Bias Sampling Correction for Supervised Learning

A Probabilistic, Bayesian Approach

Max Sklar Another Author?

July 27, 2021

Abstract

abstract

1 Background

While so-called big data has unlocked incredible new applications of machine learning, all of this data processing comes at a heavy cost in terms of time, money, and energy. Many models will perform just as well had they been trained on a fraction of the data available, or at least the performance difference is not enough to justify the cost of including all data. Therefore, sampling or removing data from learning systems is a legitimate design decision.

The obvious way to slim down data is through random sampling. For each training example, a random number could be generated uniformly between 0 and 1, and then if the random number is less than a certain value, that datapoint remains in the sampled set.

This simple process ignores the fact that sometimes - particularly when there is an imbalanced data set - certain training examples are worth more than others.

1.1 Example: Image Recognition

For example, suppose that the goal was to train an image recognition algorithm to recognize the image of a lion. Setting aside the need for test and validation sets for a moment, suppose that the training data contains 2000 images of lions, and about 400,000 images that are *NOT* lions.

A train on all of the data

B randomly sample 25 percent of the data

C randomly sample a quarter of ONLY the non-lion images, so that all 2000 of the positive examples could be used.

Can we create models that are just as good, or almost as good if we reduce the amount of data points by an order of magnitude? Not all data is equally important, so we want to sample from the original dataset Simple Example of not all data equally important (Lion detector)

Model B is probably wrong.

If we use model C (Illustrate these models) without taking into account the sampling, a model will hopefully be able to detect the core visual features of a Lion and therefore won't be too biased.

1.2 Example: Probabilistic Event Detection

The need to account for uneven sampling becomes far more critical in the case of a probabilistic model where we expect the results to be highly uncertain and are relying on accurate probabilities of the potential outputs.

One example of this is an iteration of Foursquare's attribution model, which has a pending patent (source)... As part of a product that measures the ability of an online advertisement to drive consumers to the stores of a retail chain (let's say Starbucks), the Foursquare data pipeline first had to build a model that calculates the probability that any given individual would visit starbucks on any given day.

Foursquare has access to a visit database with many examples of visits to Starbucks, but the examples of people who "did not visit" starbucks on any given day is an order of magnitude higher! Therefore, it makes sense to downsample those non-visits...

And because a precise probability is necessary for best-in-class ad measurement, that sampling much be accounted for appropriately.

1.3 Goals

Formulate a (generalized) supervised machine learning problem with a sampling function that allows the user to determine which pieces of data are most important. This also works if the original dataset is unavailable and the sampling function is still known. Find a formula for a probability distribution over models (or a loss function) that works with the information in the sampler to best model the original, raw, dataset Apply this to logistic regression (because it's so common)

1.4 Previous Work

Final Example: Maalouf paper previous work
mention most bibliography items here
Yan Lecun' Energy Paper

2 Definitions and Conventions

The following is a typical setup for a probabilistic supervised learning problem. The terminology and variables will be reused in subsequent sections of this paper.

Let X represent the input space.
Let Y represent the output space.

The dataset D consists of N examples of pairs $(x, y) \in (X \times Y)$. The goal is to predict an output $y \in Y$ from a corresponding input $x \in X$. This dataset may be generated by an oracle from which we can ask for an arbitrary number of examples, or it might be a small and limited collection. In any case, our learning will be based off of just these N examples.

For all $n \in (0, 1, 2, \ldots, N-1)$, label the specific instance (x_n, y_n).

[Explain the terminology model, and why h is used as hypothesis]

[Explain terminology probability measure vs probability distribution]

Consider a hypothesis space H which contains all the predictors, or (models), under consideration. In some setups, each hypothesis $h \in H$ is a direct function from X to Y, which means that these predictors are *directly predicting* $y \in Y$. Here we assume that each hypothesis is a function of $x \in X$ and returns a *probability measure* over Y. This is what we mean by *probabilistically predicting* $y \in Y$.

