Energy Efficient Knowledge Distribution for Query Analytics

Kurt Portelli

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

A dissertation presented in part fulfilment of the requirements of the Degree of Master of Science at The University of Glasgow

July 11, 2016

EU funds for Malta 2014 2020

The research work disclosed in this publication may be partially funded by the Endeavour Scholarship Scheme (Malta). Scholarships may be considered for part-financing by the European Union - European Social Fund (ESF) - Operational Programme II – Cohesion Policy 2014-2020 *"Investing in human capital to create more opportunities and promote the well-being of society"*.

European Union – European Structural and Investment Funds
Operational Programme II – Cohesion Policy 2014-2020
"Investing in human capital to create more opportunities and promote the well-being of society"
Scholarships may be considered for part-financing by the European Union - European Social Funds (ESF)
Co-financing rate: 80% EU Funds;20% National Funds

Abstract

abstract goes here

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name: Kurt Portelli Signature: K.Portelli

Acknowledgements

I wish to express my sincere gratitude to Dr. Christos Anagnostopoulos for his guidance and expert advice throughout all the various stages of this project. Apart from that I am also thankful to him for making me appreciate even more the subject of data science and machine learning.

Furthermore, I would like to thank my family for all their continuous support in this first year living abroad.

Contents

Chapter 1

Introduction

The Internet of Things (IoT) is a rapidly growing area, with it more data is being stored and made accessible which previously has never even been thought possible. IoT is composed of billions of devices that can gather, share information and sometimes even capable of performing operations on this information. IoT devices are physical objects connected with the internet able to share information. This can include anything ranging from smartphones to Radio Frequency Identification (RFID) tags found in everyday products. As more data is made available for anything imaginable that affects our daily lives, opportunities arise for applications that extract this information and make use of it. EXAMPLE SMARTGRID AND REFERENCE. Making all this information useful is very challenging as it needs to be collected, transferred, stored and made accessible.

In a naive system all these devices generate massive amounts of data which is periodically sent to central node with more computing resources. This in turn, might restructure the data more effectively to be sent to another node or make it available for querying. All this network transfer drains power and as the network size increases this effect is emphasized even more. This has created a need and opportunity for machine and statistical learning algorithms.[3] According to L.Bottou et al.[3] these technological improvements have outrun the exponential evolution of computing power. Thus, we must rely more on on-line learning algorithms to process these large amounts of data with comparatively less computing power.

The vision of IoT is to allow any device to be continuously connected sharing its knowledge about the surroundings. Machine learning then uses this stream of data to deduce results, infer new knowledge, reason about contextual events or make assumptions. The possibilities for such systems are endless, taking, for instance, the case of thermometers in rooms next to each other and based on previous information the temperature of various rooms can be predicted based on the temperature of adjacent ones.

1.1 Problem Synopsis

1.2 Project Aims

1.3 Document Outline

Chapter 2

Preliminaries

In this chapter, we present an overview of the concepts and base principles we mention and make use of in our work. We develop a system which deals with a continuous data stream and thus, has to use specialized algorithms which treat each data item individually without storing past data items. Various algorithms are used which extract knowledge from each individual sensor by minimizing loss and then quantize the input.

2.1 On-line Algorithms

According to M.Karp [6] an on-line algorithms is one which receives a sequence of requests and performs an immediate action to each request. These are useful in situations where decisions must be made without any future knowledge. Their effectiveness is measured by the comparing them with their off-line (batch) counterparts. In this dissertation we make use of well known common techniques already proven to work.

