $h : Z \to \mathbb{R}$.

The set H of functions is obtained using Online Gradient Descent for the within-task samples for each task. The loss function used is a square loss, so that $\ell(f(x), y) = (f(x) - y)^2$

For the experiments, task size is T= 1000, within-task sample size is m_t=10000. Learning rate for EWA-LL is $\eta = \frac{2}{C}\sqrt{\frac{2\log K}{T}}$, where C is a constant that upper bounds the loss. Learning rate used for the within-task Online Gradient algorithm is $\chi = \frac{B}{L\sqrt{2m_t}}$ where L is the Lipschitz constant for the loss function and B is the upper bound of the norm of θ, that is $||\theta|| \leq B$. The loss function used is square loss.

The resulting regret is the average taken over 5 runs, as the results were very consistent, so 5 runs were good enough to get the regret.

5.3 EWA-LL vs FTL vs FTL-DP

5.3.1 An easier scenario

In an easier scenario, the dimensionality of data is d=20 and the number of representations is $K = d + 1 (= 21)$, where 0th representation is the exact identity map which maps x to $(x_0, ...x_{d-1})$ and for $i = 1, ..., K-1$ representation i is the almost identity map, which maps x to $(x_0, ..., x_{i-2}, 0, x_i, ..., x_{d-1})$.

Now, when we have a representation that always performs better than the other representations, we see from Figure 5.1 that FTL gives much smaller regret than EWA-LL as once it is able to find the leader, which is the best representation in this case, it keeps on selecting that representation. So, if we have prior knowledge about our data, FTL is a better approach to follow. Also, this is a computationally efficient technique to follow.

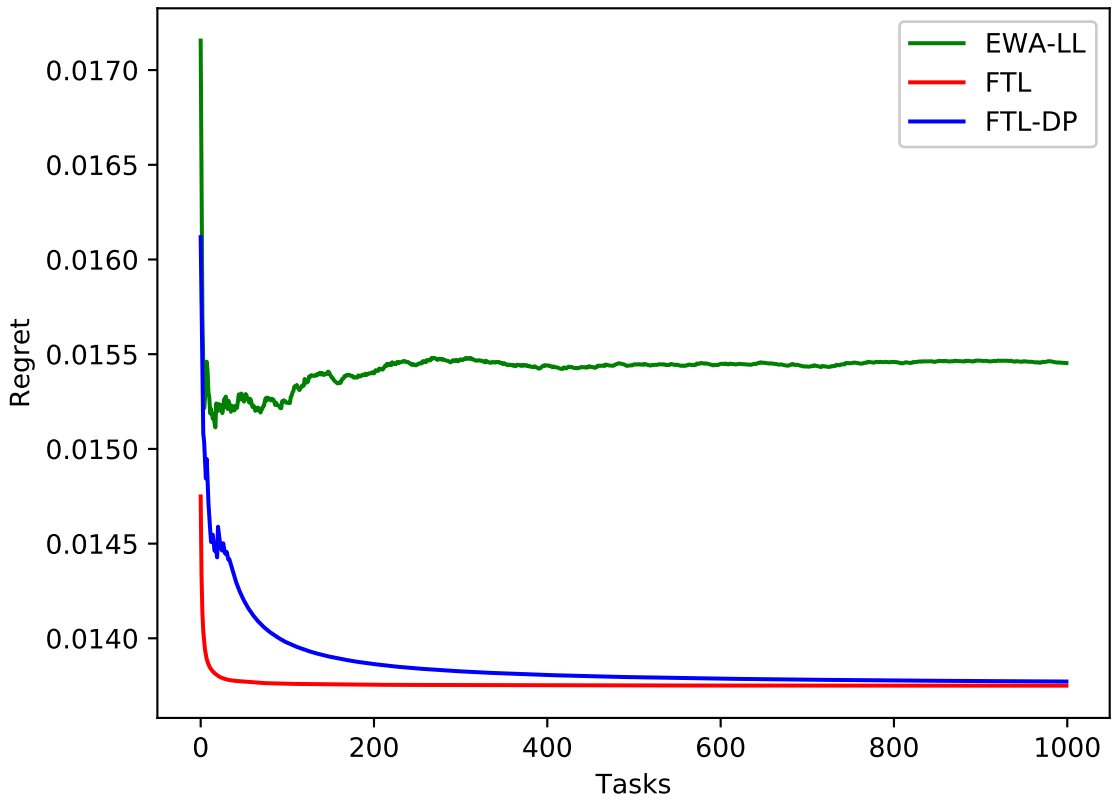

Figure 5.1: EWA-LL vs FTL vs FTL-DP for an easy case

5.3.2 Worst-case Scenario

For the worst-case, we consider the scenario with two representations. For odd-numbered tasks, the first representation acts as the best representation, mapping x to $(x_0, ...x_{d-1})$ whereas the second representation maps x to $\mathbf{0}$. For even-numbered tasks, the second representation is the best which maps x to $(x_0, ...x_{d-1})$ and the first representation maps x to $\mathbf{0}$. Follow the Leader picks up the representation that was best in the past, but that best-so-far representation performs worst on the next task.

