Dealing with Big Data: an Online, Row-by-Row, Estimation Tutorial.

Abstract

Novel technological advances allow distributed and automatic measurement of human behavior. While these technologies provide exciting new research opportunities, they also provide challenges: datasets collected using new technologies grow increasingly large, and in many applications the collected data are continuously augmented. These *data streams* make the standard computation of well-known estimators inefficient as the computation has to be repeated each time a new data point enters. In this tutorial paper, we detail *online learning*, an analysis method that facilitates the efficient analysis of Big Data and continuous data streams. We illustrate how common analysis methods can be adapted for use with Big Data using an online, or "row-by-row", processing approach. We present several simple (and exact) examples of the online estimation and we discuss Stochastic Gradient Descent as a general (approximate) approach to estimate more complex models. We end this article with a discussion of the methodological challenges that remain.

keywords Big Data, data streams, machine learning, online learning, Stochastic Gradient Descent

Dealing with Big Data: an Online, Row-by-Row, Estimation Tutorial.

The ever-increasing availability of Internet access, smart phones, and social media has led to many novel opportunities for collecting behavioral and attitudinal data. These technological developments allow researchers to study human behavior at large scales and over long periods of time (Swendsen et al., 2011; Whalen et al., 2014). Because more data is made available for research, these technological developments have the potential to advance our understanding of human behavior (Barrett and Barrett, 2001) and its dynamics. However, these novel data collection technologies also present us with new challenges: If (longitudinal) data are collected from large groups of subjects, then we may obtain extremely large datasets. These dataset might be so large that they cannot be analyzed using standard analysis methods and existing software packages. This is exactly one of the definitions used for the buzz-term "Big Data" (Sagiroglu and Sinanc, 2013; Demchenko et al., 2013): datasets that are so large that they cannot be handled using standard computing machinery or analysis methods.

Handling extremely large datasets represents a technical challenge in its own right, however, the challenge is amplified when large datasets are continuously augmented (i.e., new rows are added to the dataset as new data enter over time). A combination of these challenges is encountered when — for example — data are collected continuously using smart-phone applications (e.g., tracking fluctuations in happiness, Killingsworth and Gilbert, 2010) or when data is mined from website logs (e.g., research into improving e-commerce Carmona et al., 2012). If datasets are continuously augmented and estimates are needed at each point in time, conventional analyses often have to be repeated every time a new data point enters. This process is highly inefficient and frequently forces scholars to arbitrarily stop data-collection and analyze a static dataset. In order to resolve this inefficiency, new methods and/or adapted methods are required to analyze streaming data. To be able to capitalize on the vast amounts of (streaming) data that have become available, we must develop efficient methods. Only if these methods are widely available we will be able to truly improve our understanding of human behavior.

Failing to use appropriate methods when analyzing Big Data or data streams could result

into computer memory overflow or computations that simply take a lot of time. In favorable cases, the time to compute a statistic using standard methods increases linearly with the amount of data entering. For example, if computing the sum over n data points requires t time (where the time unit required for the computation is dependent on the type of machine used, the algorithm used, etc.), then computing the sum over $n+2$ data points requires $t+2c$ time, where c is t/n. Thus, the time increase is linear in n and is every increasing as the data-stream grows. In less fortunate cases the increase in time complexity is not linear but quadratic, or worse, amplifying the problems. Regardless of the exact scaling however, if the data are continuously augmented the required computation time eventually will become infeasible.

The aim of this tutorial paper is to introduce *online learning* (or row-by-row estimation), as a way to deal with Big Data or data streams. Online learning methods analyze the data without storing all individual data points. Therefore, online learning methods have a feasible time complexity (i.e., the time required to conduct the analysis) and they require a feasible amount of computer memory when analyzing data streams or Big Data. In the latter case, a very large static dataset is simply treated as if it were a data stream by iterating through the rows.

Online estimation methods continuously *update* their estimates when new data arrive, and *never revisit* older data points. Formally online learning can be denoted as follows:

$$\theta_n = \{\theta_{n-1}, x_n\},$$

or equivalently (and a shorthand we will use throughout):

$$\theta := \{\theta, x_n\}, \tag{1}$$

where θ is a set of sufficient statistics (not necessarily the actual parameters of interest), which is updated using a new data point, x_n. The second equation for updating θ does not include subscript n because we use the update operator ':=', which indicates that the updated θ is a function of the previous θ and the most recent data point, x_n.

A surprisingly large number of well-known conventional estimation methods used for the analysis of regular (read "small") datasets can quite easily be adapted such that they can handle

Big Data, without losing their straightforwardness or interpretation. We provide a number of examples in this paper. Furthermore, we will also introduce Stochastic Gradient Descent, a general method that can be used for the (approximate) estimation of complex models in data streams.

The paper is organized as follows: In Section 2, a number of conceptual approaches for the estimation of parameters in Big Data and/or data streams are discussed introduced, and we focus primarily on *online learning*, the method further illustrated in the remainder of this paper. In Section 3, we illustrate how often-used estimators such as sample means, variances, and covariances, can be estimated using online learning. Here the benefits of online learning methods to deal with data streams are illustrated by comparing the computational times of online and offline estimation methods. We then, in Section 4, provide an introduction of Stochastic Gradient Descent (SGD) as a general (approximate) method to estimate more complex models in data streams. We finish this section with an example of an application of SGD in the social sciences. Finally, in the last Section, we detail some of the limitations of the online learning approach and we discuss the directions for further research on data streams and Big Data.

2 Dealing with Big Data: the options

In the recent years, data streams and the resulting Big Data have received attention of many scholars. Diverse methods have been developed to deal with these vast amounts of data, which are, in the worst case, ever growing. Conceptually, four overarching approaches to handle Big Data can be identified:

1. sample from the data to reduce the size of the dataset,
2. use a sliding window approach,
3. parallelize the computation,
4. or resort to online learning.

The first option, to sample from the data, solves the problem of having to deal with a large volume of data simply by reducing its size. Effectively, when the dataset is too large to process at once, one could "randomly" split the data into two parts: a part which is used for the analyses and a part of the data that is discarded. Even in the case of data streams, a researcher can decide to randomly include new data points or let them "pass by" to reduce memory burden (Efraimidis and Spirakis, 2006). However, when a lot of data are available, it might be a waste not to use all the data we have.