While the equations in this paper generalize to all probability measures over the output space Y, we keep several cases in mind.

1. Y is finite. This is a problem of *classification*. This probability measure is the most familiar as it is simply a finite sum that adds to one.

2. Y is discrete but infinite. Here the probabily measure will still assign a number to each potential output, where not the infinite sum adds to one.

3. Y is infinite and continuous. Now each hypothesis corresponds to a *probability distribution function* (PDF) over Y, which integrates to one.

In all of these cases, the model assigns a number to each $y \in Y$ which can be used to compute the relative probability of each potential output given an input x. In the case of discrete Y this relative probability of $y_1, y_2 \in Y$ is just the ratio of their individual probabilities, and for continuous Y it is the ratio of their values in the probability distribution function. Our generalization will extend to these cases, which is very broad. It even includes the possibility of an *improper probability distribution function*, for example one that assigns an unnormalized PDF score of 1 to each real number $Y = \Re$ even though such a function cannot be normalized.

Given this setup, we can say that each model h is uniquely identified with an (unnormalized) *relative probability function* $f_h : X \times Y \to \Re^+$.

In cases where this probability function can be normalized, we use a normalized probability function $\hat{f}_h : X \times Y \to \Re^+$. When Y is discrete, we can say

$$\hat{f}_h(x, y) \left(\sum_{y' \in Y} f_h(x, y') \right) = f_h(x, y). \tag{1}$$

Once the hypothesis space has been defined, the task is either to *select* a model or *sample* a model that best explains the data. In the case of selection, the analysis provides a a single model whereas in sampling, a series of models are randomly generated according to their relative strength in explaining the data.

In the simplest case where the hypothesis space H is finite and small, we will be able to calculate the probability of each $h \in H$ and either select the model with the highest probability, or sample from it using a random number generator. In general, the hypothesis space H could be very complex and a variety of methods for selection and sampling have been developed. [SOURCE]

In all of these cases, we again rely on the concept of *relative probability*, this time between two potential models $h_1, h_2 \in H$ and their predictive power. Therefore, H can also be a discrete or continuous probability measure, just like the output space Y

[Talk about a prior here]

Bayes rule in full for discrete H to compute the probability of individual model h_i is given as follows, where $\mathbf{P}(h)$ is the *prior probability* of model h and $\mathbf{P}(D|h)$ is the probability of seeing the outputs in dataset D under the model h

$$\mathbf{P}(h_i|D) = \frac{\mathbf{P}(D|h_1)\mathbf{P}(h_1)}{\sum_{h \in H} \mathbf{P}(D|h)}$$

Bayes rule for relative probability is can be defined as follows where our probability functions such as $\mathbf{P}(h|D)$ assign unnormalized values:

$$\mathbf{P}(h|D) \propto \mathbf{P}(D|h)\mathbf{P}(h) \tag{2}$$

The relative probability of the dataset under model h, denoted by $\mathbf{P}(D|h)$, is the product of the relative probability function of each instance label y_n being produced.

$$\mathbf{P}(D|h) \propto \prod_{n \in N} f_h(x_n, y_n)$$

Finally, the formula for relative probability of a model is given by

$$\mathbf{P}(h|D) \propto \prod_{n \in N} f_h(x_n, y_n)\mathbf{P}(h)$$

.

In most cases, it is more convenient to turn this into a *negative log-likilihood loss* function by applying $-\ln(\ldots)$ of the right hand side of this equation and getting:

$$L(h) = -\sum_{n \in N} \ln(f_h(x_n, y_n)) - \ln(\mathbf{P}(h))$$

We can simplify further by assuming each hypothesis to comes with its own negative log-likelihood loss function l_h where $f_h(x_n, y_n) \propto e^{-l_h(x_n, y_n)}$ and for the prior $\mathbf{P}(h)$ to be reduced to a regularization function $\mathbf{r}(h)$ where $\mathbf{P}(h) \propto e^{-\mathbf{r}(h)}$. With that in place, the final form of the loss function is as follows:

$$L(h) = \sum_{n \in N} l_h(x_n, y_n) + \mathbf{r}(h)$$

This form is often more convenient when it comes to model selection or sampling, through techniques such as hill climbing, gradient descent, or Markov Chain Monte Carlo. [Include some citations] [Should I introduce a catch-all term for "selection or sampling"?]

