Predicting Online Community Churners using Gaussian Sequences

Matthew Rowe

School of Computing and Communications, Lancaster University, Lancaster, UK
m.rowe@lancaster.ac.uk

Abstract. Knowing which users are likely to churn (i.e. leave) a service enables service providers to offer retention incentives for users to remain. To date, the prediction of churners has been largely performed through the examination of users' social network features; in order to see how churners and non-churners differ. In this paper we examine the social and lexical development of churners and non-churners and find that they exhibit visibly different signals over time. We present a prediction model that mines such development signals using Gaussian Sequences in the form of a joint probability model; under the assumption that the values of churners' and non-churners' social and lexical signals are normally distributed at a given time point. The evaluation of our approach, and its different permutations, demonstrates that we achieve significantly better performance than state of the art baselines for two of the datasets that we tested the approach on.

1 Introduction

The churn (leaving) of a user from a service represents a loss to the service owner, be it: a telecommunications operator, where a customer leaving represents a loss of financial income; a question-answering platform, where an expert leaving could lead to a reduction of *know-how* in the community, or: a discussion forum, in which a user leaving could result in the forum's social capital, and perhaps *vibrancy*, being reduced. Therefore, predicting which users will leave a given service is important to a range of service providers; and an effective means of doing so enables retention strategies to be applied to those potential churners.

The majority of work within the area of churn prediction has focussed on building a prediction model using information about a user's social network position [2], and thus the extent to which he is interacting with other users, and/or the activity of a given user up until a given point in time. Our prior work [8] proposed an approach based upon the *lifecycle* of a user (i.e. the period of time between a user joining a service to either churning or remaining) in which social and lexical dynamics of the user were mined and a model fitted to the development curve of the user; properties of those models were then used as features for prediction models to differentiate between churners and non-churners. However, this approach was limited by only concentrating on a fixed number of lifecycle stages (e.g. 20) and did not examine how churners

and non-churners developed differently. In this paper we attempt to fill these gaps by exploring the following two questions: (i) *How do churners and non-churner develop?* And; (ii) *How can we exploit development information to detect churners?*

In exploring these questions we found that churners and non-churner do indeed differ in how they develop, both socially and lexically, over their lifetimes, and that by assuming a Gaussian distribution at each lifecycle stage then we can chain together *Gaussian Sequences* for use in prediction. This paper makes the following contributions:

- Examination of different lifecycle patterns for both churners and non-churners across three online community datasets; with different lifecycle fidelity settings (5, 10 and 20 stages).
- Prediction models based on Gaussian sequences with slack variables for tuning, and a new model learning approach called *Dual-Stochastic Gradient Descent*.
- Evaluation of the proposed models against state of the art baselines showing significantly better performance for two of the tested datasets.

We begin this paper with a review of existing churn prediction approaches before then moving on to detail the datasets used for our work and how we label both churners and non-churners within them.

2 Related Work

Churner prediction has been studied across a number of domains, for instance Zhang et al. [9] predicted churners in a Chinese telecommunications network by inducing a decision tree classifier from user activity features (e.g. call duration) and network properties (e.g. 2nd order ego-network clustering coefficient). McGowan et al. [6] also predicted churners from a telecommunications provider by experimenting with different dimensionality reduction and boosting methods. Lewis et al. [5] examined Facebook networks of college students over a 4 year period and found an association between friendship maintenance and geographical proximity and shared tastes. Quercia et al. [7] analysed Facebook friendships and users' personality traits, finding that churn was likely to happen if the ages of connected users differed and if one of the users was neurotic or introverted. Kwak et al. have examined churners from Twitter networks in [3] and [4]: in the former the authors analysed the differences between social network snapshots, separated by 6 weeks, of Korean Twitter users finding that users unfollowed other users when the users talked about uninteresting topics; while in the latter work [4], the authors induced a logistic regression model to predict churners based on pairwise features (formed between the user and each of his subscribers).

Similar to our work, the work by Karnstedt et al. in [1] and [2] examined the prediction of churners from the Irish online community platform Boards.ie, finding that the probability of a user churning was related to the number of prior users with whom the individual has communicated having churned before. The authors examined the social network properties of churners against non-churners

(i.e. in-degree, out-degree, clustering coefficient, closeness centrality, etc.), inducing a J48 decision tree to differentiate between churners and non-churners when using social network properties formed from the reply-to graph of the online communities. In this paper, we implement this model as our baseline by engineering the same features using the same experimental setup. Our approach differs from existing work by assessing churners' and non-churners' development signals, and inducing a joint-probability function from such information.