Option two, using a sliding window, effectively also solves the issue of needing increasingly more computation power by reducing the amount of data that is analyzed. In a sliding window approach the analysis is restricted to the most recent part of the data (Datar et al., 2002; Gaber et al., 2005). Thus, the data are again split into a part which is used for the analyses and a part which is not used for the analysis. The analysis part (i.e., also coined "the window") consists of the m most recent data points, while the second part contains older data which are discarded. When new data enter, the window shifts to include new data and ignore the old data. Although a sliding window approach is feasible in computation time and amount of memory needed, the sliding window approach has the downside that it requires domain knowledge to determine a proper size of the window (e.g., determine m). For instance, when studying a rare event, the window should be much larger, than in the case of a frequent event. It is up to the researcher's discretion to decide how large this window ought to be. Also, when analyzing trends, a sliding window approach might not be appropriate since historical data are ignored.

The third option, using parallel computing, is an often-used method to analyze static Big Data. Using parallel computing, the researcher splits the data in chunks, such that multiple independent machines each analyze a chunk of data, after which the results of the different chunks are combined (see, e.g., Atallah et al., 1989; Chu et al., 2007; Turaga et al., 2010). This effectively solves the problem of memory burden by allocating the data to multiple memory units, and reduces the computation time of static datasets, since analyses which otherwise would have been done 'sequentially' are conducted 'parallel'. However, parallelization is not very

effective when the dataset is continuously augmented: since all data are required for the analyses, computation power has to eventually grow without bound for as long as the dataset is augmented with new data. Also, the operation of combining the results obtained on different chunks of data might itself be a challenge.

In this paper we will focus on a fourth method: online learning (e.g., Bottou, 1998; Shalev-Shwartz, 2011). As introduced in the previous section, online learning uses all available information, but without storing or revisiting the individual data points. Online learning methods can easily be used in combination with parallel computation, (for instance, see Chu et al., 2007; Gaber et al., 2005), but here we discuss it as a unique method that has large potential for use in the social sciences. This method can be thought of as using a very extreme split of the data; the data is split into a part consisting of $n-1$ data points, where n is the total number of observations, and only 1 data point on the other hand. Additionally, in online learning methods, the $n-1$ data points are summarized into a limited set of parameters or estimates of interest, which take all relevant information of previous data points into account (Opper, 1998). The summaries required to estimate the parameters of interest (often the sufficient statistics) are captured by θ. Subsequently, θ is updated using some function of the previous θ and the new data point; historical data points are not revisited.

The two characteristics of online learning – including all the data in the estimate and not revisiting the historical data – jointly make online learning a very suitable approach to analyze data streams. However two downfalls remain, like sliding windows, online learning also requires domain knowledge to judge which information should be gathered beforehand; the researcher needs to choose the elements of θ and their update functions up front. Second, although this issue is not unique for online learning, the researcher often needs to choose starting values for the elements of θ. In the next section we further detail online learning by providing the online adaptation of a number of conventional statistics.

3 From Conventional Analysis to Online Analysis

In this section we discuss online analysis by providing several examples of the online computation of standard (often computed *offline*) estimators. We discuss the online estimation of the following parameters:

1. the sample mean,
2. the sample variance,
3. the sample covariance,
4. linear regression models, and
5. the effect size η^2 (in an ANOVA framework).

The online formulations we discuss in this Section are exact reformulations of their offline counterparts: the results of the analysis are the exact same whether one uses an offline or online estimation method.

3.1 Sample mean

The conventional estimation of a sample mean ($\bar{x}$) is computationally not very intensive since it only requires a single pass through the dataset,

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{2}$$

However, even in this case, online computation can be beneficial. The online update of a sample mean is computed as follows:

$$\begin{aligned} \theta =& \{\bar{x}, n\} : \\ n :=& n+1 \\ \bar{x} :=& \bar{x} + \frac{1}{n}(x_n - \bar{x}). \end{aligned} \tag{3}$$

where we again use the update operator ':=' and start by stating the elements of θ that need to be updated: in this case these are n (a count) and $\bar{x}$ (the sample mean). Note that, which will be true for all examples provided below, appropriate starting value(s) for all the elements θ need to be chosen. In the case of the mean one can easily choose $n = 0$ and $\bar{x} = 0$ as this starting point does not impact the final result – this regretfully will not generally hold. Also note that an online sample mean could also be computed by maintaining $n := n + 1$ and $Sx := Sx + x_n$, where Sx is the sum over x, as the elements of θ; in this case the sample mean could be computed at runtime using $\bar{x} = \frac{Sx}{n}$. This latter method however a) does not actually store the sought for statistic as an element of θ, and b) lets Sx grow without bound, which might lead to numerical instabilities.

3.2 Sample variance

In case of the sample variance (often denoted s^2) more is to be gained when moving from offline to online computation as the conventional method of computing a sample variance requires two passes through the dataset:

$$\begin{aligned} s^2 &= \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2, \\ &= \frac{SS}{n-1}, \end{aligned} \tag{4}$$

where SS is the sum of squares. Here, the first pass is used to estimate the mean $\bar{x}$, while the second pass is used to compute the sum of squares.

A numerically feasible online method to compute a sample variance in a data stream is Welford's method (1962), which, to keep notation consistent, we denote as:

$$\begin{aligned} \theta &= \{\bar{x}, SS, n\} \\ d &= x_n - \bar{x} \\ n &:= n + 1 \\ \bar{x} &:= \bar{x} + \frac{1}{n}(x_n - \bar{x}) \\ SS &:= SS + d(x_n - \bar{x}). \end{aligned} \tag{5}$$

Note the use of auxiliary variable d which is used since the online update of the sum of squares uses both the deviation from the current sample mean as well as from the previous sample mean:

$$SS_n = SS_{n-1} + (x_n - \bar{x}_{n-1})(x_n - \bar{x}_n). \tag{6}$$

In order to obtain the actual sample variance, we compute $s^2 = SS/(n-1)$.

