An Investigation on Online versus Batch Learning in Predicting User Behaviour

Nikolay Burlutskiy, Miltos Petridis, Andrew Fish, Alexey Chernov, and Nour Ali

Abstract An investigation on how to produce a fast and accurate prediction of user behaviour on the Web is conducted. First, the problem of predicting user behaviour as a classification task is formulated and then the main problems of such real-time predictions are specified: the accuracy and time complexity of the prediction. Second, a method for comparison of online and batch (offline) algorithms used for user behaviour prediction is proposed. Last, the performance of these algorithms using the data from a popular question and answer platform, *Stack Overflow*, is empirically explored. It is demonstrated that a simple online learning algorithm outperforms state-of-the-art batch algorithms and performs as well as a deep learning algorithm, Deep Belief Networks. The proposed method for comparison of online and offline algorithms as well as the provided experimental evidence can be used for choosing a machine learning set-up for predicting user behaviour on the Web in scenarios where the accuracy and the time performance are of main concern.

1 Introduction

The era of the Internet and Big Data has provided us with access to a tremendous amount of digital data generated as a result of user activities on the Web. Recently, many researchers have achieved promising results in using Machine Learning (ML) for predicting user activity at home [20, 9], in online forums [6], social media [24], and customer preference [26].

Ideally, a prediction provides accurate results delivered in real time. However, in the real world predictive models cannot provide 100% accuracy as well as instantaneous predictions. Also, a complex model which provides accurate predictions can be unacceptably slow [17]. Thus, there is a need to find a ML set-up for a trade-off

N. Burlutskiy (✉)
The University of Brighton, Brighton, United Kingdom
e-mail: {N.Burlutskiy,M.Petridis,Andrew.Fish,A.Chernov,N.Ali2}@brighton.ac.uk

in the accuracy and the speed of a prediction, especially in the cases when the speed of the prediction is crucial.

There are two modes of learning commonly used in machine learning, *online* and *offline/batch* learning. In most scenarios, online algorithms are computationally much faster and more space efficient [15]. However, there is no guarantee that online learning provides results as accurate as batch learning results [17, 9]. Thus, choosing the learning mode is a challenge.

Choosing a ML algorithm for both batch and online learning modes is a challenge as well. Recently deep learning showed promising results in terms of the accuracy but training such models may take significant time [9]. In contrast, simple linear classifier can provide fast but probably less accurate results [6]. The following question is addressed in this paper:

1. How do online and batch (offline) learning modes compare in terms of accuracy and time performance?

In order to answer this question, we compare five *batch* algorithms, including a deep learning algorithm, to three *online* learning algorithms in terms of their *accuracy* and *time* performance. Due to the conceptual difference in online and offline learning, this comparison is not straightforward. Thus, we propose a method for such comparison. We investigate the performance of these two learning modes and the chosen algorithms for Stack Overflow, the largest Q&A forum. We predict user behaviour, more specifically users' response time to questions at this forum.

To sum up, our intended contribution is as follows:

- To provide a method to compare online and offline learning algorithms in terms of accuracy and time performance;
- To compare the efficiency of online algorithms to offline state-of-the-art algorithms including a deep learning algorithm for user behaviour on a Q&A forum;
- To find and propose a ML set-up for a fast and accurate prediction of user behaviour on a Q&A forum.

The remainder of this paper is organised as follows. Section 2 summarises related works. In Section 3 we introduce ML options for predicting user behaviour. Then in Section 4, we describe our method for comparing the efficiency of such predictions. In Section 5 we empirically evaluate offline and online algorithms for an efficient prediction. Finally, we conclude the paper in Section 6.

2 Related Work

Predicting user behaviour has been broadly researched in the literature. This behaviour includes but is not limited to predicting one's location at a particular time [22], customer preferences [26], user activities at home [20, 9], and behaviour on social media [27, 6]. As a result, a variety of statistical models have been used for predicting user behaviour. For example, a predictive model of time-varying user

behaviour based on smoothing and trends of user activities was proposed in [21]. Generally speaking, advanced machine learning algorithms along with intelligent feature engineering demonstrates higher accuracy but the complexity of such models and features negatively affects the time performance of the models [25, 12]. Nevertheless, analysis and evaluation of the time needed for training and predicting user behaviour is often omitted [9, 21].