3 The Sampling Problem

The previous section sets up a loss function for a standard probabilistic supervised learning problem.

In this setup, it is assumed that D is either a unsampled dataset or if it has been sampled down it was done so randomly.

In an extreme example, suppose that the output set Y consists of 3 classifications $\{yes, no, maybe\}$ and whenever the result of one of the classes, say no, appears in the dataset, the results were thrown away. The model would learn that $\{yes, maybe\}$ were the possible responses, and would never put any significant amount of probability towards the no option!

The key point of this paper is that when the sampling function is known, a probability distribution and loss function can still be computed for the remaining data. The sampling will affect what that final loss function knows about the data, but the results will represent the best-guess from the data available.

Take the $\{yes, no, maybe\}$ output space from above as an example where all of the no examples are sampled out. If the methods in this paper are deployed, the models in H that are ultimately selected will be the ones that correctly distinguish between the yes and $maybe$ responses. Their assignments of no will vary widely - as only the information in the prior will be used to discriminate between them.

The formal statement of the problem is as follows:

Suppose that the dataset D was generated from an even larger dataset $D^+$of unknown size with a sampling function $\mathbf{s} : X \times Y \to [0, 1]$. For each element $(x, y) \in D^+$, the sampling function $\mathbf{s}(x, y)$ is calculated and returns the probability that this example will be in D. Now from just the dataset D and knowledge of $\mathbf{s}$ we calculate the posterior probability and loss function in order to correctly apply Bayesian inference.

4 The General Solution

We start with the unnormalized Bayes rule in equation (2), but now considering that the posterior distribution and likelihood both depend on the sampling function $\mathbf{s}$.

$$\mathbf{P}(h|D,\mathbf{s}) \propto \mathbf{P}(D|h,\mathbf{s})\mathbf{P}(h)$$

As before, we calculate the likelihood.

$$\mathbf{P}(D|h) = \prod_{n\in N} \mathbf{P}(y_n|x_n,h,\mathbf{s})$$

Note that the input x_n, model h and sampler $\mathbf{s}$ are all taken as given in the expression $\mathbf{P}(y_n|x_n,h,\mathbf{s})$. The question is the probability of producing the label y_n under these circumstances.

Also note that if there is no sampling, then $\mathbf{s}(x,y) = 1$ and $\mathbf{P}(y_n|x_n,h,\mathbf{s}) = \mathbf{P}(y_n|x_n,h) = f_h(x_n,y_n)$ thus reducing to the original problem.

Each model $h \in H$ is a description of the process of how dataset D is generated. The following mental model is a useful tool to understanding how the sampling function s factors into the generation of d under any given $h \in H$.

1. Consider a specific value of x_n as the given input.
2. Under a model h, we have a probability distribution over Y which is encoded by the relative probability function $f_h(x_n,y_n)$ or loss function $l_h(x_n,y_n)$.
3. Sample from that probability distribution, and make this a candidate for y_n, called y_n^*
4. Compute the sampling rate $\mathbf{s}(x_n,y_n^*)$ and use that rate to probabilistically determine whether or not y_n^* is accepted.
   (a) If it is accepted, set $y_n = y_n^*$, and we have successfully generated a label for x_n
   (b) If it is not accepted, return to step 3 to generated another candidate.

The sampling function must eventually halt which means that it cannot be equal to 0 for every potential candidate of any given x_n. This is a fair assumption because if it were false, x_n would never appear in the dataset D.

The generative description above can be used to produce a recursive equation for the conditional probability of a given output in the dataset under the sampling function and in a given model $\mathbf{P}(y_n|x_n,h,\mathbf{s})$.