3.3 Sample covariance

Next we turn to the estimation of quantities which depend on multiple variables, for instance the sample covariance between x and y, which is often computed using:

$$\begin{aligned} s_{xy} =& \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x}), \\ =& \frac{SC}{n-1}, \end{aligned} \tag{7}$$

where SC is the sum of cross products. Again, making use of Welford's method (1962), we can estimate the sample covariance online:

$$\begin{aligned} \theta =& \{\bar{x}, \bar{y}, SC, n\} \\ n :=& n+1 \\ \bar{x} :=& \bar{x} + \frac{1}{n}(x_n - \bar{x}) \\ SC :=& SC + (y_n - \bar{y})(x_n - \bar{x}) \\ \bar{y} :=& \bar{y} + \frac{1}{n}(y_n - \bar{y}). \end{aligned} \tag{8}$$

Note that, contrary to the online computation of the sample variance, we do not need auxiliary variables in this case since we can alternate updating of $\bar{x}$ and $\bar{y}$. The choice of which of the two means is updated first, is arbitrary (Pébay, 2008). Similar to the case of the sample variance, to compute the sample covariance, we compute $s_{xy} = SC/(n-1)$.

In Appendix A we present [R] code to compute covariances and correlations, r_{xy}, online. Since computing a correlation entails merely the estimation of sample means, variances, and a covariance it would be repetitious to include it here. We direct the interested reader to Appendix A.

3.4 Linear regression

In applied research, often the aim is to estimate group differences or the effect of a certain independent variable x on an dependent variable y. In such cases the computation of a sample mean of a sample variance will not necessarily suffice to answer the research question. One often used approach to answer research questions about the relationship between one or more independent variables and one dependent variable, is using a linear regression model:

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon, \tag{9}$$

where $\mathbf{y}$ is the vector containing the data of the dependent variable, β is a vector of the regression coefficients of the q independent variables (including an intercept), and $\mathbf{X}$ is the matrix $(n \times q)$ with observed data, including a column of 1's for the intercept. Finally ε denotes the error or noise. When assuming $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, the regression coefficients β are conventionally estimated as follows:

$$\beta = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}, \tag{10}$$

where $\mathbf{X}'$ denotes the transpose of $\mathbf{X}$.

Computing this row-by-row is, perhaps surprisingly, straightforward: We can define $\mathbf{A} = \mathbf{X}'\mathbf{X}$ and $\mathbf{B} = \mathbf{X}'\mathbf{y}$ and compute the update as follows:

$$\begin{aligned} \theta &= \{\mathbf{A}, \mathbf{B}\} \\ \mathbf{A} &:= \mathbf{A} + \mathbf{x}_n \mathbf{x}_n' \\ \mathbf{B} &:= \mathbf{B} + \mathbf{x}_n y_n \end{aligned} \tag{11}$$

To obtain the regression coefficients, β, one simply computes $\beta = \mathbf{A}^{-1}\mathbf{B}$. This method is well known in the parallel computing literature and is, for example, described in Chu et al. (2007).

Although fairly simple, computing the regression coefficients this way has a disadvantage: Every time that β is computed, a matrix inversion is required. Especially when the number of independent variables q is large, this itself can be a computationally intensive operation. We can address this by computing the regression coefficients, β, directly online, using the

Sherman–Morrison formula (Escobar and Moser, 1993; Plackett, 1950; Sherman and Morrison, 1950):

$$\begin{aligned} \theta &= \{\mathbf{A}_{inv}, \mathbf{B}\} \\ \mathbf{A}_{inv} &:= \mathbf{A}_{inv} - \frac{\mathbf{A}_{inv}\mathbf{x}_n\mathbf{x}_n'\mathbf{A}_{inv}}{1+\mathbf{x}_n'\mathbf{A}_{inv}\mathbf{x}_n} \\ \mathbf{B} &:= \mathbf{B} + \mathbf{x}_n y_n, \end{aligned} \tag{12}$$

where $\mathbf{A}_{inv}$ is the inverted matrix $\mathbf{A}$. Using this formulation we directly update the inverted matrix. In practice one would use a small part of the data to create matrix $\mathbf{A}$, invert this matrix, after which the original matrix $\mathbf{A}$ can be discarded from computer memory. The "small" part of the data that is used should at least have $n > p$ for $\mathbf{A}$ to be non-singular. Obtaining the regression coefficients using Equation 12 is simple and now requires only a matrix multiplication: $\beta = \mathbf{A}_{inv}\mathbf{B}$. In Appendix B, we implement online linear regression in [R] using Sherman-Morrison formula.

To illustrate the difference between online and offline methods, Figure 1 presents a comparison of the computational time required to compute, as n increases, the regression coefficients β in a data stream between the three estimation methods discussed above. While the scale of the y-axis (time) will heavily depend on the size of the model (the number of parameters p) and the type of computing system used, the qualitative results presented here will hold in general: the computational time needed to obtain an estimate of β, at each value of n, will grow quite quickly (quadratic) for the offline method, while it grows only slowly for the online methods (linear). This result clearly illustrates the computational benefits of online methods over offline methods. It can also be seen that the direct online computation of the inverted matrix $\mathbf{A}_{inv}$ is faster than inverting the matrix at each time-point. This latter difference however only affects the slope of the linear computation time.

— insert Figure 1 here —

3.5 Effect size η^2 (ANOVA)

In many studies in sociology and psychology it is of interest to examine whether k distinct groups differ from one another, for instance because one group of participants received a treatment while the other group of participants did not. When such experiments are carried out using modern interactive technologies, such as on social media platforms, sample sizes can very quickly grow. Traditionally researchers often analyze the data from such group comparisons using an ANOVA approach. Between-subjects ANOVAs can easily be computed fully online. Here we focus on the computation of the effects size η^2 which is given by:

$$\begin{aligned}\eta^2 =& \frac{SS_b}{SS_b + SS_w}, \\ =& \frac{SS_b}{SS_t}, \\ =& 1 - \frac{SS_w}{SS_t},\end{aligned} \tag{13}$$

where SS_b equals the sum of squares *between* the k groups, SS_w is the sum of the sums of squares *within* each of the k groups, and SS_t is the total sum of squares. The last expression of Equation 13 shows that computing both SS_w and SS_t in a stream suffices to compute the desired effect size.