Predicting user behaviour on social media has included, for example, predicting the posting time of messages in Q&A forums [25, 12], churn of users [27], response time to a tweet [24]. For example, the authors in [27] focused on churn prediction in social networks where the authors proposed using a modified Logistic Regression (LR) model. A similar prediction was accomplished in [24] where a novel approach of activity prediction was proposed. The problem formulated by the authors of [24] was "given a set of tweets and a future timeframe, to extract a set of activities that will be popular during that timeframe". However, in these papers the authors did not address the problem of how to build a fast and accurate model for the prediction of user behaviour.

The authors in [20] proposed an approach to predict the start time of the next user's daily activity. The approach was based on using continuous normal distribution and outlier detection. In another paper [26], the authors showed that neural networks can provide accurate predictions of customer restaurant preference. Both works were focused on providing an accurate prediction but the complexity of the approach as well as the time required to build the model were not analysed.

At the moment of writing this paper, Deep Learning (DL) was an evolving topic attracting many ML researchers and, as a result, we found some promising results of applying DL for prediction of user behaviour. For instance, in [9] the researchers used Deep Belief Networks (DBN) for predicting user behaviour in a smart home environment. They demonstrated that DBN-based algorithms outperformed existing methods, such as a nonlinear Support Vector Machines (SVM) and k-means. A similar advantage of DBN in terms of accuracy was shown in [6] where the authors showed that a DBN-based algorithm outperformed other existing methods such as LR, Decision Trees (DT), k Nearest Neighbours (k-NN), and SVM.

Online prediction is a hot research topic due to the need of the Big Data world to provide real-time results. For example, predicting daily user activity for the next few seconds or minutes prohibits the user of batch training unless a model is very simple. The authors of the paper [9] compared the accuracy of prediction using online, mini-batch, and batch training. However, they did not evaluate the time performance of the prediction.

A comparison of training and testing times for different ML algorithms was performed in [16]. However, the authors did not compare online and offline settings.

Even though many works have shown high accuracy for different prediction tasks, the time and complexity of the trained models are usually omitted. In our paper, we evaluate both the accuracy and time efficiency of online and batch learning modes across several state-of-the-art algorithms including a deep learning algorithm. First, we introduce a method for comparison of these two modes of learning. Second, we compare five batch algorithms, LR, DT, k-NN, SVM, and DBN with

three online learning algorithms, namely Stochastic Gradient Decent (SGD), Perceptron, and Passive-Aggressive (PA) algorithms. Last, we provide a guideline how to choose ML algorithms for real-time prediction of user behaviour on the Web.

3 Machine Learning for Predicting User Behaviour

We formulate the problem of predicting user behaviour as a classification task. This task is defined by an input feature set X and a class label set Y where the goal is to match labels from the set Y to instances in the set X. The input feature set X is formed from factors potentially affecting the accuracy of predicting user behaviour which are identified in the literature [25, 2, 12, 1, 14, 6, 3]. Then the class label Y is formulated. Supervised ML algorithms can be divided into two large groups, namely *batch* or *offline* learning and *online* learning algorithms. The batch training can also be divided in full-batch learning where a model is trained over the entire training data and mini-batch learning when the model is trained and then updated after some number m of training instances. On the contrary, online learning means that a model is updated after every new instance.

3.1 Batch Prediction

A batch learning algorithm uses a training set D_{tr} to generate an output hypothesis, which is a function F that maps instances of an input set X to a label set Y. Thus, batch learners build a statistical assumption on a probability distribution over the product space $X \times Y$. The batch learning algorithm is expected to generalise, in the sense that its output hypothesis predicts the labels Y of previously unseen examples X sampled from the distribution [11].

