Computational Learning Theory Final Project

Lesley Cordero (lc2958), Ojas S. Sathe (oss2116)

December 17, 2015

1 Introduction and Background

Machine Learning is the study and development of algorithms that use data to make predictions. Such "learning" is often done by giving an algorithm a training set of examples and labels that teach the algorithm how to predict in the future.

A fundamental model of learning is "online" learning, in which a learning algorithm formulates learning hypotheses in "real-time," based only on knowledge of the past[1]. In Manfred Opper's paper, "A Bayesian Approach to Online Learning," online learning is analyzed through the framework of Bayesian Statistical Inference techniques to achieve asymptotic efficiency and optimal online learning.

1.1 Online Learning

Computational Learning Theory is the branch of machine learning which focuses primarily on analyzing the computational complexity of machine learning algorithms. An application of this is constructing optimal techniques, the basis of Opper's paper.

1.1.1 Optimal Online Learning

Opper is quick to mention that surely there is "no *uniformly* optimal estimation strategy," as "uniform optimality can be achieved only within special subclasses of estimators" (3). Opper mentions that one must consider the bias and variance of the estimators when looking for optimality. He also cites various optimality criteria such as the *minimax principle*, optimizing estimators for worst true density, and the approach he would go on to use, the *Bayesian* approach, where one defines "an average case optimality" over a prior distribution $p(\theta^*)$ of true parameters θ^* (3). Further details on this section can be found in section 3.4 of this paper.

1.1.2 Online Learning Algorithms

One framework for an algorithm to perform online learning is the Online Mistake Bound (OLMB) model discussed at length in Avrim Blum's survey paper[1]. We proved in class

that for every online learning algorithm, there is a cautious analog; i.e an algorithm that only changes its hypothesis on mistakes. From this, we can concur (without loss of generality) that such an algorithm A will make at most $m+1$ hypotheses, where m is the total number of mistakes made by the algorithm.

1.2 Batch Learning

Batch learning is a type of supervised machine learning framework, where the learning instance face is defined to be an entire training set of data. Using this training set, a hypothesis is output as the learning model of prediction. Clearly this is distinguishable from the online algorithm description in section 1.1, as this involves learning from an entire training set at once and adjusting the parameters to work within that model. The Bayesian framework discussed here can be used to define optimal online learning (Section 1.1.1) as an approximation to batch learning, focusing on minimizing the loss of information from discarding previous examples (2).

1.2.1 Probably Approximately Correct Learning Algorithms

As Online Mistake Bound algorithms (mentioned in Section 1.1.2) exist in the Online learning framework, Probably Approximately Correct (PAC) learning algorithms exist in the Batch learning framework. PAC Algorithms require a set of labeled examples (drawn independently from a probability distribution $\mathcal{D}$ over X.

We say that an algorithm $A^{'}$ PAC learns concept class $\mathcal{C}$ if, for any target concept $c \in \mathcal{C}$, for any distribution $\mathcal{D}$ over X, for any accuracy parameter, $\epsilon > 0$, and for for any confidence parameter, $\delta > 0$, with probability greater than or equal to $1 - \delta$, $A^{'}$ outputs a hypothesis $h \in \mathcal{H}$ such that:

$$\mathrm{P}[h(x) \neq c(x)] \leq \epsilon$$

1.2.2 OLMB to PAC Conversion

Despite the fundamental difference between Online Mistake Bound algorithms in the Online Learning framework and Probably Approximately Correct algorithms in the Batch learning framework, OLMB algorithms can be converted to PAC algorithms.

We discussed earlier (in section 1.1.2) that for each OLMB algorithm, there exists a cautious analog that only changes it's hypothesis on mistakes. Thus an OLMB algorithm, A for concept class $\mathcal{C}$ with a total of m mistakes will use at most $m+1$ hypotheses. A PAC algorithm for this same concept class ($\mathcal{C}$) works by reading enough examples (in this case: $\frac{1}{\epsilon}\ln(\frac{m+1}{\delta})$) to guarantee that the hypothesis outputted is ϵ-accurate and δ-confident.

1.3 Neural Networks

Neural Networks are a family of models that emerged from the field of deep learning, and are based (as the name may suggest) on the nervous systems (in particular the brain) of animals. The neural network model is characterized by a varied number of unknown inputs. A key aspect of neural networks is that they contain a set of adaptive weights (similar to several of the algorithms we've learned over the course of the semester, Winnow2, Weighted Majority, Adaboost). Neural networks also are capable of approximating non-linear functions of their inputs[2].