Equation 6 already presented how sums of squares can be computed in data streams. The only complexity introduced in the ANOVA example is the computation of the sums of squares within each of the k groups. This requires computing the average within group k:

$$\bar{x}_k := \bar{x}_k + \frac{x_{k,n} - \bar{x}_k}{n_k}. \tag{14}$$

which is only updated once a data point $x_{k,n}$ originates from group k. Subsequently, Equation 6 is used within each group k, substituting $\bar{x}$ with $\bar{x}_k$ to compute SS_w. The computation of the effect size or proportion of variance explained (η^2) in a data stream thus requires the following parameters:

$$\theta := \left\{ \begin{array}{r} \bar{x}_k, n_k \\ \\ \bar{x}, n, SS_t, SS_w \end{array} \right\}, \tag{15}$$

where the top row of θ indicates the parameters at the *group* level (and hence need to be kept in memory for each group k) and the bottom row indicates the global parameters, which are only single parameters which need to be stored in memory. Thus, in total θ contains $2k+4$ elements.

In order to compute the F-statistic online, one can simply use the information that is already available:

$$F = \frac{\eta^2}{(1-\eta^2)/(n-2)}. \tag{16}$$

It is important to note that repeated testing until a certain small p-value is found, which might be attractive if results are available for each new data point, is considered a questionable research practice (John et al., 2012), due to inflation of Type 1 error. For instance, when a researchers tests whether the ANOVA is significant 10 times with significance level of 5% while new data are entering, the actual Type 1 error equals

$$\alpha = 1-(1-0.05)^{10} = 0.401$$

instead of the 5% she started with.

We will continue our discussion of online learning methods, by discussing Stochastic Gradient Descent (SGD). SGD is an optimization method which is extremely useful to estimate more complex models when analytical solution are not easily available.

4 Online Estimation using Stochastic Gradient Descent

In the previous section we have discussed how to estimate, fully online, a number of statistics and models that are often used in the analysis of sociological and psychological data. We have also illustrated the computational advantages of online estimation for very large datasets and data streams. However, for each of the methods discussed above it was relatively easy to derive exact summation methods; using simple algebra it was possible to transform standard estimation methods to online variants. Unfortunately, this is not always the case. Many estimation methods, especially those that require multiple iterations through a static dataset, cannot exactly be implemented online, in part because even when using conventional offline analysis the estimation is

approximate. However, this does not mean that we can only estimate very simple models online: there is a multitude of methods available for the online estimation of more complex models. In this section we focus on Stochastic Gradient Descent (SGD), a general online estimation method that can be used for estimation of more complex statistical models.

To explain SGD, we will first discuss Gradient Descent (GD), an optimization method that is often used in conventional offline analysis and provides a logical starting point for SGD. We provide a general intuition to GD / SGD, and subsequently provide the technical details. Finally, we will provide an applied example of fitting a logistic regression model using SGD, for which [R] code is provided in Appendix C.

4.1 Offline Gradient Descent

There are multiple ways to obtain estimates for parameters of statistical models, for instance using a least squares approach (see, for example, Bretscher, 1995), using Maximum Likelihood estimation (ML), or using the method of moments (e.g., Arvas and Sevgi, 2012). In the social sciences we often use the maximum likelihood framework (see, for an introduction, Myung, 2003). In such cases we formalize the likelihood of the data as a function of the model parameters, and we are seeking for values of the parameters that maximize the likelihood function. Assuming independent observations, the likelihood of the data for many models of interest takes the following form:

$$\mathscr{L}(\zeta|x_1,\ldots,x_n) = \prod_{i=1}^{n} f(x_i|\zeta), \tag{17}$$

where ζ is a set of parameters, $f()$ is a probability density function (PDF) (or probability mass function in the discrete case), and as before $x_1,\ldots,x_n$ denote the observations. In words, Equation 17 states that the likelihood of the data is obtained by multiplying the individual likelihoods of each of the data points. In practice it is often (much) simpler to obtain the maximum likelihood estimates by taking the logarithm of the likelihood:

$$\ell(\zeta|x_1,\ldots,x_n) = \sum_{i=1}^{n} \ln f(x_i|\zeta), \tag{18}$$

which effectively replaces the product term with a sum, and gives the same solution for the maximum since the logarithm is a monotonic function. For some models, obtaining a maximum likelihood estimate analytically after the log-likelihood is defined is straightforward: we simply take the derivative of the log-likelihood and set it to 0 and solve for the parameters to obtain the required estimates. Effectively this has already been demonstrated: the estimation of the sample mean, and of linear regression models as discussed in previous sections, actually *are* analytical maximum likelihood estimates given the appropriate models (see for example, Gelman, 2007).

However, exact analytical solutions are not always easy to find and are sometimes simply unavailable. In such cases one can resort to *approximate* methods, which are also frequently applied in offline analysis. One such approximate algorithm is called Gradient Descent, or actually Gradient *Ascent* because we use it in the context of a likelihood function which we want to *maximize*. Gradient Descent is the name most often used in the machine learning literature and classically used to *minimize* the error.

The GD algorithm can be stated as follows:

$$\zeta := \zeta + \lambda \nabla \ell(\zeta | x_1, \ldots, x_n), \tag{19}$$

where λ is a learn rate (also known as step size) chosen by the researcher and $\nabla l()$ denotes the *gradient* (vector of first order derivatives) of the log-likelihood function. Intuitively this algorithm states that one picks a starting value of the parameters, ζ, and evaluates the gradient at this value. In the simple case where ζ is scalar this evaluation gives information regarding the slope of the log-likelihood function: if the slope is positive[1] then the maximum can be found at higher values of ζ and we can make a step towards higher values of ζ. If the slope at ζ is negative, we need to step in the opposite direction: we need to choose a lower value. Using this intuition, GD iteratively – passing through the dataset multiple times – takes steps towards the maximum of the log-likelihood function. In the case that ζ is a vector, GD takes a step in q dimensions: for each parameter (i.e., dimension) GD determines whether the slope of the

[1] Here we are assuming the log-likelihood function to be well-behaved.

derivative is positive or negative, accordingly GD takes a step in the q dimensional space which causes the steepest ascent towards the maximum of the likelihood function.