Once a given candidate y is selected, the probability of ultimately accepting y depends on whether the candidate is accepted by the sampler or not. If it is accepted, this probability is equal to 1 if $y_n = y$ and 0 otherwise, given by the indicator function $[y_n = y]$. If it is not accepted, then we the probability reverts to the original value of $\mathbf{P}(y_n|x_n,h,\mathbf{s})$. Therefore, the probabilty of landing on y_n given that the candidate is y comes to

$$\mathbf{s}(x_n, y)\,[y_n = y] + (1 - \mathbf{s}(x_n, y))P(y_n|x_n, h, \mathbf{s}).$$

If we assume that Y is discrete, we can use the normalized probability function of model h given in equation (1) to sum over all possible candidates y and setup the recursive equation

$$\mathbf{P}(y_n|x_n, h, \mathbf{s}) = \sum_{y \in Y} \hat{f}_h(x_n, y)\big(\mathbf{s}(x_n, y)\,[y_n = y] + (1 - \mathbf{s}(x_n, y))P(y_n|x_n, h, \mathbf{s})\big). \tag{3}$$

With algebraic manipuation detailed in appendix A, we can solve for $\mathbf{P}(y_n|x_n, h, \mathbf{s})$ and reduce $\hat{f}$ to f and derive the formula

$$\mathbf{P}(y_n|x_n, h, \mathbf{s}) = \frac{f_h(x_n, y_n)\mathbf{s}(s_n, y_n)}{\sum_{y \in Y} f_h(x_n, y)\mathbf{s}(x_n, y)}. \tag{4}$$

For a continuous output space Y, the following analogous equation can be derived in a similar way, using integrals rather than a sums.

$$\mathbf{P}(y_n|x_n, h, \mathbf{s}) = \frac{f_h(x_n, y_n)\mathbf{s}(s_n, y_n)}{\int_{y \in Y} f_h(x_n, y)\mathbf{s}(x_n, y)}. \tag{5}$$

The feasibility of computing the sum or integral term depends on the structure of Y, but is at least easy when Y is finite and small enough to enumerated by a machine. The formula generalizes to more exotic probability measures on Y, though the rigorous treatment of this is outside our scope.

4.1 Formula for Negative Log-Likelihood

With equation (4), we now have all the tools needed to assign relative probabilities to the hypothesis space H. To get it in the form most helpful to form for machine learning algorithms, we derive a negative log likelihood loss function. Start with Bayes:

$$\mathbf{P}(h|D, \mathbf{s}) \propto \prod_{n \in N} \left[\mathbf{P}(y_n|x_n, h, \mathbf{s})\right] \mathbf{P}(h)$$

Then, use equation (4) to get it in terms of f_h:

$$\mathbf{P}(h|D, \mathbf{s}) \propto \prod_{n \in N} \left[f_h(x_n, y_n)\mathbf{s}(s_n, y_n) \left[\sum_{y \in Y} f_h(x_n, y)\mathbf{s}(x_n, y) \right]^{-1} \right] \mathbf{P}(h)$$

Finally, derive a negative likelihood loss function on h:

$$L(h) = \sum_{n \in N} -\ln \left[f_h(x_n, y_n)\mathbf{s}(s_n, y_n) \left[\sum_{y \in Y} f_h(x_n, y)\mathbf{s}(x_n, y) \right]^{-1} \right] + \mathbf{r}(h)$$

$$L(h) = \sum_{n \in N} \left[l_h(x_n, y_n) - \ln \mathbf{s}(s_n, y_n) + \ln \sum_{y \in Y} f_h(x_n, y)\mathbf{s}(x_n, y) \right] + \mathbf{r}(h)$$

5 Solution for Binary Logistic Regression

Binary logistic regression is a special case where the following is true:

The output space is binary: $Y = 0, 1$. The input space is a list of $|F|$ real valued features: $X = \Re^{|F|}$.