In class, we briefly discussed how non-linear functions could be approximated. We learned about how the basic Perceptron algorithm could be used to find a target linear threshold function, and how the dual Perceptron (capable of being "Kernelized"), allowed the Perceptron to be run over high-dimensional spaces. Interestingly, its first implementation (in custom hardware) was one of the first neural networks to be produced[3].

In the following section (1.4), we discuss some of the statistical inference concepts used in the field of neural networks. We then build upon this information in section 4.1 to gain further understanding as to how Bayesian Statistics are relevant to neural networks.

1.4 Statistical Inference

In order to properly understand the details of "A Bayesian Approach to On-line Learning," a great deal of statistical inference concepts should be addressed, as they are essential to understanding the nuances of the described algorithm, as well as its application to neural networks.

Fundamentally, this optimal online algorithm expands upon the fact that it can learn from random examples. Given a dataset $D_t = (y_1, ..., y_t)$, where t is the number of independently generated examples, a sample is generated randomly according to the distribution,

$$P(D_t|\theta) = \prod_{k=1}^{t} P(y_k|\theta)$$

where θ is the unknown parameter that must be estimated from D_t- this estimation can be found in section 3.3.

2 Online Learning for Estimation

Before we delve into the algorithm, it is necessary for us to understand how online algorithms estimate unknown parameters. As was mentioned in section 1.4, θ de-

notes the unknown parameter, and by the principle of Maximum Likelihood (ML), we should "choose a parameter θ which maximizes the likelihood $P(D_t|\theta)$ of the observed data"(4).

The goal of this online learning example is to calculate a new estimate $\hat{\theta}(t+1)$ which is based on the new data point y_{t+1} "and possibly a set of auxiliary quantities which have to be updated at each time step" but are much smaller than $||D_t||$(5).

Opper chooses to use a procedure that uses a gradient descent algorithm (first order optimization) and iterates until convergence is achieved. In the online learning scheme, the following equation is presented to find the difference in estimates, using E_T as the function of training energy of all the examples. This is the energy required by the learner to update its estimate. Naturally, in batch learning, E_T would be much higher than in the online environment.

$$\begin{aligned}\hat{\theta}_i(t+1) - \hat{\theta}_i(t) &= \eta(t)\partial_i \ln P(y_{t+1)|\hat{\theta}(t)} \\ &= \eta(t)\partial_i E_T(y_{t+1)|\hat{\theta}(t)}\end{aligned}$$

In the above equation we see a new term, $\eta(t)$, defined as the learning rate. For the algorithm to converge asymptotically, η must be decreased during learning. This is relevant to the goal of the overall algorithm, as the learning rate is directly correlated with how the algorithm approaches the optimal parameter.

3 Algorithmic Details

As previously mentioned, the Bayesian Statistical Inference Algorithm can be split into two general steps, the first being the approximate posterior update and the second being optimal projection. In the following two sections, we will thoroughly but concisely go through the details of each step.

3.1 Step 1: Approximate Posterior Update

As a brief explanation, this step of the algorithm can be described as following: when a a new example of data arrives, the approximate posterior is updated. This is repeatedly done in conjunction with the second step until the algorithm achieves asymptotic efficiency, further described in section 3.3.

To figure how the posterior distribution changes when a new data point is viewed, the posterior distribution of all the previous data points can be used. However, this methodology would be computationally expensive as it would require iteration of all previous examples or a computationally expensive amount of space. Indeed, Opper mentions that the previous method of updating the posterior required the entire old

dataset D_t (i.e. batch learning). In contrast of our method of an OLMB algorithm becoming a PAC algorithm (section 1.2.2), Opper creates an an online "version" of the old algorithm that required batch processing. In other words, Opper creates a set of parameters that captures the essential information of the previous data and keeps track of and updates it at each step.

Given all these adjustments, the following formula can be applied to obtain the new approximate posterior.

$$p(\theta|y_{t+1}, par(t)) = \frac{P(y_{t+1}|\theta)p(\theta|par(t))}{\int d\theta P(y_{t+1}|\theta)[(\theta|par(t))}$$

3.2 Step 2: Optimal Projection

After step 1, the parametric family needs to be synced. In other words, at this point we have $p(\theta|par(t))$ when $p(\theta|par(t+1))$ is needed. In order to accomplish this, the parameter par(t+1) has to be chosen such that the difference between $p(\theta|par(t+1))$ and $p(\theta|y_{t+1}, par(t+1))$ is as close to 0 as possible.

