Computer Science in the Information Age

John E. Hopcroft, *Life Fellow, IEEE*, Kun He, *Member, IEEE*

***Abstract*—For fifty years, computer science was focused on making computers useful. The emphasis was on algorithms, programming languages, and operating systems. Today the focus has changed to applications. Some of the drivers are the amount of computing power available, the quantities of data, and the internet. The impact on computer science created an interest in applications and interdisciplinary research. In this article we will review some of the recent work in clustering in social networks, learning theory, the size of data, and the digitalization of records and the need for privacy in computer systems.**

***Index Terms*—Community clustering, deep neural networks, large graph, privacy and security**

I. Clustering

EARLY work in the area of clustering was involved in partitioning vertices of social networks into disjoint clusters. This work evolved into finding overlapping communities. More recently there have been two major advances. One is as the size of social networks increased to billions of vertices, global clustering was replaced by local clustering. In global clustering if one partitions a billion vertex graph into ten or so clusters, the clusters would have hundreds of millions of vertices. With recursive partitioning one may find clusters of size a hundred, but the process is inefficient. Overlapping community detection is also more costly and hard to scale to large networks. Instead one might want local clusters for the seed members of interest. Say a cluster of fifty friends of three or so designated individuals. One method for doing this is to use spectral clustering. In spectral clustering one creates a matrix whose columns are the first few spectral vectors of the adjacency matrix of the network and then finds the minimum one norm vector in the space spanned by the columns [1, 2].

Instead of finding the minimum one norm vector in the space spanned by the singular vectors one might start random walks from a few vertices in the local community, but halt the process when the walks have converged to the stationary probabilities for the vertices in the community but not for the whole graph. The minimum one norm vector in the space spanned by these unconverged vectors will give the local community.

This work was supported by US Army Research Office (W911NF-14-1-0477), and National Natural Science Foundation of China (61772219).

John E. Hopcroft is with the Department of Computer Science, Cornell University, Ithaca, NY 14853, USA (e-mail: jeh@cs.cornell.edu).

Kun He, is with the Department of Computer Science, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: brooklet60@hust.edu.cn).

Another direction in clustering is finding hidden structure [3, 4]. Suppose one had images of a number of letters and the letters were in several shades of gray and different type fonts. If you were asked to cluster the images you would probably cluster them by letter. However, one could cluster them by color or by type font. The letter is the dominant structure. The type font and color of letters are the hidden structures, which are weaker and incoherent with the dominant clustering. Real networks have both dominant and several levels of hidden structure. How do you find the hidden structure in a social network?

Select your favorite clustering algorithm and find the dominant structure. Then weaken the dominant structure in the graph by randomly removing edges in the clusters. The again apply your clustering algorithm to the graph and it will find the hidden structure. If you go back to the original graph and now weaken the hidden structure and again find the dominant structure you will probably improve the clustering algorithms. If you alternately weaken the dominant and then the hidden structure you will converge to good clusterings of both the dominant and the hidden structure. Some real world networks have several levels of hidden structure that can be retrieved.

Applying this technique to the Facebook data of Rice University students [5] one gets a dominant structure and three levels of hidden structure. The dominant structure is the dorm the student lives in and one of the hidden levels is the year of the student, freshman, sophomore, junior, or senior. The other two levels have high modularity and are incoherent with earlier levels but we were unable to identify what they corresponded to; maybe sports or other interests.

II. Deep Learning

Machine learning has been extremely valuable for a number of years. Its major tool was the support vector machine. However, in 2012 advances in deep learning changed the field. Until 2012 reducing the classification error in the ImageNet ILSVRC competition [6,7] was very small. ImageNet has 1.2 million images classified in 1,000 categories.

The task is to train a network on a training set of the images and see how well it generalizes to a test set. Prior to 2012 the error rate was approximately 25%. Then in 2012, AlexNet dropped the error rate to 15%, a truly major improvement. Two years later GoogleNet reduced the error rate to 6.7% and in 2015 ResNet reduced it further to 3.6%. The human error rate with training is about 5%. These deep networks outperform humans. Since 2012 deep networks have been applied in many applications and they have performed exceptionally well although little is known as to why they work so well.

One of the issues with supervised learning is the lack of sufficiently large labeled data sets. Recently there has been progress in unsupervised learning. Instead of training a deep network to classify images one can train the network to reproduce the image. This results in good internal representations of images and raises the question of what internal gates learn. This has significantly increased the research on unsupervised learning.

There are many research problems that might shed some insight into deep learning. Do some gates learn the same thing as other gates? If we train a network twice from different random starting positions do the gates learn the same things or does the network develop entirely different way to classify images [8]? What determines what a gate learns. How does what a gate learns evolve over time. Do gates in the first or second convolution levels learn features of images independent of what the images are? In training a deep network one encounters many local minima with the same error rate. Which local minima will generalize better? Experimentally broader local minima seem to generalize better. This may be because the error function for the training set is very close to the true error function and a small shift will not disturb a broad local minimum as much as it will on a sharp local minimum.