The hypothesis space consists of the weights and intercept of a logistic regression. $H = \Re^{|F|}$ where $(c, \mathbf{w}) \in H$ indicates that $c \in \Re$ is the constance intercept and $\mathbf{w} \in \Re^{|F|}$ is an —F—-dimentional vector of weights corresponding to each input feature.

Each hypothesis $(c, \mathbf{w}) = h \in H$ corresponds to the following probability distribution function:

$$f_h(x_n, y_n) = f_{c,\mathbf{w}}(x_n, y_n) = \frac{e^{y_n \cdot (c + \mathbf{w} \cdot x_n)}}{1 + e^{c + \mathbf{w} \cdot x_n}}$$

Therefore, using equation (4):

$$\mathbf{P}(y_n | x_n, h, \mathbf{s}) = \frac{f_h(x_n, y_n)\mathbf{s}(s_n, y_n)}{\sum_{y \in Y} f_h(x_n, y)\mathbf{s}(x_n, y)}$$

$$\mathbf{P}(y_n | x_n, h, \mathbf{s}) = \frac{\frac{e^{y_n \cdot (c + \mathbf{w} \cdot x_n)}}{1 + e^{c + \mathbf{w} \cdot x_n}}\mathbf{s}(s_n, y_n)}{\sum_{y \in Y} \frac{e^{y \cdot (c + \mathbf{w} \cdot x_n)}}{1 + e^{c + \mathbf{w} \cdot x_n}}\mathbf{s}(x_n, y)} = \frac{e^{y_n \cdot (c + \mathbf{w} \cdot x_n)}\mathbf{s}(s_n, y_n)}{\sum_{y \in Y} e^{y \cdot (c + \mathbf{w} \cdot x_n)}\mathbf{s}(x_n, y)}$$

$$\mathbf{P}(y_n | x_n, h, \mathbf{s}) = \frac{e^{y_n \cdot (c + \mathbf{w} \cdot x_n)}\mathbf{s}(s_n, y_n)}{e^{0 \cdot (c + \mathbf{w} \cdot x_n)}\mathbf{s}(x_n, 0) + e^{1 \cdot (c + \mathbf{w} \cdot x_n)}\mathbf{s}(x_n, 1)} = \frac{e^{y_n \cdot (c + \mathbf{w} \cdot x_n)}\mathbf{s}(s_n, y_n)}{\mathbf{s}(x_n, 0) + e^{c + \mathbf{w} \cdot x_n}\mathbf{s}(x_n, 1)}$$

Typically in these cases, we want to focus on the probability that $y_n = 1$, or the probability that the target condition is met. This target condition is also usually the rare condition that incentivized the use of adaptive sampling to begin with.

$$\mathbf{P}(y_n = 1 | x_n, h, \mathbf{s}) = \frac{e^{c + \mathbf{w} \cdot x_n}\mathbf{s}(s_n, 1)}{\mathbf{s}(x_n, 0) + e^{c + \mathbf{w} \cdot x_n}\mathbf{s}(x_n, 1)}$$

$$\mathbf{P}(y_n = 1|x_n, h, \mathbf{s}) = \frac{e^{c+\mathbf{w}\cdot x_n}}{r_n + e^{c+\mathbf{w}\cdot x_n}}$$

Here, $r_n = \mathbf{s}(s_n, 0)(\mathbf{s}(s_n, 1))^{-1}$ is the true-to-false sample ratio for each instance n. This ratio is all that is needed to compute results for the binary logistic regression case. Note that for $r_n = 1$, we are left with the usual binary logistic regression formula.