Table 1. Splits of users within the datasets and the churn window duration

Platform	#Churners	#Non-churners	Churn Window
Facebook	1,033	1,199	[04-11-2011, 28-08-2012]
SAP	10,421	7,255	[29-11-2009,07-09-2010]
Server Fault	12,314	11,144	[13-06-2010,24-12-2010]
Boards.ie	65,528	6,120,008	[01-01-2005,13-02-2008]

3 Datasets

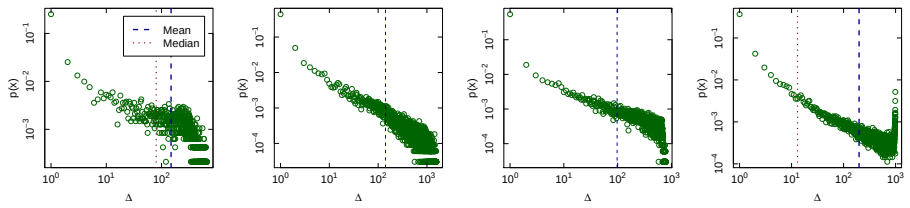
To provide a broad examination of user lifecycles across different online community platforms we used data collected from four independent platforms:

1. *Facebook:* Data was obtained from Facebook groups related to Open University degree course discussions. Although Facebook provides the ability to collect social network data for users, we did not collect such data in this instance and instead used the reply-to graph within the groups to build social networks for individual users.
2. *SAP Community Network (SAP):* The SAP Community Network is a community question answering system related to SAP technology products and information technologies. Users sign up to the platform and post questions related to technical issues, other users then provide answers to those questions and should any answers satisfy the original query, and therefore solve the issue, the answerer is awarded points.
3. *Server Fault.* Similar to SAP, Server Fault is a platform that is part of the Stack Overflow question answering site collection.[1] The platform functions in a similar vein to SAP by providing users with the means to post questions pertaining to a variety of server-related issues, and allowing other community members to reply with potential answers.
4. *Boards.ie* This platform is a community message board that provides a range of dedicated forums, where each forum is used to discuss a given topics (e.g. Rugby Union, Xbox360 games, etc.). We were provided with data covering the period 1998-2008 and, like SAP and ServerFault, we also had access to the reply-to graph in each forum.

[1] `http://stackoverflow.com/`

Unlike on subscription-based services where a *churner* is identifiable by the cancellation of the service (e.g. cancelling a contract), on online community platforms we do not have such information from which to label churners and non-churners. Instead, we examined users' activity and then decided on a suitable *inactivity* threshold where, should a user remain inactive for more than that period (i.e. x days), then we can say he has *churned*. To derive this threshold, we first defined Δ as the maximum number of days between posts across the platforms' datasets for each user, and then plotted the relative frequency distribution of Δ across the platforms in Figure 1. We then selected each distribution's mean as the *churn control window* size: 149 days, 141 days, 97 days and 198 days for Facebook, SAP, ServerFault and Boards.ie respectively.

To derive churners and non-churners we took the final post date in a given dataset and went back n (size of the churn control window) days, this date gives the *churn window end point* (t''). We then went back a further $2n$ days to give the *churn window start point* (t'); thus the *churn window* is defined as a closed date interval $[t', t'']$. Users who posted for the final time in $[t', t'']$ were defined as *churners* and users who posted after $[t', t'']$ were labeled as a *non-churners*. Table 1 shows the number of churners and non-churners derived using this approach. We split each platform's users up into a *training* and *test* set using an 80:20% split respectively - using the former set to inspect how users develop and evolve and the latter set (test) to detect churners. All analysis that follows and the features engineered for our experiments use data from *before* the churn window, thereby not biasing our prediction experiments and reflecting a real-world churn prediction setting where we only have information up until a given time point.



(a) Facebook (b) SAP (c) Server Fault (d) Boards.ie

Fig. 1. Gap distributions across users of the different platforms. The mean and median of the distributions are shown using blue dashed and red dotted lines respectively.

4 User Lifecycles

In this section we briefly describe our approach for representing the lifecycles of users on the online community platforms - for a more comprehensive description we refer the reader to our prior work [8]. We begin by segmenting a user's

lifecycle into k stages, where each stage contains the same number of posts. The setting of k controls the *fidelity* of a user's lifecycle and in this paper we experiment with various settings where $k = \{5, 10, 20\}$. For each lifecycle stage (i.e. $s \in S = \{1, 2, \ldots, k\}$) we wish to inspect the social and lexical dynamics of the user, as follows:

4.1 Social Dynamics

For examining the *social dynamics* of each user we looked at the distribution of his *in-degree* and *out-degree* - i.e. the number of edges that connect to a given user and the the number of edges from the user. As we are dealing with conversation-based platforms for our experiments we can use the *reply-to* graph to construct these edges, where we define an edge connecting to a given user u if another user v has replied to him. Given our use of lifecycle periods we use the discrete time intervals that constitute $s \in S$ to derive the set of users who replied to u, defining this set as $\Gamma_s^{IN} = \{author(q) : p \in P_u, q \in P, time(q) \in s, q \rightarrow p\}$.[2] We also define the set of users that u has replied to within a given time interval using Γ_s^{OUT} with the reply direction reversed. From these definitions we can then form a discrete probability distribution that captures the distribution of repliers to user u, using Γ_s^{IN}, and user u responding to community users using Γ_s^{OUT}. For an arbitrary user ($v \in \Gamma_s^{IN}$) who has contacted user u within lifecycle stage s we define this probability of interaction as follows:

$$Pr(v \mid \Gamma_s^{IN}) = \frac{|\{q : p \in P_u, q \in P_v, time(q) \in s, q \rightarrow p\}|}{\sum_{x \in \Gamma_s^{IN}} |\{q : p \in P_u, q \in P_x, time(q) \in s, q \rightarrow p\}|}$$

Given this formulation we now have time-dependent discrete probability distributions for a given user's in-degree and out-degree distributions, thereby allowing the *social* changes of users to be analysed in terms of the users communicating with a given user over time.

4.2 Lexical Dynamics

We modelled the *lexical dynamics* of users based on their term usage over time. We first retrieved all posts made by a given user within a lifecycle period and then removed stop words and filtered out any punctuation. We defined a multiset of the set of terms used by u in a given time period: $t \in C_s$ and a mapping function $g : C_s \rightarrow \mathbb{N}$ that returns the multiplicity of a given term's usage by the user at a given time period. We then defined the discrete conditional probability distribution for a given user u and lifecyle stage s based on the relative frequency distribution of terms used by u in that lifecycle period.

[2] We use $p \rightarrow q$ to denote message q replying to message p, P_u to denote posts authored by u, P to denote all posts.

4.3 Modelling User Evolution

Given the use of discrete probability distributions derived for each dynamic (e.g. in-degree) and lifecycle stage (s) we can gauge the changes that each user goes through by assessing for changes in this distribution. To this end we assess: (i) period variation, using entropy; (ii) historical contrasts to assess how the user is diverging from prior dynamics, using cross-entropy measured between a given stage's distribution and all prior stage distributions and then taking the minimum, and; (iii) community contrasts to assess how the user is diverging from the general behaviour of the community, using cross-entropy also.

Period Variation. For each platform (Facebook, SAP, and ServerFault) we derived the entropy of each user in each of his individual lifecycle periods based on the in-degree, out-degree and term distributions; we then recorded the mean of these entropy values over each lifecycle period for churners and non-churners. Figure 2 shows the differences in the development signals between churners and non-churners for ServerFault users,[3] where, although the development signals remain relatively level across the lifecycle stages, there are clear differences in the magnitude of the entropy values - in particular for lower fidelity settings the 95% confidence intervals of the signals do not overlap. Such distinct signals between churners and non-churners resonate with the theory of social exchange: users who remain in the community share more connections (higher in-degree and out-degree entropy) and thus invest more and get more out of the community, churners meanwhile are the converse.

Historical Comparisons. Figure 3 shows the in-degree, out-degree and lexical period cross-entopies for Server Fault, deriving the values as above for the period variation measures for both the *churners* and *non-churners*. We note that across all of the plots churner signals are lower in magnitude than non-churners signals, indicating that the properties of the non-churners tend to have a greater divergence with respect to earlier properties than the churners. This suggests that churners' behaviour is more *formulaic* than non-churners, that is they exhibit less divergence from what has occurred beforehand. In general, the curve of churners and non-churners diminishes towards the end of their lifecycles but with different gradients.

Community Comparisons. For examining how users diverged from the community in which they were interacting we used users' in-degree, out-degree and lexical term distributions and compared them with the same distributions derived globally over the same time periods. For the global probability distributions we used the same means as for forming user-specific distributions, but rather than using the set of posts that a given user had authored (P_u) to derive

[3] We use this platform throughout as an example due to brevity. The remaining platforms exhibit similar curves.