2.1.1 K-Means

The K-means algorithm attempts to find K representatives (centroids) of a data set in such a way as to best represent the data. [2] Although it is algorithmically simple, it still manages to be relatively robust and give close to optimal results on a wide range of data sets. Its disadvantage is that one has to predefine K. Furthermore, the initial K centroids chosen have a drastic effect upon the clusters chosen. The first k inputs initialize the codebook vectors (centroids) $m_j, j = 1, ..., k$. The rest of the inputs $x^t \in X$ are used for the update function.

$$i \longleftarrow argmin_j \parallel x^t - m_j \parallel \tag{2.1}$$

$$m_i \longleftarrow m_i + \eta(x^t - m_i) \tag{2.2}$$

Equation 2.1 finds the closest centroid to the input. Then the closest centroid is updated as shown in equation 2.2 depending on the learning rate defined by η. A large η enhances accuracy but makes m_i oscillate. Therefore to reach convergence one needs to use a small or decaying η.

2.1.2 Adaptive Resonance Theory (ART)

ART[4] is another unsupervised learning algorithm similar to k-means. It does not require the number of clusters to be predefined before hand. Instead one has to define *vigilance* which represents the degree of similarity between points. An data item $x^t \in X$ is a member of a cluster m_j only if the distance between them is less than vigilance. Similar to the K=means procedure, once m_j is chosen, equation eq:updateCluster is used to update the centroid. If no centroid satisfies this constraint, then a new cluster is created. This allows a better flexibility but has the disadvantage of creating an infeasible and unpredictable amount of clusters in certain situations.

2.1.3 Linear Regression

Once it is proven that there is a statistically significant correlation between two variables, linear regression allows us to make predictions about one variable based only on the other variable. The relationship between these two values must be linear, else this technique would not be able to model the data accurately.

To perform this supervised learning we must first start by defining the hypothesis example the linear function 2.3.

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \tag{2.3}$$

The θ_i's represent the weights of the linear function, thus, determining the mapping between X and Y. Let $x_0 = 1$, we then rewrite equation 2.3 to a more compact form represented by equation 2.4.

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x \tag{2.4}$$

For this hypothesis to be useful, the θ parameters need to be trained to fit a particular dataset. Conceptually the closest $h(x)$ is to Y, the better we are at representing the data. Thus, we define the cost function equation 2.5 which measures the total discrepancy for each input $x^{(i)}$ with output $y^{(i)}$.

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{2.5}$$

Naturally, we would like to choose a theta which minimizes the error. One such technique which does this is the gradient descent rule. It calculates the gradient of the error and takes a step proportional to the negative gradient. To achieve this we need to use the partial derivative of equation 2.5. For all θ the update rule then becomes equation 2.6. This update rule is called least mean squares (LMS) and it has a very interesting property. The magnitude of the update is proportional the the error. This means that the greater the error, the steeper the descent towards the minima. After completing the derivative we are left with the complete equation 2.7. This has to be repeated for each θ until convergence. Meaning until we reach the best possible theta values.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{2.6}$$

$$\text{Repeat until convergence: (For each j) } \theta_j := \theta_j - \alpha \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)} \tag{2.7}$$

One might argue that with this iterative approach we might get stuck at a local minima instead of finding the optimal minima. In this particular scenario J is a convex quadratic function, meaning it has only one minima. Thus, gradient descent will always converge at the global minima assuming α is not too large.

Multivariate Stochastic Gradient Descent

The previous algorithm is called batch gradient descent as for each update step it uses all all the training set. This would not be feasible to compute especially as the dataset size increases. Another issue would be that previous data items have to be stored, and this is not always possible. In fact we will be using the more lightweight and on-line version called stochastic gradient descent. In batch gradient descent we go through all the data set for a single update step while when using equation 2.8 we start making progress right away. We use this update rule for each new training item we encounter. This in fact has the tendency of getting close to θ much faster than batch gradient descent, however it may never converge and instead keep oscillating around the minimum. The reason it is called multivariate is that we will be using multiple inputs (more than one x) to get y.

$$\text{(For each j) } \theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)} \tag{2.8}$$