Generative adversarial networks [9] have become very popular. If one wants to generate realistic looking images one might train an adversarial network to distinguish between real images and generated images. Then they could feed the adversarial component the output of the generative component and train the generative component until the adversarial component could not distinguish between the generated image and a real image. At that point one trains the adversarial component to do better. By interacting with the two units one can generate good images.

Another application might be language translation. In the past one used pieces of text where one had the same text in both languages to train a network. But if one does not have sufficient samples in both languages they could use an adversarial network as follows. To create a translator from English to German one first build a translator that will take an English sentence and output German words. The one build an adversarial network that distinguishes between German words and German sentences. Finally, one takes the output of the first devices that outputs German words and builds a device that creates English sentences and compares the sentences generated to the original sentence. Training the three networks forces the output of the first device to be a German sentence rather than just German words. And training the last device forces the German sentence to be a true translation.

There are many other problems researchers are exploring. An interesting one is how one can fool a deep network by making changes to an image that are so small a human cannot detect the changes, but cause the deep network to change the classification of the image [10]. All of a sudden what appears to be an image of a cat is classified as a car. The reason this is possible is that the set of images of a cat map into a manifold of dimension much smaller that the dimension of the activation space. Thus if one moves in activation space perpendicular to the surface of the manifold one is likely to change the classification.

AI programs do not extract the essence of an object and understanding its function or other important aspects. It may be another 40 years before we have another information revolution where function or other property is extracted. This will lead to an enormous range of intellectual ability.

III. Size of Graphs

In early research in the 1960's, graphs had ten to fifteen vertices. When the computer came graph size increased to 1,000 vertices, with faster computers 10^6 vertices. Then sparse graph such as the world wide web came with billions of vertices. Today we compute with 10^{100} graphs with 10^{100} vertices. Remember the number of atoms in the visible universe is only 10^{70}. How do we store a graph with 10^{100} vertices in the computer? We don't. One can do a random walk on a graph without storing it in the computer. All they need is an algorithm which given a vertex will identify the adjacent vertices. All we need to keep is the current vertex the random walk is at. However, how long does it take for a random walk to converge to its stationary probability? It turns out that if the graph is an expander, the random walk will converge to its stationary probability in logarithmic number of steps. For the 10^{100}vertex graph this means some number of step within 100 times some constant. Problems in of this size occurring in many applications are handled every day.

Given the size of data and graphs that are dealt with frequently requires that we randomly sample the data. This might require a random sequence which is a sequence with no short description. How can you store such a sequence? You don't. Instead you use a pseudo random sequence. This raises the question of how much randomness do you need. Usually one only needs two-way randomness. A sequence of zeros and ones is two-way pseudo random if each element is equally likely to be a zero or a one and given the value of one element in the sequence it does not give any information about any other element. If I give you two elements, I may be giving you information about all elements in the sequence.

An example where one uses randomness is in determining the number of distinct elements in a sequence. Suppose you work for a major chain store such as Walmart and want to know how many customers you have. You have access to a data stream of every purchase world-wide along with a credit card number associated with the purchase. You wish to count the number of card numbers. Each number is 16 digits long. You could set up a Boolean vector of length 10^{16}, or you could keep a linked list of numbers, or you could use a hash table or some other technique. However, if you are happy with a good estimate you can do this with only one word of storage. Keep track of the minimum credit card number. If you lay out a sequence of integers from one to $10^{16}$and mark every number you see, the expected distance between elements will be the 10^{16}divided by the number of distinct elements. Hence the minimum is approximately 10^{16} divided by the number of

distinct elements. Thus a good approximation for the number of distinct elements is 10^{16} divided by the minimum number seen.

One problem is that algorithm assumes the elements are random. This is not likely to be so since the credit card numbers might not be issued randomly. Thus you want to use a hash function to make the data statistically independent. We cannot store a hash function that will give full independence. However, only two-way independence is needed.

IV. Digitalization of Medical Records

As we digitize medical records the need for privacy and security becomes critically important. For example, if my entire medical history was digitized and I became ill somewhere in the world, I would like my doctor to be able to see my entire medical history to give me the best possible treatment. However, I do not want my insurance company to see my entire medical history. In fact, the insurance company does not need to see my medical record at all. All they need is a rigorous proof that they owe a doctor a certain amount of money. Medical researchers would like to access every one's medical record to improve medical techniques. How do we allow them to access statistical information without letting them have access to any individual information? Two techniques are emerging to help with this issue: zero knowledge proofs [11] and differential privacy [12].

A. Zero Knowledge Proof

A zero knowledge proof of a statement is a proof that the statement is true without providing any other information. To illustrate a zero knowledge proof, consider the game Sudoku. I can prove that I know how to fill in a Sudoku board without giving you any information on how to do it. I will take pieces of cardboard and write the appropriate number on each piece and place the card board pieces down over the appropriate squares so the numbers are not visible. Now you want to check that I have correctly filled in the first row. I pick up the cardboard pieces from the first row and shuffle them and show you that I have the correct numbers for the row. You check every row, column and three by three square and see that I have correctly filled in each. Actually this is not quite sufficient to prove that I have a correct solution since you don't know that I put the card pieces back in the same order each time. However, if I do not have a solution you will detect it with some probability and as you ask about more rows, columns and three by three squares with repetitions, you can drive the probability that I do not have a solution to zero.