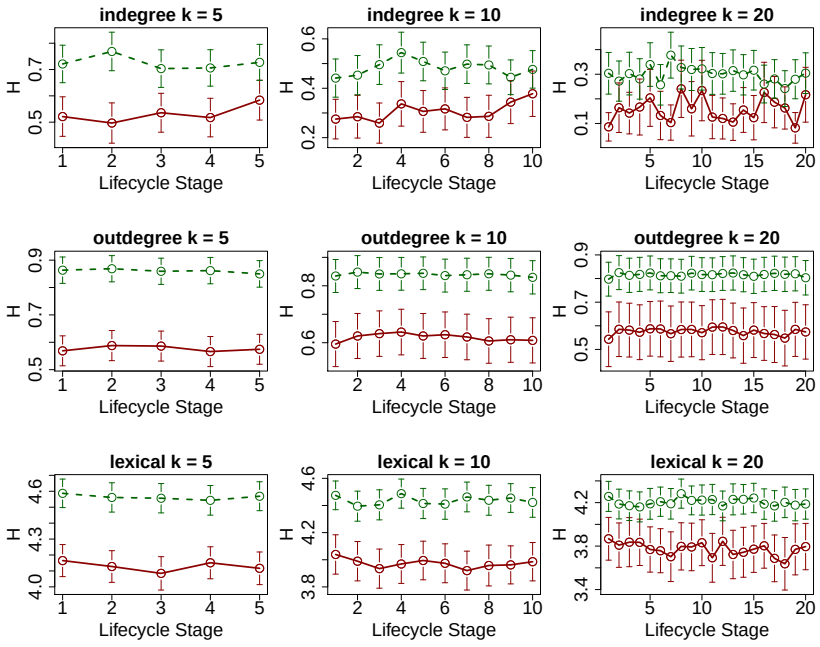


Fig. 2. Period entropy distribution on ServerFault for different fidelity settings (k) for users' lifecycles and different measures of social (indegree and out degree) and lexical dynamics. The green dashed line shows the non-churners, while the red solid line shows the churners.

the probability distribution, we instead used all posts to return Q.[4] We then calculated the the cross-entropy as above between the distributions. ($H(P_u, Q)$) over the different lifecycle stages. Again, as with period cross-entropies, we find churners' signals to have a lower magnitude than non-churners suggesting that non-churners' properties tend to diverge from the community as they progress throughout their lifetime within the online community platforms.

5 Churn Prediction from Gaussian Sequences

Above we plotted the 95% confidence intervals of a given measurement m (e.g. the period entropy of users' in-degree at lifecycle stage 1) for both churners and non-churners. If we assume that the distribution of a given measurement (m) at a particular lifecycle stage (s) is normally distributed, then for each measurement we have two signals (one for churners and one for non-churners)

[4] For instance, for the global in-degree distribution we used the frequencies of received messages for all users.

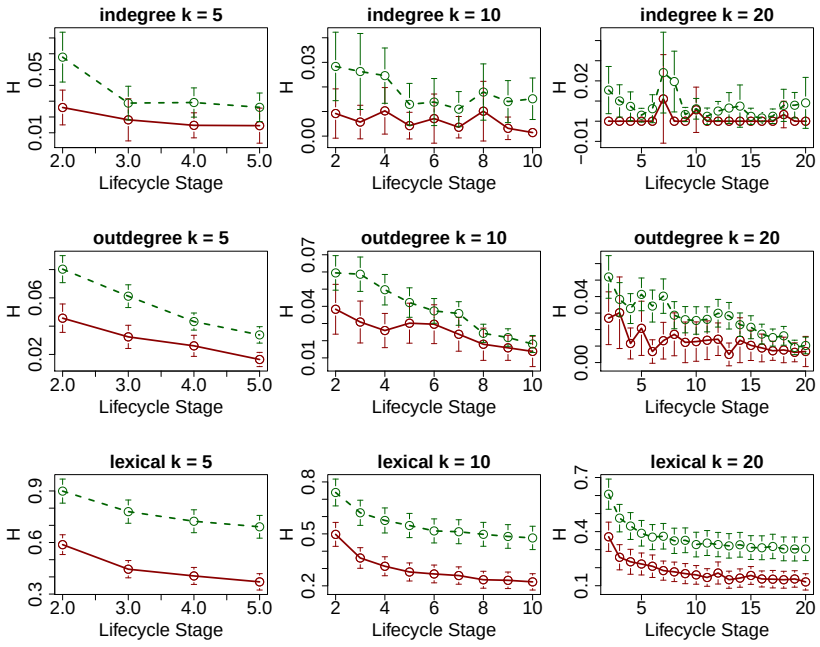


Fig. 3. Period cross-entropy distribution on ServerFault for different fidelity settings (k) for users' lifecycles and different measures of social (indegree and out degree) and lexical dynamics.

that each correspond to a *sequence* of Gaussians measured over the k lifecycle stages:

Definition 1 (Gaussian Sequence). *Let m be a given measurement, s be a given lifecycle stage drawn from the set of lifecycle stages $s \in S$, then m is said to be normally distributed on s and defined by $\mathcal{N}\big(\hat{\mu}_{m,s}, (\hat{\sigma}_{m,s})^2\big)$ where $\hat{\mu}_{m,s}$ and $\hat{\sigma}_{m,s}$ denote the maximum likelihood estimates of the mean and standard deviation respectively. Then the Gaussian Sequence of m is defined as follows:* $G_m = \Big(\mathcal{N}\big(\hat{\mu}_{m,1}, (\hat{\sigma}_{m,1})^2, \mathcal{N}\big(\hat{\mu}_{m,2}, (\hat{\sigma}_{m,2})^2, \ldots, \mathcal{N}\big(\hat{\mu}_{m,|S|}, (\hat{\sigma}_{m,|S|})^2\big)\Big)$.

5.1 Single-Gaussian Sequence Model

Under the assumption that a given measurement has a Gaussian distribution at s then for an arbitrary user (u) we may measure the likelihood that the user belongs within a given distribution given his measurement at that stage. Using the convenience function $f(u, m, s)$ we can compute the probability that the user belongs to the *churn* gaussian, at that time step (s), using:

$$P(u|\beta_{m,s}) \propto \beta_{m,s} \mathcal{N}\big(f(u, m, s)|\hat{\mu}^c_{m,s}, (\hat{\sigma}^c_{m,s})^2\big)$$

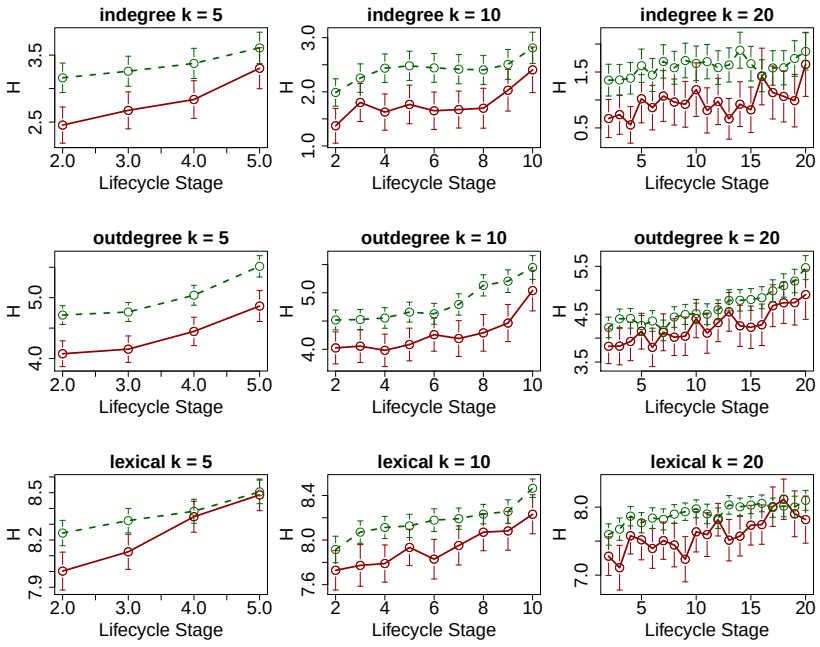


Fig. 4. Community cross-entropy distribution for different fidelity settings (k) for users' lifecycles and different measures of social (indegree and out degree) and lexical dynamics.

In the above equation, $\mathcal{N}\big(f(.)|\hat{\mu}, \hat{\sigma}^2\big)$ defines the conditional probability of the observed measurement $f(.)$ being drawn from the given gaussian of measure m in lifecycle stage s. We have also included a slack variable $\beta_{m,s}$ to control for *influence* on the churn probability; its inclusion is necessary because we may have an outlier measure for u and should limit over fitting as a consequence - note that this variable is indexed by both m and s as it is specific to both the lifecycle stage, and the measure under inspection. Given our formulation of the churn probability in a particular lifecycle stage s and based on measure m, we can therefore derive the joint probability of u churning over the observed sequence of measures ($m \in M$) and his lifecycle stages ($s \in S$) as follows - we term this the *Single-Gaussian Sequence Model*:

$$Q(u|\mathbf{b}) = \prod_{m \in M} \prod_{s \in S} \rho\Big(\beta_{m,s} \mathcal{N}\big(f(u,m,s)|\hat{\mu}^c_{m,s}, (\hat{\sigma}^c_{m,s})^2\big)\Big)$$

The parameter ρ *smooths* zero probability values given our joint calculation. Assuming we have $|S|$ lifecycle stages, and $|M|$ measures, then the slack variables are stored within a parameter vector: $\mathbf{b}$ where - where $\beta_{m,s} \in \mathbf{b}$.