At this point, design decisions are more flexible. What algorithm you use to to measure the differences between the different distributions is up to interpretation, though Opper chooses KL-divergence in his paper, shown below:

$$D_{KL}(p(\cdot|y_{t+1}, par(t))||p(\cdot|par)) = \int d\theta p(\theta|y_{t+1}, par(t) \ln \frac{p(\theta|y_{t+1}, par(t))}{p(\theta|par)}$$

3.3 Gaussian Ansatz

To elaborate further on step 1, we'll discuss how $p(\theta|par)$ can be used to update the posterior mean. In this paper's implementation of Bayes On-Line Learning, a general multivariate Gaussian distribution is used for $p(\theta|par)$, where par = (mean, covariance) $= (\hat{\theta}_i, C_{ij})$. Using the Gaussian approximation to the posterior technique, an update for the posterior mean is achieved.

Since $E(zf(z)) = E(f'(z)) \cdot E(z^2)$, according to Gaussian random variable properties for well behaved functions, we can update par as follows, where u is a zero mean Gaussian random vector with covariance $C(t)$ and the expectation from step 2 is written as $E_u[P(y_{t+1}|\hat{\theta}(t) + u)]$:

$$\hat{\theta}(t+1) = \hat{\theta}(t) + \Sigma_j C_{ij}(t) x \partial_j ln E_u[P(y_{t+1}|\hat{\theta}(t) + u)]$$

$$C_{ij}(t+1) = C_{ij}(t) + \Sigma_{kl} C_{ik}(t) C_{lj}(t) x \partial_k \partial_l ln E_u[P(y_{t+1}|\hat{\theta}(t) + u)]$$

3.4 Asymptotic Efficiency

The foundation of Bayesian Online Learning is asymptotic efficiency of the algorithm. Once the algorithm reaches asymptotic efficiency, an optimal model has been formed. This is particularly important for batch learning (an application of Bayes On-line Learning, discussed in section 4 of this paper), as it gives rise to maximum likelihood estimators.

Furthermore, when optimal learning is used in the context of batch learning, it can shown that learning can be done within the same asymptotic speed in some cases. Though there are limitations to this fact, further discussed in section 5, its results are still highly important to a field whose primary goal is to analyze the computational complexity of machine learning algorithms.

Despite there being no uniformly optimal estimation strategy, uniform optimality can be achieved within special subclasses of estimators. Section 3 of Opper's paper discusses various statistical optimality criteria, highlighting the minimax principle, which optimizes predictions for the worst true density.

Just as importantly, efficiency in the context of parameter estimation is explained in section 3 of Opper's paper. He explains that $\hat{\theta}$'s optimality can be forced into approaching θ^* at a certain asymptotic speed through the Rao-Cramer Inequality. It can be shown that no estimator can perform better than Rao-Cramer's Inequality as the amount of data grows under smoothness assumptions. Such requirement is shown below:

$$E_{D_t}(\hat{\theta}_i - \theta_i^*)(\hat{\theta}_j - \theta_j^*) = \frac{1}{t}\left(J^{-1}(\theta*)\right)_{ij}$$

3.5 Implementation

In the pseudocode provided below, we use "blackbox" functions to represent the mathematically intensive parts since our goal was to represent the overall technique in its simplest form. It should be strongly noted that the following pseudocode does not provide any new information; rather, it's just another form of our learning process of this paper.

```
def bayes_online(firstPoint):
    par = [ ] # set of pars from prev data
    def bayes_steps(newPoint):
        while not check_eff():
            update_approxPost()
            # info.extract(): new information obtained
            par.append(info.extract())
            opt_projection()
        newPoint = getData() # returns a new data point
```

```
10
11              bayes_steps(newPoint)
12
13      def update_approxPost():
14          ''' Blackbox algorithm to update approximate posterior '''
15
16      def opt_projection():
17          ''' Blackbox algorithm for optimal projection '''
18
19      def check_eff():
20          ''' Black box algorithm to check asymptotic efficiency
21              Details can be found in section 3.4 '''
22              return bool #indicates if efficient or not
```

Listing 1: Optimal Online Learning Pseudocode

4 Applications

Looking at Online learning through the lens of Bayesian Statistical Inference yields the best estimator for *all* true parameters θ^*. The applications of statistical inference are increasingly relevant to the study Neural Networks.