A similar technique can be applied to three coloring a graph so that no two adjacent vertices have the same coloring. This is an NP-complete problem and there is no known polynomial time algorithm for the three coloring problem. Suppose you have a graph with a million vertices that you want to color and I have a business where I provide colorings. However, we cannot do business since we don't trust one another and you do not want to pay me until you know I actually have a coloring for your graph and I do not want to show you the coloring until you pay me. The solution is to give you a proof that I have a coloring without giving you any information as to the coloring. Again we use a zero knowledge proof.

For each vertex I place the appropriate color in an envelope and seal it. You ask to see the color of two adjacent vertices and I allow you to open the two appropriate envelops. This gives you no information about a coloring since one could permute the colors of the vertices to achieve the two colors of these two vertices. However, if I allow you to see another vertex, I have given you some information. So instead I destroy all the envelopes, permute the colors on the graph and recreate envelops with the appropriate color for each vertex. This sound like a lot of work. However, we don't use physical envelops, instead we agree on a digital encoding. When you want to see two vertices I give you the key to decode those two vertices. Since this is all done electronically it takes only a few minutes to convince you I indeed have a coloring and we can do business.

These toy problems are just examples of zero knowledge proofs.

B. Differential Privacy

Privacy is needed in many business applications involving car guidance, supply chains, and transportations systems. For example, the route guidance system in your car does not give you the best route guidance since it does not know the conditions of back roads and thus it keeps you on main roads. If the guidance system could improve routing and reduce mileage by a few percent, it would be a savings in millions of dollars of gasoline.

The route guidance system may record your GPS coordinates for the last month. If when you take your car in for service the GPS coordinates were down loaded, the route guidance system could improve its guidance by making use of the knowledge of local drivers. However, one may not want the GPS coordinates down loaded since one could determine the car owner by where the car was parked at night, where one works, shops, etc. However, if we could provide the condition of back roads without reveling any individual information it would be a success. Many systems that will be created in the future will face such problems of privacy.

V. Conclusion

The availability of large amounts of data, enormous computing power, the internet, and advances in AI are driving an information revolution. Intellectual tasks will be automated, changing the nature of work. In this paper we discussed some of the applications and advances that will influence our future. Those individuals, institutions, and nations that position themselves for the future will benefit enormously.

References

[1] Kun He, Yiwei Sun, David Bindel, John E. Hopcroft and Yixuan Li, "Detecting overlapping communities from local spectral subspaces," in

15th IEEE International Conference on Data Mining (ICDM), Atlantic City, USA, 2015, pp. 769-774.

[2] Yixuan Li, Kun He, David Bindel and John E. Hopcroft. "Uncovering the small community structure in large networks," in 24th International Conference on World Wide Web (WWW), Florence, Italy, 2015, pp. 658-668.

[3] Kun He, Sucheta Soundarajan, Xuezhi Cao, John E. Hopcroft and Menglong Huang, "Revealing multiple layers of hidden community structure in networks," CoRR abs/1501.05700, 2015.

[4] Kun He, Yingru Li, Sucheta Soundarajan and John E. Hopcroft, "Hidden community detection in social networks," Information Sciences, vol. 425, pp. 92-106, Jan. 2018.

[5] Alan Mislove, Bimal Viswanath, P. Krishna Gummadi, Peter Druschel, "You are who you know: inferring user profiles in online social networks," in Proceedings of the 3rd International Conference on Web Search and Web Data Mining (WSDM), New York, NY, USA, 2010, pp. 251-260.

[6] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Fei-Fei Li, "ImageNet Large Scale Visual Recognition Challenge," International Journal of Computer Vision, vol. 115, no. 3, pp. 211-252, April 2015.

[7] ImageNet Large Scale Visual Recognition Challenge (ILSVRC), [Online]. Available: http://www.image-net.org/challenges/LSVRC/.

[8] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John E. Hopcroft, "Convergent learning: Do different neural networks learn the same representations? " in ICLR, 2016.

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, Yoshua Bengio, "Generative adversarial networks," in Advances in Neural Information Processing Systems(NIPS). Montreal, Quebec, Canada, 2014, pp. 2672-2680.

[10] Anh Nguyen, Jason Yosinski, and Jeff Clune, "Deep neural networks are easily fooled: high confidence predictions for unrecognizable images," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015.

[11] Blum, Manuel, Feldman, Paul; Micali, Silvio. "Non-interactive zero-knowledge and its applications," in Proceedings of the twentieth annual ACM symposium on Theory of computing (STOC), Chicago, Illinois, USA, 1988, pp. 103–112.

[12] Cynthia Dwork. "Differential Privacy, Automata, Languages and Programming," in 33rd International Colloquium (ICALP), Venice, Italy, 2006, pp. 1-12.