3.2 Online Prediction

Online prediction is based on a training algorithm in which a learner operates on a sequence of data entries. At each step t, the learner receives an example $x_t \in X$ in a d-dimensional feature space, that is, $X = \mathbb{R}^d$. The learner predicts the class label $\hat{y}_t$ for each example x_t as soon as it receives it:

$$\hat{y}_t = sgn(f(x_t, w_t)) \in Y \tag{1}$$

Where $\hat{y}_t$ is the predicted class label, x_t is an example, w_t is the weight assigned to the example, $sgn()$ is the sign function returning $\{0,1\}$, f is a function mapping x_t, w_t into a real number $r \in \mathbb{R}$, and Y is a class label. Then the true label $y_t \in Y$

is revealed which allows the calculation of the loss $l(x_t, y_t, w_t)$ which reflects the difference between the learner's prediction and the revealed true label y_t. The loss is used for updating the classification model at the end of each learning step.

A generalised online learning algorithm is shown in Algorithm 1 which was described in [13]. This algorithm can be instantiated by substituting the prediction function f, loss function l, and update function Δ.

Online algorithms process data sample by sample which is natural for predicting

Algorithm 1 Online Learning

1: Initialise: $w_1 = 0$
2: **for** t=1,2,..,T **do**
3: The learner receives an incoming instance: $x_t \in X$;
4: The learner predicts the class label: $\hat{y}_t = sgn(f(x_t, w_t))$;
5: The true class label is revealed from the environment: $y_t \in Y$;
6: The learner calculates the suffered loss: $l(w_t, (x_t, y_t))$;
7: **if** $l(w_t, (x_t, y_t)) > 0$ **then**
8: The learner updates the classification model: $w_{t+1} \leftarrow w_t + \Delta(w_t, (x_t, y_t))$;

user behaviour since the records of user activities are often in a chronological order. In this paper. we chose Stochastic Gradient Descent (SGD), Perceptron, and Passive-Aggressive (PA) as three of the most popular instances of the algorithm [13].

- *Stochastic Gradient Descent*: the function f in Algorithm 1 for this learner is the dot product: $f(x_t, w_t) = w_t \cdot x_t$ and the function Δ is calculated as follows: $\Delta = \eta(y_t - \hat{y_t})x_t$ where η is the learning rate. For simplicity, we chose hinge loss as the loss function: $l(y_t) = \max(0, 1 - y_t(w_t \cdot x_t))$.
- *Perceptron*: the difference to SGD is in the way the learner's loss is calculated: $l(y_t) = \max(0, -y_t(w_t \cdot x_t))$.
- *Passive-Aggressive*: the family of these algorithms includes a regularisation parameter C. The parameter C is a positive parameter which controls the influence of the slack term on the objective function. The update function is: $\Delta = \tau y_t x_t$ where $\tau = min(C, \frac{l_t}{||x_t||^2})$ and the loss function l is hinge loss. It was demonstrated that larger values of C imply a more aggressive update step [10].

3.3 Time Complexity of ML Algorithms

The complexity of ML algorithms for both training and testing times varies significantly across different families of algorithms and depends on the number of samples, number of features, and algorithm parameters. For some algorithms, a formal time complexity analysis has been performed [23, 8, 4, 18]. This analysis can be used in choosing an algorithm for a prediction. For example, if time performance is crucial then it might be advantageous to choose an algorithm with lesser time complexity.

In this paper, we decided to choose algorithms with different time complexity. As a result, we chose five algorithms, namely a Decision Tree Algorithm (DT), Logistic Regression (LR), Support Vector Machines (SVM), k Nearest Neighbors (k-NN), and Deep Belief Networks (DBN), and three online algorithms, namely Stochastic Gradient Descent (SGD), Perceptron, and Passive-Aggressive (PA). The time complexity of training these models is in Table 1.

For the DT learning algorithm C4.5, the time complexity is $O(mn^2)$ where m is the number of features and n is the number of samples [23]. The time complexity of Logistic Regression (LR) is $O(mn)$ but might be worse depends on the implementation of the optimisation method [18]. For SVM, the time complexity, again, depends on the optimisation method but for one of the most popular implementations, in LibSVM, the time complexity is $O(n^3)$ [8]. For a k-NN learner implemented as a kd-tree, the training cost is $O(n\log(n))$ in time and the predicting complexity is $O(k\log(n))$, where k is the number of nearest neighbors and n is the number of instances. For more complicated models, such as Deep Belief Networks (DBNs), a formal complexity analysis is hard since there is no measure to evaluate the complexity of functions implemented by deep networks [4].