In the following section, we study the details of Neural Networks in the context of Bayesian Statistical Inference. Background information can be found in section 1.3 of this paper.

4.1 Neural Networks

Applying the statistical inference concepts in section 1.4 and given a noisy classification problem in a single layer neural network, we can set y = (S, x), where S is the label and x is an N dimensional vector of input features and $\theta = (\theta_1, ..., \theta_N)$, where θ is an N dimensional vector of network weights in popular neural network models.

Such an example, given by Opper (3), is given below. It should be noted that $\theta \cdot x = \Sigma_{i=1}^{N} \theta_i x_i$ and represents the inner product of the weights on all inputs. In conjunction with that representation, f(x) is the density of all inputs and $\phi(h)$ is usually a smooth sigmoid function.

$$P(y|\theta) = \phi(S\theta \cdot x) f(x)$$

To further explain the applications to neural networks, Opper contextualizes these concepts with an explanation of a regression problem. According to Opper, we can set y = (z,x), where z is the function value and x is the set of inputs. Assuming a Gaussian noise model, we'd then have the following learning model:

$$P(y|\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(z-r(\theta,x))^2} f(x)$$

where $r(\theta, x)$ is the regression function.

5 Limitations

Since the days of ENIAC, computing power has grown almost immeasurably. As the landscape of computing is rapidly changing, most research leaves questions that have yet to be answered. In his paper, Opper acknowledges many of the shortcomings of the Bayesian Statistical Inference framework.

For example, there are still questions surrounding the equivalence in asymptotic speed between online and batch algorithms. It was shown before that some cases of online algorithms can learn with the same asymptotic speed as optimized batch algorithms. However, no generalization could be made because a general learning algorithm approach is not yet clear (2). This is because:

- For order parameters to be introduced to the algorithm, very specific assumptions about the probability distributions of the inputs have to made to made. This is an issue because this information is usually not available.
- The optimization requires some knowledge about the learning model. This is an issue because the learning model has yet to have been learned, making this information seldom available to be used.

Some of these untouched areas of this paper include its applications to multilayer neural networks and finding better non-asymptotic estimates of the performance of the algorithm we described in this paper.

Another setback is in the use of the Gaussian asymptotics. Opper reveals that the Gaussian asymptotic bound of the posterior yield consistent approximations for large samples only for smooth models. Opper doesn't indicate whether the approximation remains consistent given samples without smooth models. Furthermore, when Opper is considering the applications of this Bayesian Statistical framework, he notes that "for most models, it will not be easy to perform the Gaussian averages ... exactly in order to implement the algorithm" (12). Opper mentions that these Gaussian averages can be computed analytically only for linear models and mixtures of Gaussians. This implies that the Gaussian averages are only efficiently computable for few models, and as Opper himself states, "further approximations may be necessary" (12).

6 Future Research

Noting the need for additional useful approximations for Gaussian averages, Opper suggests the "stochastic version of the algorithm based on Monte Carlo sampling" (15). Opper also recommends that "mean field methods may be useful" (15). Opper notes that much of the future of OLMB algorithms depend on theoretical asymptotic bounds not yet understood.

It is clear that better "nonasymptotic" estimates are required to understand "global convergence properties" (15). Opper mentions how such an algorithm in the Bayesian Statistical Inference framework works for non-smooth models (as we noted in section 5), referencing the "noisefree perceptron" (15). The Statistical Bayesian framework used by Opper has its benefit in understanding how certain simple models work in the Online setting; however, more research is necessary to gain a holistic picture of how the framework treats non-ideal models.

References

[1] Avrim Blum. 1998. On-line Algorithms in Machine Learning. In Developments from a June 1996 seminar on Online algorithms: the state of the art, Amos Fiat and Gerhard J. Woeginger (Eds.). Springer-Verlag, London, UK, UK, 306-325.

[2] Wikipedia contributors, "Artificial neural network," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Artificial$_n eural_n etwork\&oldid$ = 695605493($accessed December 18, 2015$).

[3] Wikipedia contributors, "Perceptron," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Perceptron&oldid=694807123 (accessed December 18, 2015).