Gradient Descent can be a very effective method of finding the maximum likelihood value of ζ, although it is not without difficulties. For example, the parameter λ controls the size of the steps and has to be chosen carefully: a learn rate which is too large can be problematic since the algorithm could make jumps over the maximum likelihood solution. A learn rate which is too small causes the algorithm take very small steps, and thus many iterations will be needed to obtain the maximum likelihood estimate. It depends on the model (e.g., complexity of the model, complexity of the likelihood function, etc.) what learn rate will be appropriate. One can choose for either a *fixed* learn rate or a learn rate which is adaptive, for instance one could choose to let the learn rate decrease with the number of iterations. A more extensive discussion on choosing the appropriate learn rate for complex models can be found in Wilson and Martinez (2003).

4.2 Online or Stochastic Gradient Descent

Gradient Descent provides an iterative, approximate method to find maximum likelihood estimates. It is surprisingly simple to derive an effective *online* version of GD: instead of iterating over the full dataset multiple times and updating ζ each iteration, we simply take a small step to a more likely parameter value every time a data point enters:

$$\zeta := \zeta + \lambda \nabla \ell_n(\zeta | x_n), \tag{20}$$

where we use ℓ_n to denote that we are evaluating the log-likelihood with respect to a single data point. Hence, instead of updating our parameters based on iterations through all of the data, we update based on each arriving data point. SGD will converge to an unbiased estimate of the parameters as long as the order in which the data points arrive is random (Bottou, 2010). This means that the process that generated the data, does not change over the period in which the data are arriving.

Note that in the case that the dataset is no longer augmented, SGD can still be a useful tool: Analyzing static Big Data using SGD circumvents that the entire dataset needs to be available in memory. By simulating that the data enter a point at a time and letting the data stream in repeatedly, SGD can obtain unbiased estimates while still estimate the parameters without seeing all the data at once.

4.3 Logistic regression: an Example of the Usage of SGD

We present an example to illustrate SGD in which we are interested in the effect of independent variables on a binary dependent variable. In applied research dependent variables are often binary, examples include whether and how people intent to vote (democratic versus republican Anderson, 2000), or whether or not people smoke cigarettes (Emmons et al., 1998). In the case of a binary dependent variable, often a logistic regression model is chosen to describe the relationship between a binary dependent variable and continuous independent variables:

$$Pr(y = 1|\mathbf{X}) = p(\mathbf{X}) = \frac{\exp(\beta\mathbf{X})}{1+\exp(\beta\mathbf{X})}, \tag{21}$$

where p is the probability to score a 1 on y, $\Pr(y = 1|\mathbf{X})$ and is understood to be a function of the dependent variables X. Unlike linear regression, logistic regression does not have a closed form solution to estimate the parameters β using a maximum likelihood approach, and hence even for offline analysis approximate methods are used. Estimating the parameters online can be done using SGD as follows. First, we specify the log-likelihood:

$$\ell(\beta) = \sum_{i=1}^{n} y_i \log p_i(\mathbf{x}_i) + (1-y_i) \log 1 - p_i(\mathbf{x}_i) \tag{22}$$

Second, we compute the gradient (see, for more details for instance Agresti, 2002):

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{n} (y_i - p_i(\mathbf{x}_i))\mathbf{x}_i, \tag{23}$$

which in the case of offline estimation would be evaluated for all the data at once. When we use SGD to estimate the β's, the following online algorithm is obtained:

$$\begin{aligned} \theta =& \{\beta, \lambda\} \\ \lambda :=& \lambda + f(\lambda, x_n) \\ \beta :=& \beta + \lambda(y_n - p(\mathbf{x}_n))\mathbf{x}_n. \end{aligned} \tag{24}$$

Here we include λ, the learn rate, in θ. This will not be necessary for a fixed value of λ, but it highlights that the learn rate could be a function of the data stream. Given an appropriate choice of λ, and a large enough data stream, SGD will correctly estimate the parameters of interest (Bottou, 2010). See Appendix C for an implementation of SGD for the estimation of logistic regression in [R].

4.3.1 Using online learning in practice: fitting logistic regression in a data stream

To illustrate a logistic regression in a data stream, we use an example dataset, which is used and fully described in Gelman (2007). The dataset contains information regarding households in Bangladesh and whether or not they switch to a safe well to collect water. Among other variables, the dataset includes the distance in meters to a safe well (X_{dist}) from the household, and arsenic level that is present in the water (X_{ars}). The relatively small dataset consists of $N = 3020$ households. We simulate that the data enter a point at a time by analyzing the data row-by-row using both offline and online implementations to predict whether the household switched to a safe well (coded 1) or did not switch (coded 0). The model we estimate contains two independent variables and an interaction term:

$$Pr(y = 1|X_{dist}, X_{ars}) = \frac{\exp(b_0 + b_1 X_{dist} + b_2 X_{ars} + b_3 X_{dist} X_{ars})}{1 + \exp(b_0 + b_1 X_{dist} + b_2 X_{ars} + b_3 X_{dist} X_{ars})}, \tag{25}$$

We thus estimate the four coefficients b_0, b_1, b_2, and b_3. Here θ includes n because we choose an adaptive learn rate. Equation 26 presents the elements of θ and their respective update Equations

during the data stream. The starting values for all four β's equalled 0.

$$\begin{aligned}\theta =&\{\beta, n\}\\ n :=&n+1\\ \lambda =&\frac{1}{\sqrt{n}}\\ \beta :=&\beta+\lambda\left(y_n-\frac{\exp(\beta\mathbf{x}_n)}{1+\exp(\beta\mathbf{x}_n)}\right)\mathbf{x}_n.\end{aligned} \tag{26}$$

In Figure 2 we present the results of fitting a logistic regression in a data stream with four coefficients and an adaptive learn rate, $\lambda = \frac{1}{\sqrt{n}}$. The estimates of the effect of the arsenic level (b_2) and the interaction term (b_3) are very accurate from the beginning of the data stream. The estimates of the intercept (b_0) and the effect of the distance to the next safe well (b_1) require some more data. The dashed line is fluctuating, even towards the end of the dataset: this is due to the fact that the learn rate is still quite substantial ($\frac{1}{\sqrt{3020}} = 0.018$) for a dataset this size. A smaller learn rate (or one that decreases more rapidly) would stabilize the SGD algorithm more, but do so at the risk of introducing more bias. For larger datasets (and for continuous streams, which is what we primarily focused on in this paper, the performance of SGD is often surprisingly accurate.