The training cost for all linear online learners is $O(kn\bar{p})$, where n is the number

Table 1 Time complexity of the algorithms.

Algorithm	Implementation	Complexity	Ref.
DT	C4.5	$O(mn^2)$	[23]
SVM	LibSVM	$O(n^3)$	[8]
k-NN	kd-tree	$O(k\log(n))$	–
LR	LM BFGS	$O(mn)$	[18]
DBN	m hidden layers	high	[4]
SGD, P, PA	Algorithm 1	$O(kn\bar{p})$	[5]

of training samples, k is the number of iterations (epochs), $\bar{p}$ is the average number of non-zero attributes per sample [5]. Although kernel-based online learners potentially can achieve better accuracy compared to linear online learners, kernel-based online learners require more memory to store the data, as well as more computational effort, which makes them unsuitable for large-scale applications. Thus, we considered only linear online learners in this paper.

4 The Method to Compare Online and Offline Learning Modes

Due to the conceptual difference in online and offline learning, it is hard to make a comparison between them. Thus, we propose a method for comparing the accuracy and the time performance of training and testing the learners. The standard performance measurements for offline learning are accuracy, precision, recall, and F1-measure [19].

The performance of online learners is usually measured by the cumulative loss a

Fig. 1 Proposed training and testing scheme for comparison of online and offline learning algorithms.

learner suffers while observing a sequence of training samples. In order to compare online and offline learning algorithms, we introduced a method which is a modification of mini-batch training (see Figure 1).

First, we used a classical 5-fold validation for training and testing the models.

Algorithm 2 Calculating the accuracy of learners

1: Initialise training and testing times: $T_{tr} = 0, T_{tt} = 0$
2: Sort the data D chronologically from the oldest to the latest;
3: Divide the data D into m equal batches D_i
4: **for** i=1,2,..,m **do**
5: Divide each batch D_i into a training set D_{itr} and test set D_{itt};
6: Train a learner l_i on D_{itr};
7: Record the time taken for the training T_{itr};
8: Update the training time $T_{tr} = T_{tr} + T_{itr}$;
9: Test l_i on D_{itr};
10: Record the time taken for the testing T_{itt};
11: Update the testing time $T_{tt} = T_{tt} + T_{itt}$;
12: Calculate the average accuracy A_i over D_{itr};
13: Calculate the average accuracy A_{av} over all A_i

We divided the whole set into five parts (Figure 1), trained a model on any four parts and then tested on the fifth part. Then we changed the parts for training and testing and repeated the process until we eventually used all the five possible combinations of the parts for training and testing. Then we calculated the average accuracy A_{av} and the time performance for training T_{tr} and testing T_{tt} for these five tests. We repeated the same process for both online and offline algorithms.

In the second set-up, we applied Algorithm 2 to online learners l_{on} and then we repeated the same algorithm for offline learners l_{off}. The purpose of applying the same aforementioned algorithm to both online and offline learners was to compare

the accuracy A_{av} and the performance of online and offline algorithms in terms of the training time T_{tr} and testing time T_{tt} on the same data D in the same set-up.

5 Empirical Study

In order to compare offline to online learning, and to evaluate the performance of a deep learning algorithm, DBN, we conducted an empirical study to predict user behaviour in a Q&A forum. Q&A forums are the platforms where users ask and answer questions. We chose *Stack Overflow* website for the experiments since it is one of the most popular and fastest growing Q&A platforms (see Table 2).

We formulated the prediction task as a binary classification task – we predicted users' response time to asked questions [6]. *Stack Overflow* allows an asker to accept a satisfactory answer explicitly by clicking a button 'accept an answer'. In our experiments, we used this information to predict users' response time - the time when the first accepted answer for a question will be received. We created the label for prediction by assigning '1' to response times $t < T_m$ and '0' to response times $t \geq T_m$ where T_m = 26 minutes was the median time. Since we chose the median time for binarisation, the class for prediction was balanced.