5.2 Dual-Gaussian Sequence Model

The above formulation can be extended further to include two *competing* Gaussian distributions at a particular lifecycle stage: the *churn gaussian*, formed from measurements of the known churner users, and; the *non-churn gaussian*, formed from measurements of known non-churners. We can therefore adapt the probability of the user belonging to the churn gaussian to be as follows:

$$P(u|\beta_{m,s}) \propto \Big[\beta_{m,s}\mathcal{N}\big(f(u,m,s)|\hat{\mu}^{c}_{m,s},(\hat{\sigma}^{c}_{m,s})^2\big) - (1-\beta_{m,s})\mathcal{N}\big(f(u,m,s)|\hat{\mu}^{n}_{m,s},(\hat{\sigma}^{n}_{m,s})^2\big)\Big]_+$$

In this instance we wrap the subtraction of the churn-distribution membership probability by the β-scaled non-churn-distribution membership probability within the positive value operand $[]_+$ in order to return a non-negative value. As above, we can then calculate the joint churn probability over observed measures and lifecycle stages as follows - we term this the *Dual-Gaussian Sequence Model*:

$$Q(u|\mathbf{b}) = \prod_{m\in M}\prod_{s\in S}\rho\Big[\beta_{m,s}\mathcal{N}\big(f(u,m,s)|\hat{\mu}^{c}_{m,s},(\hat{\sigma}^{c}_{m,s})^2\big) - (1-\beta_{m,s})\mathcal{N}\big(f(u,m,s)|\hat{\mu}^{n}_{m,s},(\hat{\sigma}^{n}_{m,s})^2\big)\Big]_+$$

5.3 Model Learning: Dual-Stochastic Gradient Descent

For both the single and dual-gaussian models, our objective is to minimise the squared-loss between a user's forecasted churn probability and the observed churn label - given that the former is in the closed interval $[0,1]$ and the latter is from the set $\{0,1\}$ - our parameters are L2-regularised to control for over-fitting:

$$\arg\min_{\mathbf{b}^*} \sum_{(\mathbf{x}_i,y_i)\in\mathcal{D}} \big(y_i - Q(u|\mathbf{b})\big)^2 + \lambda||\mathbf{b}||_2^2 \tag{1}$$

Using this objective, we can then use gradient descent to calculate the setting of each $\beta \in \mathbf{b}$ by minimising the loss between a single user's forecasted churn probability and his actual churn label (i.e. either 0 - did not churn - or 1 - did churn). We experimented with two learning procedures: stochastic gradient descent (SGD), and dual-stochastic gradient descent (D-SGD) - the latter being a novel contribution of this paper. This latter procedure learns $\mathbf{b}$ with the approach in Algorithm 1, which takes as input a given regularisation weight λ, learning rate η, smoothing variable ρ, the dataset to use for parameter tuning D, the dimensionality of the feature space m, and the convergence threshold ϵ. The algorithm runs two loops: the *outer* loop (lines 4-12) shuffles the order of the dataset's instances and iterates through them, the inner loop (lines 7-11) then shuffles the order of the features. For each feature, the error of predicting the churn label of the user is derived (line 9) and this is used to update the parameter for feature j in the model; this process is repeated until the model's parameters have converged to a degree less than ϵ. We used D-SGD here to avoid sequential updating of parameters, and to examine its effects.

Algorithm 1 Learning the model's parameters using dual-stochastic gradient descent. **Input:** λ, η, ρ, D, m, ϵ. **Output: b**

$\mathbf{b} = \mathbf{0}^m$; $\mathbf{b}_{OLD} = random(m, [0, 1])$; $J = \{0, 1, \ldots m\}$
while $|\beta - \beta_{OLD}| < \epsilon$ **do**
  $\beta_{OLD} = \beta$
  Shuffle D
  for $(\mathbf{x}_i, y_i) \in D$ **do**
    Shuffle J
    for $j \in J$ **do**
      $e = y_i - Q(i|\mathbf{b})$
      $\beta_j \leftarrow \beta_j + \eta * (e - \lambda\beta_j)$
    end for
  end for
end while
return b

6 Evaluation

We now turn to the evaluation of the above models, there are two stages to this: we begin by first tuning the various models' hyperparameters, before then applying the best performing model hyperparameters to the held-out test split of users. All proposed models use a fixed smoothing setting of $\rho = 0.1$ and the convergence threshold to be $\epsilon = 10^{-7}$.