— insert Figure 2 here —

5 Some Considerations analyzing Big Data

In this paper we have discussed online learning as a way to deal with Big Data. However, some important issues remain. Here we discuss two practical and two conceptual issues related to analyzing Big Data.

Practically, it has to be noted that at this moment not many off-the-shelf statistical packages are available to actually analyze Big Data in a streaming fashion. The currently available software, for instance *RStorm* (Kaptein, 2014), *Apache Spark* (Karau et al., 2015), and *S4* (Neumeyer et al., 2010) often require extensive programming knowledge. There is still a large

gap between the methods and software developed by computer scientists, and those that can be used by social scientist to analyze their data streams using models that they are accustomed to.

Second, we have to stress that for the application of online methods the analyst has to know beforehand what type of analysis and model is required to answer the research question. Online learning methods make use of a limited set of quantities – referred to the elements of θ throughout this text – to store the relevant information and to subsequently estimate model parameters. This means that it is important to know what information is required *before* the analysis. Any information that is not stored is forgotten and is impossible to retrieve if the data themselves are not stored.

A solution to this latter issue could be to run simultaneously different analyses and/or models, such that at a later point in time a decision can be made which analysis or model to use. This, of course, does require that enough computer memory is available to store the sufficient statistics of multiple models. A frequently adopted practical solution to this in the computer science literature is to adopt a so-called λ-architecture (Marz and Warren, 2013): the data stream is operated on online (for those computations that where specified in advance), but also stored and can thus be analyzed offline at a later point in time (often using parallelization methods to deal with the size of the dataset).

Conceptually, we need to stress that we are not promoting repeated null hypothesis significance testing in data streams; this should be avoided because it results in Type I error inflation (i.e., too many false positives, Strube, 2006). It is considered a questionable research practice to repeatedly test for a significant effect and stop data gathering once the effect yields a $p < .05$ (John et al., 2012). When adopting an online learning approach we encourage researchers to focus on obtaining precise estimates of the size of the effects of interest, in adherence to the APA guidelines (Affairs and the Task Force on Statistical Inference, 1999), as opposed to null hypothesis testing.

Finally, it is not always feasible to translate all analyses from the offline framework to the online framework. For instance, the analysis of dependent data, data that are nested within units,

which are in the offline case often analyzed using multilevel (or random effects-) models, have not yet found a proper online synonym. Therefore future research should be aimed at translating complex models, such as the multilevel models, to the online learning framework. Note that active research work is carried out in this field, with for example recent publications describing online approximations of the well-known Expectation-Maximization algorithm (Cappé and Moulines, 2009).

6 Discussion

Using new data collection methods and technologies, for instance experience sampling (Barrett and Barrett, 2001), to collect social and psychological data have made data streams more apparent and more prevalent in recent years. In this paper we discussed how social scientists can deal with these large datasets, and how regular estimation methods can be applied in the context of continuous data streams. We hope to have contributed to opening up the possibilities to answer both existing research questions as well as new types of research questions using large datasets or continuous data streams. We do want to stress that we have only touched upon a few methods which are used to analyze data streams; there are many more techniques available to analyze data streams, for instance the Bayesian framework can in some cases also be used to update the estimated parameters (using Bayesian updating of the posterior, e.g., Gelman et al., 2004).

There is however a clear need for more transformations of methods used in offline analysis to online methods that can be used in data streams. Further development of estimation methods and easy to use software would make the analysis of Big Data more easily available for applied researchers. For instance, currently methods such as (latent) factor analysis, mixture models, or multilevel models are difficult to fit in a data stream. A possible way to deal with these types of analyses is to alter for instance the EM algorithm (Dempster et al., 1977). Suggestions for parallel computations and more efficient procedures for the EM algorithm have already been proposed (Cappé and Moulines, 2009; Neal and Hinton, 1998; Wolfe et al., 2008), and this work should be extended to make the EM algorithm applicable for streaming data.

We hope that the current article motivates applied researchers to explore new research areas that are opened up by the technological opportunity to monitor individuals in a data stream. We believe that data streams can provide social scientists with many new insights in human behavior and can provide new research areas to study human emotions and attitudes.

Bibliography

Affairs, L. W. and the Task Force on Statistical Inference (1999). Statistical methods in psychology journals. *American psychologist*, 54(8):594–604.

Agresti, A. (2002). *Second Edition :*. Wiley series in probability and statistics, Gainesville, Florida, 2nd edition.

Anderson, C. J. (2000). Economic voting and political context: a comparative perspective. *Electoral Studies*, 19(2-3):151–170.

Arvas, E. and Sevgi, L. (2012). A tutorial on the method of moments. *Antennas and Propagation Magazine, IEEE*, 54(3):260–275.

Atallah, M. J., Cole, R., and Goodrich, M. T. (1989). Cascading Divide-and-Conquer : A Technique for Designing Parallel Algorithms. *SIAM Journal on Computing*, 18(3):499–532.

Barrett, L. F. and Barrett, D. J. (2001). An Introduction to Computerized Experience Sampling in Psychology. *Social Science Computer Review*, 19(2):175–185.

Bottou, L. (1998). Online learning and stochastic approximations. *On-line learning in neural networks*, pages 1–34.

Bottou, L. (2010). Large-Scale Machine Learning with Stochastic Gradient Descent. In *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187.

Bretscher, O. (1995). *Linear Algebra with Applications*. Pearson, 4th edition.

Cappé, O. and Moulines, E. (2009). Online expectation-maximization algorithm for latent data models. *Journal of the Royal Statistics Society: Series B (Statistical Methodology)*, 71(3):593–613.

Carmona, C. J., Ramírez-Gallego, S., Torres, F., Bernal, E., Del Jesus, M. J., and García, S. (2012). Web usage mining to improve the design of an e-commerce website: OrOliveSur.com. *Expert Systems with Applications*, 39(12):11243–11249.