The raw data dump (September 2014 data dump) of *Stack Overflow* consists of eight xml files: Users, Posts, Badges, Posts History, Post Links, Comments, Tags, and Votes (∼96.6 GB). We imported Users, Posts, Badges, Comments, Tags, and Votes into an sqlite database (∼41.3 GB). Then we extracted features influencing user behaviour on the Web. These features potentially affecting the accuracy of predicting user behaviour were identified from the literature [25, 2, 12, 1, 14, 6, 3]. As a result, we extracted 32 features (see Table 3) for 1,537,036 samples (∼1.5 GB).

System set-up

To run the experiment, we designed and implemented a software system for storing, retrieving, visualisation, statistical analysis, and prediction using machine learning algorithms [1]. We built the system using a Python library *scikit-learn* for machine learning [2]. For Deep Belief Networks, we used *Theano Lasagne* library [3]. For calculating the text-related features, *NLTK* library was deployed [4]. As a machine for processing this data (∼96.6 GB), a computer with 8 GB RAM and 4 CPUs was used.

[1] The source code is at *https://github.com/Nik0l/UTemPr*

[2] http://scikit-learn.org/

[3] https://github.com/Lasagne/Lasagne

[4] http://www.nltk.org/

Table 2 The statistics on the *Stack Overflow* dataset used in the experiments.

Forum	Users	Questions		Question Response Time						
		Total	Answered	Temporal statistics				Answered within		
				Mean, days	Med, min	Min, min	Max, days	1 hour	1 day	1 month
Stack Overflow	3,472,204	7,990,488	4,596,829	5.7	26	0.20	2,087	61	84	96

5.1 Experiments

In our experiment, we trained and tested three online (SGD, Perceptron, PA) and five offline (LR, *k*-NN, DT, SVM, DBN) algorithms on the same dataset with $1,537,036$ samples of 32 features (see Table 3). We scaled the data to $[0,1]$ since the SGD algorithm is sensitive to feature scaling. We did not shuffle the training data since the records of user activities were recorded in a chronological order and we wanted to preserve that order.

Online algorithms: we chose the number of iterations equal to one since it was enough for the convergence of SGD. We did not use any regularisation methods [5]. The learning rate η was set to be equal '1'.

Batch algorithms: DBN was trained with 10 epochs, the hidden layer was represented as a Restricted Boltzmann Machine (RBM) with 300 nodes. We chose k=3 nearest neigbours for the k-NN learner.

In the first experiment we use the proposed method for comparison of online and offline learning modes. This set-up has mini-batchs with 5000 samples each (see Table 4 columns'1' and '2' correspondingly). In the second experiment we use a classic 5-fold validation. Since there is variance in prediction models trained on the same data, each algorithm was executed ten times and then we calculated the average of these ten independent runs. The results of these runs for the two experiments are shown in Table 4.

5.2 Results and Discussions

The results for the prediction of users' response time are in Table 4. DBN slightly outperformed other machine learning algorithms in all set-ups by 3% - 9% in terms of the accuracy. There was no significant difference in the accuracy of the batch learning algorithms compared to online learning algorithms. However, the time spent on training and testing the batch models was significant especially compared to DBN and SVM.

The fastest batch learning algorithm, LR, showed almost 3% poorer accuracy and was 13 times slower in terms of the training time. SGD showed the highest accu-

[5] In our experiments, we tried L_1 and L_2 regularisation but we did not find any significant improvements in the results compared to the results without regularisation reported in this paper.

Table 3 Features for the Prediction of Users' Response Time.