6.1 Model Tuning: Setup

For the above proposed gaussian sequence models we have two hyperparameters that are to be tuned: (i) λ the regularisation weight, and; (ii) η the learning weight. For each model and learning routine (i.e. stochastic or dual-stochastic gradient descent) we set the possible settings for the hyperparameters of each be from $\{10^{-8}, 10^{-7}, \ldots, 10^{-1}\}$. To tune the hyperparameters we used 10-fold cross-validation over the training split with 9 segments for training using a given hyperparameter vector ($\theta = \{\lambda, \eta\}$) to derive the parameter vector $\mathbf{b}$, we then applied this to the 1 segment held-out and recorded the Area Under the ROC Curve (ROC). We repeated this 10 times for the 10 different segments and recorded the mean of these as the 10-fold CV average ROC. Appendix A presents the tuned hyperparameters for the proposed models and learning procedures.

6.2 Baselines

In order to judge how well our approach, and its variant models, performs against existing work we included two baselines. The first baseline we denote as B1-J48: for this we induced a J48 decision tree classifier using the above mentioned features (e.g. in-degree entropy of a user in lifecycle stage 1) using the training split users and applied this to the test split. For the second baseline, that

we denote by B2-NB, we implemented the approach from [2] using features derived from the social network of users: *in-degree, out-degree, closeness-centrality, betweenness-centrality, reciprocity, average number of posts in initiated threads, average number of posts within participated threads, popularity (% of user authored posts that receive replies), initialisation (% of threads authored by the user), and polarity.* We first tested the J48 classifier, as used in [2], but found this to be poor performing[5] therefore we used the Naive Bayes classifier instead.

Table 2. Area under the Receiver Operator Characteristic (ROC) Curve results for the different Gaussian Sequence Models and Learning Procedures

		Baselines		SGD		D-SGD	
Platform	Lifecycle Fidelity	B1-J48	B2-NB	Single-$\mathcal{N}$	Dual-$\mathcal{N}$	Single-$\mathcal{N}$	Dual-$\mathcal{N}$
Facebook	5	0.559	0.461	**0.570**	0.472	0.548	0.478
	10	0.531	0.491	0.569	0.554	**0.593**	0.545
	20	0.478	0.444	**0.664**	0.500	0.528	0.583
SAP	5	**0.594**	0.497	0.573	0.527	0.545	0.533
	10	0.533	0.494	0.553	0.503	0.584	**0.590**
	20	0.478	**0.582**	0.500	0.500	0.540	0.525
ServerFault	5	0.583	0.530	0.522	0.556	**0.583**	0.577
	10	0.534	0.546	0.500	0.557	0.569	**0.589**
	20	0.463	0.530	0.500	**0.634**	0.486	0.484
Boards.ie	5	0.504	**0.611**	0.524	0.547	0.526	0.518
	10	0.512	**0.593**	0.500	0.539	0.501	0.496
	20	**0.560**	0.553	0.500	0.501	0.500	0.502

6.3 Results: Churn Prediction Performance

For the model testing phase of the experiments, we took the best performing hyperparameters for each model and learning procedure, trained the model using this setting using with entire training split, and then applied it to the test split; we did this twenty-times for each model (as each induction of the parameter vector is affected by the stochastic nature of the learning procedure) and took the average ROC value. These ROC values for the different models and baselines are shown in Table 2. The results show that for certain proposed models we significantly outperformed the baselines for two of the datasets.[6] Surpassing B1-J48 indicates that our proposed Gaussian models beat a widely-used classification model when detecting churners - given that this baseline makes use of the same features as our proposed model.

The results indicate variance across the prediction model as to which model performs best and under what conditions. For instance, the single-gaussian model

[5] We also tested support vector machines and the perceptron classifier.

[6] Testing for significance using the Student T-test for independent samples.

performs better overall than the dual-gaussian model: this is largely due to the latter model smoothing zero-probability values through the setting of ρ. Future work will experiment with ρ, either by indexing prediction models by this value or by tuning it as a hyperparameter. There appears to be no discernible *winner* in terms of the learning procedure to adopt, thus with dual-stochastic gradient descent being more computationally expensive we would lean towards using stochastic gradient descent in its place.

7 Conclusions and Future Work

In this paper we have presented a means to predict churners based on Gaussian Sequences. Our approach assumes that measures of user development are normally distributed, at each discrete lifecycle stage, and from which the probability of a user belonging to a churner or non-churner class can be gauged. We proposed two models to detect churners: the first using a single Gaussian Sequence formed from known churners' development information, and a second approach using dual-Gaussian Sequences from both churners' and non-churners' development information. Evaluation demonstrated that our detection models outperformed the two baselines - including the popular J48 decision tree classifier - for two of the tested online community datasets.