For the prior, we can use a Gaussian distribution (aka L2, or ridge regression) with weight λ so:

$$\mathbf{r}(c, \mathbf{w}) = \frac{1}{2}\lambda(\mathbf{w} \cdot \mathbf{w})$$

We can put it together to compute a negative log-likelihood loss function. Some constant factors were dropped in this case.

$$L(h) = \sum_{n \in N} \left(\ln\left(r_n + e^{c+\mathbf{w}\cdot x_n}\right) - y_n \cdot (c + \mathbf{w} \cdot x_n)\right) + \frac{1}{2}\lambda(\mathbf{w} \cdot \mathbf{w})$$

The derivative on a single weight $\mathbf{w}_f$ can be computed to the following simple form, which is required for gradient descent and some versions of Markov Chain Monte Carlo.

$$\frac{\partial}{\partial \mathbf{w}_f} L(h) = \sum_{n \in N} \left(x_{n,f} \frac{e^{c+\mathbf{w}\cdot x_n}}{r_n + e^{c+\mathbf{w}\cdot x_n}} - y_n \cdot x_{n,f} \right) + \lambda \cdot w_f$$

$$\frac{\partial}{\partial \mathbf{w}_f} L(h) = \sum_{n \in N} x_{n,f} \left(\mathbf{P}(y_n = 1|x_n, h, \mathbf{s}) - y_n\right) + \lambda \cdot w_f$$

6 Future Work

Concept of the marginal Value of a Data Point

Appendices

A Solving for the General Formula

Here we start with equation (3) and go through the series of steps neccesary to derive its final form in equation (4). Equation (3) beings as follows:

$$\mathbf{P}(y_n|x_n,h,\mathbf{s}) = \sum_{y\in Y} \hat{f}_h(x_n,y)\big(\mathbf{s}(x_n,y)\,[y_n=y] + (1-\mathbf{s}(x_n,y))P(y_n|x_n,h,\mathbf{s})\big)$$

Break out the summation as follows:

$$\mathbf{P}(y_n|x_n,h,\mathbf{s}) = \sum_{y\in Y} \hat{f}_h(x_n,y)\mathbf{s}(x_n,y)\,[y_n=y] + \sum_{y\in Y} \hat{f}_h(x_n,y)(1-\mathbf{s}(s_n,y))P(y_n|x_n,h,\mathbf{s})$$

Note that in the first summation, the only non-zero addend will occur when $y_n = y$ and the corresponding indicator function is 1. Therefore, we can replace y with y_n and remove the summation and the indication function.

$$\mathbf{P}(y_n|x_n,h,\mathbf{s}) = \hat{f}_h(x_n,y_n)\mathbf{s}(x_n,y_n) + P(y_n|x_n,h,\mathbf{s})\sum_{y\in Y} \hat{f}_h(x_n,y)(1-\mathbf{s}(s_n,y))$$

Now collect the term $\mathbf{P}(y_n|x_n,h,\mathbf{s}$ and break down the remaining summation.

$$\mathbf{P}(y_n|x_n,h,\mathbf{s})\left[1-\sum_{y\in Y} \hat{f}_h(x_n,y)(1-\mathbf{s}(x_n,y))\right] = \hat{f}_h(x_n,y_n)\mathbf{s}(s_n,y_n)$$

$$\mathbf{P}(y_n|x_n,h,\mathbf{s})\left[1-\sum_{y\in Y} \hat{f}_h(x_n,y)+\sum_{y\in Y} \hat{f}_h(x_n,y)\mathbf{s}(x_n,y)\right] = \hat{f}_h(x_n,y_n)\mathbf{s}(s_n,y_n)$$

Because we are using the normalized probability function $\hat{f}$, and from equation (1) we know that for all possible inputs x, $\sum_{y\in Y} \hat{f}_h(x,y) = 1$, we can replace the summation over $\hat{f}$ and cancel as follows:

$$\mathbf{P}(y_n|x_n,h,\mathbf{s})\left[1-1+\sum_{y\in Y} \hat{f}_h(x_n,y)\mathbf{s}(s_n,y)\right] = \hat{f}_h(x_n,y_n)\mathbf{s}(x_n,y_n)$$

$$\mathbf{P}(y_n|x_n,h,\mathbf{s})\left[\sum_{y\in Y} \hat{f}_h(x_n,y)\mathbf{s}(s_n,y)\right] = \hat{f}_h(x_n,y_n)\mathbf{s}(x_n,y_n)$$