F_i	**Description**	**Comments**	**Ref.**
F_1	The total number of question asked by a user	This number is calculated from the time a user registered at a website	[25, 2, 14]
F_2	The total number of answers given by a user	This number is calculated from the time a user registered at a website	[25, 2, 14]
F_3	The total number of user's profile views by other users	Some websites provide with the information on how many times user's profile has been viewed	[1]
F_4	The number of user's answers accepted by other users	*Stack Exchange* websites provide a functionality for an asker to accept an answer	[6]
F_5	The reputation of the user	*Stack Exchange* provides with user's reputation.	[1, 14]
F_6	The number of user's upvotes	Some websites provide this feature	[2, 14]
F_7	The number of user's downvotes	Some websites provide this feature	[2, 14]
F_8	The location of a user	For example, *USA, FL, Miami*	[6]
F_9	User's latitude	For example, 101.67	[6]
F_{10}	User's longitude	For example, 20.27	[6]
F_{11}	User's time zone	For example, UTC+2	[6]
F_{12}	The number of days a user has been registered	$T_{now} - T_{registered}(u_i)$ where T_{now} is the current time and $T_{registered}$ is the time when a user u_i registered	[14]
F_{13}	The average answering time	The time is calculated for all questions	[6]
F_{14}	12 entries vector: the number of posts for each month in last year	$(p_1(u_i), ..., p_{12}(u_i))$ where p_i is the number of posts by a user u_i in a particular month j	[6]
F_{15}	24 entries vector: the average number of posts for each hour	$(p_1(u_i), ..., p_{24}(u_i))$, where p_j is the average number of posts by a user u_i at a particular hour j	[25, 12]
F_{16}	The number of questions asked by a user during a time interval	In this paper, we used the number of questions asked by a user during the last week	[7]
F_{17}	The number of answers given by a user during a time interval	In this paper, we used the number of answers given by a user in the last week	[7]
F_{18}	The length of the question title	We ignored spaces in the titles	[2, 3]
F_{19}	The length of the question body	We ignored spaces in the text	[2, 3]
F_{20}	# of 'wh' words in the title	'Wh' words are question words, i. e. 'who','what'	[12, 3]
F_{21}	# of 'wh' words in the body	Wh' words are question words, i. e. 'who','what'	[12, 3]
F_{22}	# of active verbs in the title	Active verbs include such verbs as 'do', 'make'	[3]
F_{23}	# of active verbs in the body	Active verbs include such verbs as 'do', 'make'	[3]
F_{24}	# of times a user mentioned himself, (for ex., 'we', 'I', 'me')	If a user mentioned themselves in a question means they tried to solve the problem	[3]
F_{25}	The number of url links	For example, the number *href* words	[12]
F_{26}	The number of images	For example, the number of *img* words in the text	[3]
F_{27}	The total number of times a question was viewed	A question can be viewed by both registered and not registered users	[6]
F_{28}	The number of tags in a question	Usually questions are tagged with some words, for example, 'computers' or 'linux'.	[25, 14]
F_{29}	The popularity of the tags	Frequency of the tags in the questions	[3]
F_{30}	The number of popular tags	The tags are divided into popular and non-popular	[3]
F_{31}	The 'togetherness' of tags a question is tagged with	$\frac{p(x,y)}{p(x)p(y)}$, $p(x,y)$ is the probability of tags x and y occur together, whereas $p(x)$ and $p(y)$ - independently	[2, 3]
F_{32}	The number of comments for a question during a time interval	In this paper, we used the number of comments and replies for a question during the last day	[6]

racy of 63.8% out of the online learners (see Table 4). On contrary, a Perceptron learner showed relatively poor performance with the accuracy of 57.7% probably due to the fact that it is sensitive to badly labelled examples even after training on many examples whereas the SGD and Passive-Aggressive algorithms are more robust to badly labelled examples. Also, the aspect that the used online algorithms put different importance on each training instance over time might have influenced the results. Also, the online learners had less variation in the accuracy over the process of training the models compared to the batch learners.

The time performance of batch algorithms corresponds to their time complexity evaluated in Section 3.3 earlier. We can notice that the time complexity of DBN is similar to the time complexity of SVM since both DBN and SVM performed the worst and relatively similar in terms of the training time.

The time performance of online algorithms in the conducted experiments corresponds to their time complexity evaluated in Section 3.3 as well. However, training a Perceptron algorithm took slightly longer than the other algorithms.

We expected online learning to perform faster than batch learning in both training and the prediction. The reason is in the nature of these algorithms - in online learning the weights of the model are updated at each step compared to the computationally expensive training in the batch mode. However, we did not expect such significant differences. Even though the experimental results are very hardware dependent, in our experiments the online training and predicting took 13-3000 times less time than batch training and predicting.