2.1.4 Mean and Variance

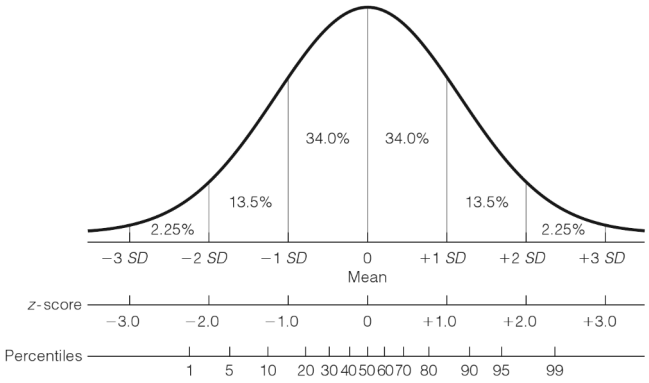
2.2 Probability Distribution Function

2.3 Normalization

We use the standard score [5] to normalize the dataset. This is done by subtracting each item x by the mean and then dividing it by the standard deviation of the dataset as shown in equation 2.9. Performing this normalization allows us to better understand the results as it gives a meaning to a raw value. Figure 2.1 demonstrates the distribution of z, a negative value means that x was less than the mean. Furthermore, the magnitude of this value represents the distance from the mean in terms of standard deviation.

$$z = \frac{x - \mu}{\sigma} \tag{2.9}$$

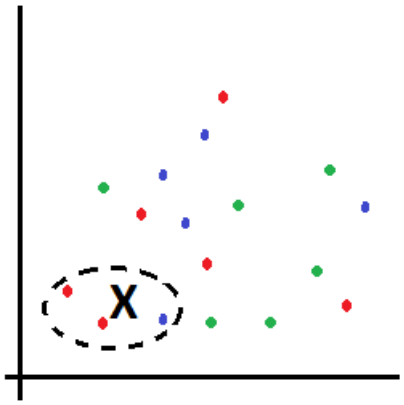
Figure 2.1: Standard score normalization [5]



2.4 K Nearest Neighbours (KNN)

KNN [1] is a straightforward classification technique, in which given an input it is able to classify it to certain class. As the name implies it takes the closest K neighbours and depending on the majority of their classes it determines the class of the input. Figure 2.2 demonstrates KNN ($K = 3$) while classifying X using a data set containing 3 different classes. In this case input X is classified as Red due to the fact that the majority of its 3 closest neighbours are Red.

Figure 2.2: At KNN=3, Input X is classified as Red



Chapter 3

Literature Review

Chapter 4

Contributions

4.1 Network Efficiency

4.2 Ensemble Learning

4.3 Optimal Stopping

Chapter 5

Design and Implementation

5.1 Network Architecture

5.2 Query Analytics

Chapter 6

Evaluation and Case Study

6.1 Data Set

6.2 Error Allowance vs Messages Sent

6.3 Probability Distribution Functions

6.4 Query Validation

Chapter 7

Conclusion

7.1 Achievements

7.2 Future Work

7.3 Final Remarks

Appendix A

First appendix

A.1 Section of first appendix

Appendix B

Second appendix

Bibliography

[1] N. S. Altman. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, 46(3):175–185, 1992.

[2] WESAM BARBAKH and COLIN FYFE. Online clustering algorithms. *International Journal of Neural Systems*, 18(03):185–194, 2008. PMID: 18595148.

[3] Lon Bottou and Yann LeCun. Large scale online learning. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schlkopf, editors, *NIPS*, pages 217–224. MIT Press, 2003.

[4] Gail A. Carpenter and Stephen Grossberg. *Adaptive Resonance Theory*, pages 22–35. Springer US, Boston, MA, 2010.

[5] R. Charles Fawcett University of Virginia Edward S. Neukrug Old Dominion University. *Essentials of Testing and Assessment: A Practical Guide for Counselors, Social Workers, and Psychologists, 3rd Edition*. Brooks Cole, 3 edition, 2015.

[6] Richard M. Karp. On-line algorithms versus off-line algorithms: How much is it worth to know the future? In *Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture - Information Processing '92, Volume 1 - Volume I*, pages 416–429, Amsterdam, The Netherlands, The Netherlands, 1992. North-Holland Publishing Co.