Finally, we note that because of the equation (1), the $\hat{f}_h$ terms are simply a constant factor of same terms with the unnormalized f_h. Because this normalization factor appears on both sides of the equation, $\hat{f}_h$ can be reduced to f through cancellation. Thus with one extra step of division, the final form can be given as (4).

$$\mathbf{P}(y_n|x_n,h,\mathbf{s}) = \frac{f_h(x_n,y_n)\mathbf{s}(s_n,y_n)}{\sum_{y\in Y} f_h(x_n,y)\mathbf{s}(x_n,y)} \tag{4}$$

If we assume that there was no sampling, or the sampling was uniform where $\mathbf{s}(x, y) = p$, equation (4) should reduce to the original model probability function $\hat{f}_h(x_n, y_n)$. This sanity check can be easily worked out:

$$\mathbf{P}(y_n|x_n, h, \mathbf{s}) = \frac{f_h(x_n, y_n)\mathbf{s}(x_n, y_n)}{\sum_{y \in Y} f_h(x_n, y)\mathbf{s}(x_n, y)} = \frac{f_h(x_n, y_n)p}{\sum_{y \in Y} f_h(x_n, y)p} = \hat{f}_h(x_n, y_n)$$

7 Notes: Potential references

Yan Lecun's Energy Paper?

The paper Ely found that has the equation in it for Binary Logistic: ¡ https://onlinelibrary.wiley.com/doi/full/10.1111/coin.121 ¿. This has other citations that may be worthwhile for us to read, especially if they involve anything about the history of this modified logistic regression algorithm.

Reference the most recently Foursquare patent. Find out when it is available.

The following references come from my 2014 paper on MLE for Dirichlet Prior. They should be changed to what we want for this paper

References

[1] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. the Journal of machine Learning research, 3, 993-1022.

[2] Dishon, M., & Weiss, G. H. (1980). Small sample comparison of estimation methods for the beta distribution. Journal of Statistical Computation and Simulation, 11(1), 1-11.

[3] Hariharan, H. S., & Velu, R. P. (1993). On estimating dirichlet parameters—a comparison of initial values. Journal of statistical computation and simulation, 48(1-2), 47-58.

[4] Heinrich, G. (2005). Parameter estimation for text analysis. Web: http://www. arbylon. net/publications/text-est. pdf.

[5] Hijazi, R. H., & Jernigan, R. W. (2009). Modelling compositional data using Dirichlet regression models. Journal of Applied Probability & Statistics, 4, 77-91.

[6] Minka, T. (2000). Estimating a Dirichlet distribution. Technical report, M.I.T., 2000.

[7] Narayanan, A. (1991). Algorithm AS 266: maximum likelihood estimation of the parameters of the Dirichlet distribution. Journal of the Royal Statistical Society. Series C (Applied Statistics), 40(2), 365-374.

[8] Ng, K. W., Tian, G. L., & Tang, M. L. (2011). Dirichlet and Related Distributions: Theory, Methods and Applications (Vol. 888). John Wiley & Sons.

[9] Robert, C. (2007). The Bayesian choice: from decision-theoretic foundations to computational implementation. Springer.

[10] Ronning, G. (1989). Maximum likelihood estimation of Dirichlet distributions. Journal of statistical computation and simulation, 32(4), 215-221.

[11] Sklar, M., Shaw, B., & Hogue, A. (2012, September). Recommending interesting events in real-time with foursquare check-ins. In Proceedings of the sixth ACM conference on Recommender systems (pp. 311-312). ACM.

[12] Wallach, H. M. (2008). Structured topic models for language. Unpublished doctoral dissertation, Univ. of Cambridge.

[13] Sklar, M., Stewart R., Li, R., Bakula, A., & Spears, E. (2020) Visit Prediction. US Patent App. 16/405,481, 2020