It is important to mention that most learners, both online and batch except DT, SVM, and LR, showed poorer accuracy in the results for 5-fold validation (see Table 4) whereas DT, SVM, and LR showed almost the same accuracy. One possible cause of such drop in the accuracy can be the sequential nature of user behaviour and, as a result, the models must be trained in a sequential order rather than shuffling parts of data as it is done in 5-fold validation.

Table 4 The results for online and offline learning in comparison (1,537,036 samples) for 5-fold validation (5 fold) any mini-batches of 5,000 samples (batch).

Algorithm	**Accuracy, %**		**Training Time, s**		**Testing Time, s**	
	5 fold	**batch**	**5 fold**	**batch**	**5 fold**	**batch**
k-NN	57.4 ± 0.72	57.7 ± 0.72	819.7	98.2	2813	944.3
DT	60.0 ± 0.20	59.2 ± 0.43	193.8	60.8	0.568	0.993
LR	63.7 ± 0.35	61.1 ± 0.69	**85.3**	**18.1**	**0.068**	**0.356**
SVM	62.4 ± 0.46	61.9 ± 0.46	2,933.3	2,333.3	380.6	360.6
DBN	**64.5 ± 0.82**	**66.5 ± 0.52**	2,812.2	3,009.2	705.3	725.7
Perceptron	55.6 ± 0.61	57.7 ± 0.61	4.8	1.4	0.066	**0.002**
PA	59.6 ± 0.52	63.0 ± 0.52	**4.2**	**0.7**	**0.052**	0.003
SGD	**61.2 ± 0.42**	**63.8 ± 0.82**	4.7	1.3	0.056	**0.002**

6 Conclusions

We investigated the problem of comparing online and batch ML set-ups in the context of user behaviour on the Web. This comparison allows to choose a ML set-up which provides a fast and accurate prediction. Predicting user behaviour on the Web often requires fast predictions performed in units of seconds or minutes. Thus, as a case study, we predicted the time when a user will answer a question on the largest technical Q&A forum, *Stack Overflow*. Our experiment showed that even though a deep learning algorithm demonstrated slightly more accurate results, the time for both training and predicting was several magnitudes higher compared to the simplest online learning algorithms. Thus, in the world of Big Data, online learning can serve as a simple solution for providing a fast, near real-time prediction. Even though we conducted the experiment only for *Stack Overflow*, we are planning to explore more datasets and prediction tasks for improving the generalisation of our findings beyond the context of user behaviour on the Web. In the future, these findings will be useful for choosing a ML setup for predictions when there are tough requirements on computational complexity as well as the accuracy of a real time prediction system.

Acknowledgments The authors are grateful for illuminating discussions to Dr Yuri Kalnishkan's team in the project "On-line Self-Tuning Learning Algorithms for Handling Historical Information" (funded by the Leverhulme Trust).

References

1. Anderson, A., Huttenlocher, D., Kleinberg, J., and Leskovec, J.. Discovering Value from Community Activity on Focused Question Answering Sites: A Case Study of Stack Overflow. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 850–858, New York, USA, 2012. ACM.
2. Asaduzzaman, M., Mashiyat, A. S., Roy, C. K., and Schneider, K. A.. Answering Questions about Unanswered Questions of Stack Overflow. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 97–100, Piscataway, NJ, USA, 2013.
3. Bhat, V., Gokhale, A., Jadhav, R., Pudipeddi, J., and Akoglu, L.. Min(e)d your Tags: Analysis of Question Response Time in Stack Overflow. In *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*, pages 328–335, 2014.
4. Bianchini, M., and Scarselli, F.. On the Complexity of Neural Network Classifiers: A Comparison between Shallow and Deep Achitectures. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(8):1553–1565, Aug 2014.
5. Bottou, L.. *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, chapter Large-Scale Machine Learning with Stochastic Gradient Descent, pages 177–186. Physica-Verlag HD, Heidelberg, 2010.
6. Burlutskiy, N., Petridis, M., Fish, A., and Ali, N.. Prediction of Users' Response Time in Q&A Communities. In *ICMLA'15, International Conference on Machine Learning and Applications*, 2015.
7. Cai, Y., and Chakravarthy, S.. Answer Quality Prediction in Q&A Social Networks by Leveraging Temporal Features. *IJNGC*, 4(1), 2013.