To the best of our knowledge this is the first work to *directly* compare the development signals of churners and non-churners and use that information to inform predictions. Our own prior work [8] focused on inducing regression models that capture the development trajectory of the above-mentioned measures. We implemented this same approach across the reduced lifecycle fidelity settings (i.e. $k = \{5, 10\}$) and found that: (i) its fit was poor for lower lifecycle fidelities; and (ii) prediction experiments resulted in low ROC values, in many cases zero. The presented approach in this paper therefore surpasses our own prior work in terms of its applicability to lower settings of lifecycle fidelities, and thus more users.

The first area of further work will explore the use of different objective functions that are to be optimised: above we used a reduction in the squared-error, yet an objective that accounts for rankings of users, based on their churn probability, would be better suited given the use ROC as our evaluation measure. The second area of future work will explore the task of *churn point prediction*: forecasting the day at which the user posts for the last time, our approach is amenable to such a setting via changing predictive function's codomain.

References

1. Marcel Karnstedt, Tara Hennessy, Jeffrey Chan, Partha Basuchowdhuri, Conor Hayes, and Thorsten Strufe. Churn in social networks. In *Handbook of Social Network Technologies and Applications*, pages 185–220. Springer, 2010.
2. Marcel Karnstedt, Matthew Rowe, Jeffrey Chan, Harith Alani, and Conor Hayes. The effect of user features on churn in social networks. *Proceedings of the ACM WebSci*, 11:14–17, 2011.

3. Haewoon Kwak, Hyunwoo Chun, and Sue Moon. Fragile online relationship: a first look at unfollow dynamics in twitter. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1091–1100. ACM, 2011.
4. Haewoon Kwak, Sue B Moon, and Wonjae Lee. More of a receiver than a giver: Why do people unfollow in twitter? In *ICWSM*, 2012.
5. Kevin Lewis, Marco Gonzalez, and Jason Kaufman. Social selection and peer influence in an online social network. *Proceedings of the National Academy of Sciences*, 109(1):68–72, 2012.
6. D. McGowan, A. Brew, B. Casey, and N.J. Hurley. Churn prediction in mobile telecommunications. In *Proceedings of the 22nd Irish Conference on Artificial Intelligence and Cognitive Science*, 2011.
7. Daniele Quercia, Mansoureh Bodaghi, and Jon Crowcroft. Loosing friends on facebook. In *Proceedings of the 3rd Annual ACM Web Science Conference*, pages 251–254. ACM, 2012.
8. Matthew Rowe. Mining user lifecycles from online community platforms and their application to churn prediction. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 637–646. IEEE, 2013.
9. Xiaohang Zhang, Zhiyu Liu, Xuecheng Yang, Wenhua Shi, and Qi Wang. Predicting customer churn by integrating the effect of the customer contact network. In *Service Operations and Logistics and Informatics (SOLI), 2010 IEEE International Conference on*, pages 392–397. IEEE, 2010.

A Appendix: Model Tuning Results

Table 3. Tuned hyperparameters for the various proposed models as λ, η pairs

		SGD		D-SGD	
Platform	Fidelity	Single-$\mathcal{N}$	Dual-$\mathcal{N}$	Single-$\mathcal{N}$	Dual-$\mathcal{N}$
Facebook	5	$0.1, 0.01$	$0.1, 0.01$	$10^{-5}, 0.1$	$0.1, 0.01$
	10	$0.1, 0.01$	$10^{-5}, 0.01$	$0.01, 0.1$	$10^{-5}, 0.01$
	20	$0.1, 0.01$	$0.001, 0.1$	$0.1, 0.1$	$0.01, 0.1$
Sap	5	$0.1, 0.001$	$10^{-5}, 0.01$	$10^{-6}, 0.01$	$0.01, 0.01$
	10	$0.1, 0.01$	$0.1, 0.01$	$0.001, 0.1$	$0.01, 0.1$
	20	$0.1, 0.01$	$0.1, 0.01$	$0.001, 0.1$	$0.001, 0.1$
ServerFault	5	$10^{-5}, 0.01$	$0.1, 0.01$	$0.01, 0.1$	$0.1, 0.01$
	10	$0.1, 0.1$	$0.01, 0.01$	$0.001, 0.1$	$0.01, 0.1$
	20	$10^{-6}, 0.1$	$0.01, 0.01$	$10^{-5}, 0.1$	$0.01, 0.1$
Boards.ie	5	$10^{-6}, 0.001$	$0.1, 0.001$	$0.001, 0.1$	$0.1, 0.001$
	10	$0.1, 0.1$	$0.01, 0.001$	$0.001, 0.1$	$0.001, 0.001$
	20	$0.1, 0.1$	$0.1, 0.01$	$0.1, 10^{-6}$	$10^{-5}, 0.1$