EWA-LL quickly learns to randomize equally between both the experts, and so it performs much better than FTL.

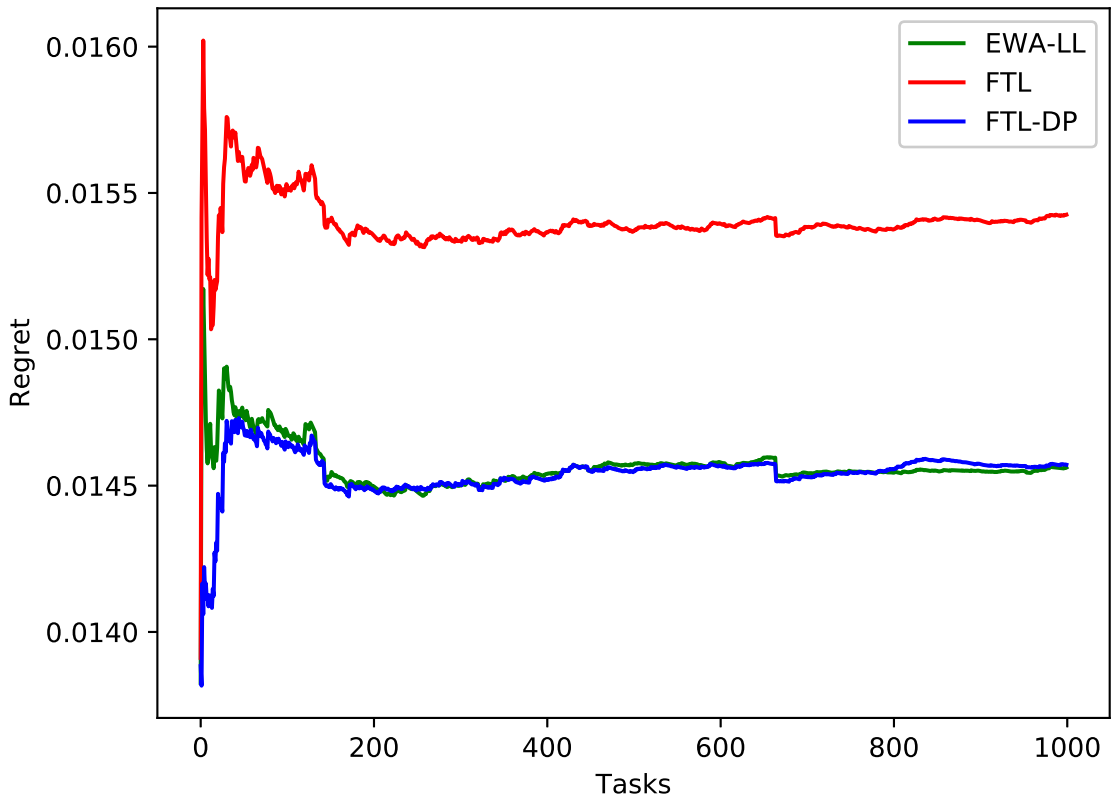

Figure 5.2: EWA-LL vs FTL vs FTL-DP for the worst-case

5.4 Results

To get the best of both the worlds, we implemented Follow the Leader with Dropout Perturbations. Figure 5.1 shows the for an easy case scenario, it performs almost as good as FTL and much better than EWA-LL. In the worst-case scenario, it still maintains regret bounds similar to EWA-LL.

Follow the Leader with dropout perturbations performs well in the worst case scenario (Figure 5.2) too because the perturbation of the cumulative loss of representation g depends on the loss of representation g in the the given task t.

Chapter 6

Conclusion and Future Work

In this project, the aim is to use an adaptive algorithm that can consider which type of data is under consideration. It implements FTL-DP in the lifelong learning setting. For the easy case scenario, the algorithm gives performance similar to FTL. In the worst-case scenario the algorithm overcomes the drawbacks of FTL and gives regret bounds similar to EWA-LL.