8. Chapelle, O.. Training a Support Vector Machine in the Primal. *Neural Comput.*, 19(5):1155–1178, May 2007.
9. Choi, S., Kim, E., and Oh, S.. Human Behavior Prediction for Smart Homes using Deep Learning. In *RO-MAN, 2013 IEEE*, pages 173–179, Aug 2013.
10. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y.. Online Passive-Aggressive Algorithms. *J. Mach. Learn. Res.*, 7:551–585, Dec. 2006.
11. Dekel, O.. From Online to Batch Learning with Cutoff-Averaging. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 377–384. Curran Associates, Inc., 2009.
12. Dror, G., Maarek, Y., and Szpektor, I.. Will my Question be Answered? Predicting "Question Answerability" in Community Question-Answering Sites. In Blockeel, H., editor, *Machine Learning and Knowledge Discovery in Databases*, volume 8190 of *Lecture Notes in Computer Science*, pages 499–514. Springer Berlin Heidelberg, 2013.
13. Hoi, S. C., Wang, J., and Zhao, P.. Libol: A Library for Online Learning Algorithms. *Journal of Machine Learning Research*, 15:495–499, 2014.
14. Lezina, C. G. E., and Kuznetsov, A. M.. Predict Closed Questions on Stack Overflow, 2012.
15. Liang, N. Y., Huang, G. B., Saratchandran, P., and Sundararajan, N.. A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks. *Neural Networks, IEEE Transactions on*, 17(6):1411–1423, Nov 2006.
16. Lim, T. S., Loh, W. Y., and Shih, Y. S.. A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Three Old and New Classification Algorithms. *Machine Learning*, 40(3):203–228, 2000.
17. Loumiotis, I., Adamopoulou, E., Demestichas, K., and Theologou, M.. On Trade-off between Computational Efficiency and Prediction Accuracy in Bandwidth Traffic Estimation. *ELECTRONICS LETTERS*, 50(10):754–756, 2014.
18. Minka, T. P.. A Comparison of Numerical Optimizers for Logistic Regression. Technical report, 2003.
19. Mohri, M., Rostamizadeh, A., and Talwalkar, A.. *Foundations of Machine Learning*. The MIT Press, 2012.
20. Nazerfard E., and Cook, D.. Using Bayesian Networks for Daily Activity Prediction, 2013.
21. Radinsky, K., Svore, K., Dumais, S., Teevan, J., Bocharov, A., and Horvitz, E.. Modeling and Predicting Behavioral Dynamics on the Web. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 599–608, New York, NY, USA, 2012. ACM.
22. Sadilek, A., and Krumm, J.. Far out: Predicting Long-Term Human Mobility. In *AAAI*, 2012.
23. Su, J., and Zhang, H.. A Fast Decision Tree Learning Algorithm. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 500–505. AAAI Press, 2006.
24. Weerkamp, W., and De Rijke, M.. Activity Prediction: A Twitter-based Exploration. In *Proceedings of TAIA'12*, Aug. 2012.
25. Yang, L., Bao, S., Lin, Q., Wu, X., Han, D., Su, Z., and Yu, Y.. Analyzing and Predicting Not-Answered Questions in Community-Based Question Answering Services. In Burgard, W., editor, *AAAI*. AAAI Press, 2011.
26. Zheng, B., Thompson, K., Lam, S. S., Yoon, S. W., and Gnanasambandam, N.. Customers Behavior Prediction using Artificial Neural Network. In *Industrial and Systems Engineering Research Conference (ISERC)*, pages 700–709. Institute of Industrial Engineerings, 2013.
27. Zhu, Y., Zhong, E., Pan, S. J., Wang, X., Zhou, M., and Yang, Q.. Predicting User Activity Level in Social Networks. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 159–168, New York, NY, USA, 2013. ACM.