Chu, C., Kim, S. K., Lin, Y., and Ng, A. Y. (2007). Map-Reduce for Machine Learning on Multicore. In Schölkopf, B., Platt, J. C., and Hoffman, T., editors, *Advances in Neural Information Processing Systems*, volume 19, page 281. Massachusetts Institute of Technology, 19 edition.

Datar, M., Gionis, A., Indyk, P., and Motwani, R. (2002). Maintaining Stream Statistics over Sliding Windows. *SIAM Journal on Computing*, 31(6):1794–1813.

Demchenko, Y., Grosso, P., De Laat, C., and Membrey, P. (2013). Addressing big data issues in Scientific Data Infrastructure. *Proceedings of the 2013 International Conference on Collaboration Technologies and Systems, CTS 2013*, pages 48–55.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Efraimidis, P. S. and Spirakis, P. G. (2006). Weighted random sampling with a reservoir. *Information Processing Letters*, 97(5):181–185.

Emmons, K. M., Wechsler, H., Dowdall, G., and Abraham, M. (1998). Predictors of smoking among US college students. *American journal of public health*, 88(1):104–7.

Escobar, L. and Moser, E. (1993). A Note on the Updating of Regression Estimates. *The American Statistician*, 47(3):192–194.

Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S. (2005). Mining Data Streams : A Review. *SIGMOD*, 34(2):18–26.

Gelman, A. (2007). Rich State, Poor State, Red State, Blue State: What's the Matter with Connecticut? *Quarterly Journal of Political Science*, 2(4):345–367.

Gelman, A., Carlin, J., Stern, H., and Rubin, D. (2004). *Bayesian Data Analysis*. Chapman&Hall/CRC, 2nd editio edition.

John, L. K., Loewenstein, G., and Prelec, D. (2012). Measuring the Prevalence of Questionable Research Practices With Incentives for Truth Telling. *Psychological Science*, 23(5):524–532.

Kaptein, M. (2014). {RStorm}: Developing and Testing Streaming Algorithms in {R}. *The R Journal*, 6(1):123–132.

Karau, H., Konwinski, A., Wendell, P., and Zaharia, M. (2015). *Learning Spark*. O'Reilly Media.

Killingsworth, M. A. and Gilbert, D. T. (2010). A Wandering Mind Is an Unhappy Mind. *Science*, 330(6006):932.

Marz, N. and Warren, J. (2013). *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications.

Myung, I. (2003). Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, 47:90–100.

Neal, R. and Hinton, G. E. (1998). A View Of The Em Algorithm That Justifies Incremental, Sparse, And Other Variants. *Learning in Graphical Models*, pages 355–368.

Neumeyer, L., Robbins, B., Nair, A., and Kesari, A. (2010). S4: Distributed stream computing platform. *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 170–177.

Opper, M. (1998). A Bayesian Approach to Online Learning. In Saad, D., editor, *On-Line Learning in Neural Networks*, pages 363–378. Cambridge University Press, Cambridge.

Pébay, P. (2008). Formulas for Robust, One-Pass Parallel Computation of Covariances and Arbitrary-Order Statistical Moments. *Sandia Report*, SAND2008-6(September):1–18.

Plackett, R. (1950). Some Theorems in Least Squares. *Biometrika*, 37:149–157.

Sagiroglu, S. and Sinanc, D. (2013). Big data: A review. *International Conference on Collaboration Technologies and Systems (CTS)*, pages 42–47.

Shalev-Shwartz, S. (2011). Online Learning and Online Convex Optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194.

Sherman, J. and Morrison, W. J. (1950). Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix. *The Annals of Mathematical Statistics*, 21(1):124–127.

Strube, M. J. (2006). SNOOP:a program for demonstrating the consequences of premature and repeated null hypothesis testing. *Behavior research methods*, 38(1):24–27.

Swendsen, J., Ben-Zeev, D., and Granholm, E. (2011). Real-time electronic ambulatory monitoring of substance use and symptom expression in schizophrenia. *Am J Psychiatry*, 168(2):202–209.

Turaga, D., Andrade, H., Venkatramani, C., Verscheure, O., Harris, J. D., Cox, J., Szewczyk, W., and Jones, P. (2010). Design principles for developing stream processing applications. (August):1073–1104.

Welford, B. (1962). Note on a Method for Calculating Corrected Sums of Squares and Products. *Technometrics*, 4(3):419–420.

Whalen, C. K., Jamner, L. D., Henker, B., Delfino, R. J., and Lozano, J. M. (2014). The ADHD spectrum and everyday life: experience sampling of adolescent moods, activities, smoking, and drinking. *Child development*, 73(1):209–27.

Wilson, D. and Martinez, T. R. (2003). The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429–1451.

Wolfe, J., Haghighi, A., and Klein, D. (2008). Fully distributed EM for very large datasets. *Proceedings of the 25th international conference on Machine learning - ICML '08*, pages 1184–1191.

Appendix A: Online correlation

```
> N <- 1000                                  #number of observations
> x <- rnorm(N, 5,2)                         #generate data
> y <- 1.5*x+rnorm(N)
> # because a correlation requires at least 2 points we start with n=1
> n = 1; xbar = x[1]; ybar = y[1]; SC = 0; SSx = 0; SSy = 0;
> for (i in 2:N)
+ {
+    dx   <- (x[i]-xbar)                     #deviance x
+    dy   <- (y[i]-ybar)                     #deviance y
+    n    <- n+1                             #update number of observations
+    xbar <- xbar+(x[i]-xbar)/n              #update mean x
+    SSx  <- SSx+dx*(x[i]-xbar)              #update sum of squares for x
+    SC   <- SC+(x[i]-xbar)*(y[i]-ybar)      #update sum of cross products
+    Sxy  <- SC/(n-1)                        #compute covariance
+    ybar <- ybar+(y[i]-ybar)/n              #update mean y
+    SSy  <- SSy+dy*(y[i]-ybar)              #update sum of squares for y
+    sx   <- sqrt(SSx/(n-1))                 #estimate std.dev. x
+    sy   <- sqrt(SSy/(n-1))                 #estimate std.dev. y
+    rxy  <- Sxy/(sx*sy)                     #estimate correlation
+ }
>
```