This project is a preliminary work done in the field of adaptive lifelong learning. This project gives empirical proof that the adaptive algorithms, even as simple as Follow the Leader with dropout perturbations perform better than EWA-LL. The aim of the this project is to motivate the research for theoretical proofs for regret bounds of the the adaptive lifelong learning. [23] provides proofs for better regret bounds in adaptive online learning with FTL-DP. Their work can be further utilized in lifelong learning.

Also, this project implemented the within-task algorithm using OGA. In future, various algorithms such as Exponentially Weighted Aggregation can be used for the within-tasks and the performance can be observed. Complex adaptive algorithms such as AdaHedge and FlipFlop implemented in online prediction with expert advice setting, can also be explored as meta-algorithms for lifelong learning.

Bibliography

[1] Pierre Alquier, Massimiliano Pontil, et al. Regret bounds for lifelong learning. In *Artificial Intelligence and Statistics*, pages 261–269, 2017.

[2] Haitham Bou Ammar, Rasul Tutunov, and Eric Eaton. Safe policy search for lifelong reinforcement learning with sublinear regret. In *International Conference on Machine Learning*, pages 2361–2369, 2015.

[3] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[4] Zhiyuan Chen and Bing Liu. Topic modeling using topics from many domains, lifelong learning and big data. In *International Conference on Machine Learning*, pages 703–711, 2014.

[5] Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(3):1–145, 2016.

[6] Luc Devroye, Gábor Lugosi, and Gergely Neu. Prediction by random-walk perturbation. In *Conference on Learning Theory*, pages 460–473, 2013.

[7] Tim V Erven, Wouter M Koolen, Steven D Rooij, and Peter Grünwald. Adaptive hedge. In *Advances in Neural Information Processing Systems*, pages 1656–1664, 2011.

[8] Geli Fei, Shuai Wang, and Bing Liu. Learning cumulatively to become more knowledgeable. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1565–1574. ACM, 2016.

[9] James Hannan. Approximation to bayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139, 1957.

[10] Elad Hazan and Satyen Kale. Extracting certainty from uncertainty: Regret bounded by variation in costs. *Machine learning*, 80(2-3):165–188, 2010.

[11] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[12] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

[13] Qian Liu, Bing Liu, Yuanlin Zhang, Doo Soon Kim, and Zhiqiang Gao. Improving opinion aspect extraction using semantic similarity and aspect associations. In *AAAI*, pages 2986–2992, 2016.

[14] Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bo Yang, Justin Betteridge, Andrew Carlson, B Dalvi, Matt Gardner, Bryan Kisiel, et al. Never-ending learning. *Communications of the ACM*, 61(5):103–115, 2018.

[15] Anastasia Pentina and Christoph Lampert. A pac-bayesian bound for lifelong learning. In *International Conference on Machine Learning*, pages 991–999, 2014.

[16] Paul Ruvolo and Eric Eaton. Active task selection for lifelong machine learning. In *AAAI*, 2013.

[17] Paul Ruvolo and Eric Eaton. Ella: An efficient lifelong learning algorithm. In *International Conference on Machine Learning*, pages 507–515, 2013.

[18] Daniel L Silver, Qiang Yang, and Lianghao Li. Lifelong machine learning systems: Beyond learning algorithms. In *AAAI Spring Symposium: Lifelong Machine Learning*, volume 13, page 05, 2013.

[19] Fumihide Tanaka and Masayuki Yamamura. An approach to lifelong reinforcement learning through multiple environments. In *6th European Workshop on Learning Robots*, pages 93–99, 1997.

[20] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in neural information processing systems*, pages 640–646, 1996.

[21] Sebastian Thrun. *Explanation-based neural network learning: A lifelong learning approach*, volume 357. Springer Science & Business Media, 2012.

[22] Sebastian Thrun and Tom M Mitchell. Lifelong robot learning. In *The biology and technology of intelligent autonomous agents*, pages 165–196. Springer, 1995.

[23] Tim Van Erven, Wojciech Kotłowski, and Manfred K Warmuth. Follow the leader with dropout perturbations. In *Conference on Learning Theory*, pages 949–974, 2014.

[24] Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. In *Advances in neural information processing systems*, pages 351–359, 2013.

[25] Shuai Wang, Zhiyuan Chen, and Bing Liu. Mining aspect-specific opinion using a holistic lifelong topic model. In *Proceedings of the 25th international conference on world wide web*, pages 167–176. International World Wide Web Conferences Steering Committee, 2016.

[26] Sida Wang and Christopher Manning. Fast dropout training. In *international conference on machine learning*, pages 118–126, 2013.