Appendix B: Online linear regression

```
> N  <- 1000    # generate data
> x0 <- rep(1, N)
> x1 <- rnorm(N, 5,2)
> x  <- matrix(c(x0,x1),nrow=N)
> y  <- 3+1.5*x[,2]+rnorm(N)
> A  <- matrix(0,nrow=2,ncol=2); B <- c(0,0)
> #the as.matrix and as.numeric are required to get [r] running
> for (i in 1:N)
+ {             #fit linear regression:
+    if(i<3)
+    {          #update A as long as it is not invertible
+      A     <- A+x[i,]%*%t(x[i,])
+    }          #update B
+    B       <- B + as.matrix(x[i,])%*%y[i]
+    if(i==3)
+    {          #invert A when n>p
+      A_inv <- solve(A)
+    }
+    if(i>=3)   #update inverted matrix A_inv
+    {          #C is a scalar
+      C     <- as.numeric((1+x[i,]%*%A_inv%*%x[i,]))
+      A_inv <- A_inv - ((A_inv%*%x[i,]%*%x[i,]%*%A_inv)/C)
+      beta  <- A_inv%*%B #compute coefficients
+    }
+ }
>
```

Appendix C: Logistic regression using Stochastic Gradient Descent

```
> N    <-3000
> x    <-rnorm(N,1,1)
> e    <-rnorm(N)
> y    <-rbinom(N,1, (exp(-2+1.5*x+e)/(1+exp(-2+1.5*x+e))))
> beta <- c(0,0)
> for(i in 1:N)
+ {
+    p    <- exp(beta[1]+beta[2]*x[i])/(1+exp(beta[1]+beta[2]*x[i]))
+    beta <-beta + lambda*(y[i]- p) %*%c(1,x[i])
+ }
>
```

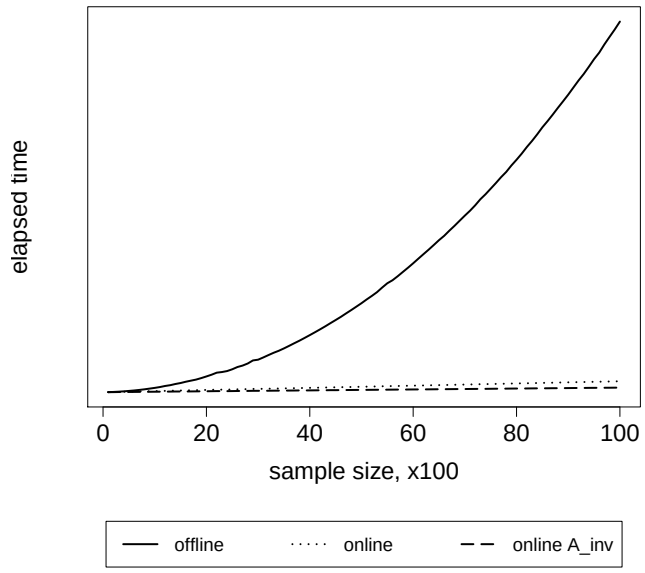

Figure 1: Computation time of regression coefficients using offline estimation (solid line), online estimation by inverting the matrix (online A_{inv}, dotted line), or online estimation by online updating the inverted matrix (dashed line).

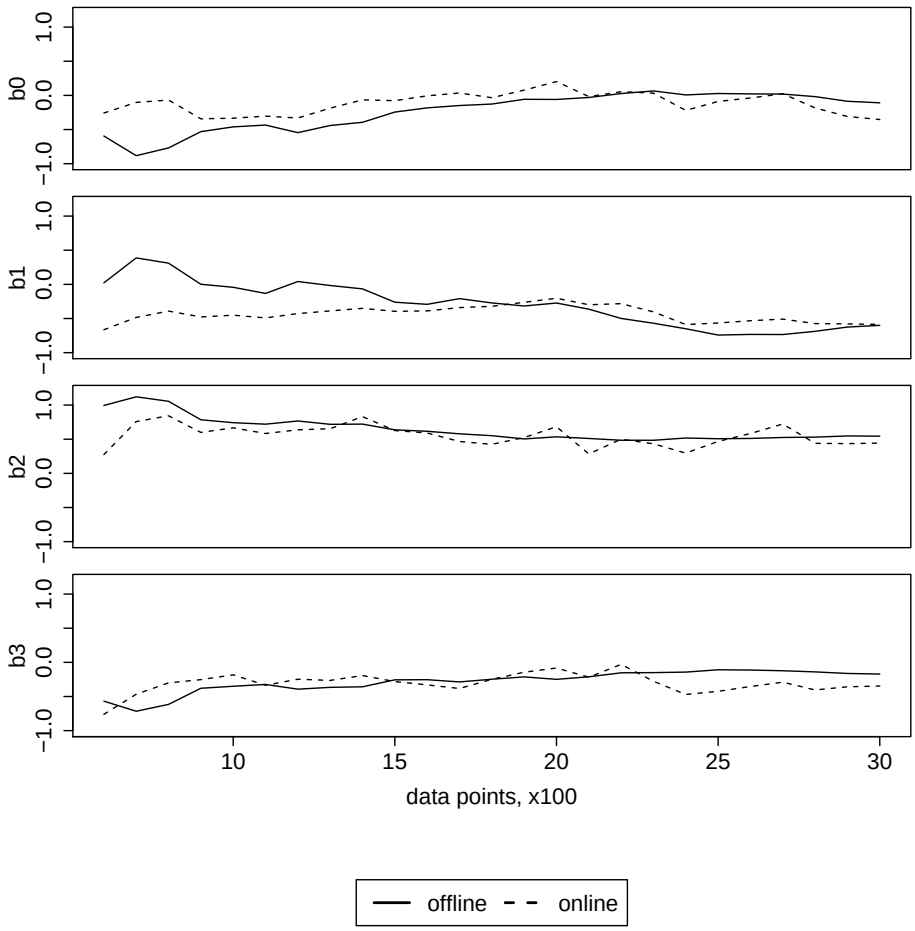

Figure 2: Online (dotted) and offline (solid) estimated beta coefficients of logistic regression as more data enter.