Machine Learning Methods

Subjects: Computer Science, Artificial Intelligence
Contributor: Daniele Mazzei, Reshawn Ramjattan

Machine learning (ML) has a well-established reputation for successfully enabling automation through its scalable predictive power.

Keywords: machine learning ; topic modelling ; deep learning

1. Learning Paradigms

- Supervised Learning. This refers to methods that are trained using labelled examples. They can be highly accurate and trustworthy if the inferences made in real use are similar enough to the examples used during training.
- Unsupervised Learning. This refers to methods that are used on unlabelled data. They are relatively less accurate but can be effective depending on the scenario and problem.
- Reinforcement Learning. This refers to methods that reward positive or correct behaviour and punishes incorrect behaviour.

2. Neural Networks

An artificial neural network (ANN) is generally comprised of an input layer, one or many hidden layers of neurons and an output layer. An artificial neuron consists of input, weights applied to each input, a bias that is applied to the sum of the weighted inputs, and an activation function that converts the result to the expected form of the output (for example, the sigmoid function for a classification value of 0 or 1) [1][2][3].

A neural network with a single hidden layer is typically called a perceptron network, while networks with many hidden layers are referred to as Deep Neural Networks or Deep Learning and are at the core of many modern and popular ML methods [4][5].

3. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are one of the most popular and successful deep learning architectures. Although based on Neural Networks, they are mainly used in the field of Computer Vision (CV), for image-based pattern recognition tasks such as image classification.

Aside from input and output, the CNN architecture is typically composed of these types of layers: convolutional layers, pooling layers and fully connected layers. A convolutional layer computes the scalar product between a small region of the input image or matrix and a set of learnable parameters known as a kernel or filter. These calculations are the bulk of the CNN's computational cost. The rectified linear unit (ReLU) activation function is also applied to the output before the next layer. A pooling layer performs downsampling, by replacing some output with a statistic derived from close information. This reduces the amount of input for the next layer, therefore reducing computational load. A fully connected layer is where all neurons are connected as in a standard ANN. This followed by an activation function helps produce scores in the expected format of the output (i.e., a classification score). Despite being computationally expensive, CNNs have seen many successful applications in recent years [6][7][8].

4. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) perform particularly well on problems with sequential data such as text or speech or instrument readings over time. This is because, unlike other deep learning algorithms, they have an internal memory that is meant to remember important aspects of the input. A feedback loop instead of forward-only neurons is what enables this memory. The output of some neurons can affect the following input to those neurons [9][10].

However, because of the vanishing and exploding gradient problems caused by the way the neurons affect many others through memory in RNNs, their ability to learn effectively becomes limited. Hence, the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) methods aimed to solve this issue and rose to popularity as well. They do so by using gates to determine what information to retain [11][12][13][14].

5. Support Vector Machines

Support Vector Machines (SVMs) are linear models that can be used for both classification and regression problems. SVMs approximate the best lines or hyperplane for separating classes by maximising the margin between the line or hyperplane and the closest data points [15][16][17]. Although this best-fit separator can be used for regression, it is more commonly used for classification problems. It is considered to be a traditional ML method compared to its deep learning counterparts but can achieve good results with relatively lower compute and training data requirements.

6. Decision Trees and Random Forests

Decision trees are graphs comprised of nodes that branch off based on thresholds. They can be constructed by recursively evaluating nodes or features to find the best predictive features [18][19]. By itself, it can be used to make predictions, but to increase the performance of the method and mitigate overfitting, an aggregated collection of decision trees called a random forest can be used. Random forests as an ensemble learning method can accomplish this by training some trees on subsets of the data or features and aggregating the results [20][21]. The technique of training trees on different samples or subsets of data is called bootstrap aggregating or “bagging” [22].

They generally outperform decision trees but, depending on the data and problem, may not achieve an accuracy as high as gradient-boosted trees. Boosting is a technique where the random forest is an ensemble of weak learners or shallow decision trees that perform slightly better than guessing [23]. The intuition here is that weak learners are too simple to overfit and therefore their aggregated model is less likely to overfit. Gradient boosting builds on top of this by introducing gradient descent to minimize the loss in training [24][25]. An example of a popular and practical library implementation of gradient boosting is XGBoost [26].

Much like the aforementioned SVMs, algorithms based on decision trees are considered to be more traditional than deep learning methods and work especially well in situations with low compute and limited training data.

7. Autoencoders

Autoencoders are ANNs that follow the encoder–decoder architecture. They aim to learn efficient encodings of data in an unsupervised way. The encoder is responsible for learning how to produce these lower dimension representations from the input, while the decoder reconstructs the encodings to their original dimensions [27][28][29]. Autoencoders are commonly associated with dimensionality reduction, as a deep learning approach to the problem traditionally handled by methods such as Principal Component Analysis (PCA) [30]. Reconstruction by the decoder can be useful for evaluating the quality of encodings, generating new data or detecting anomalies if performance significantly differs from normal cases. So, generally, some common applications of autoencoders include anomaly detection, especially in cyber-security, facial recognition and image processing such as compression, denoising or feature detection [31][32][33].

8. Reinforcement Learning

Unlike the previously described supervised and unsupervised learning methods, Reinforcement Learning (RL) trains models by rewarding and punishing behaviour [34][35]. The intuition behind this is to let models explore and discover optimal behaviours instead of trying explicitly to train that behaviour with many samples. In RL, the model is defined as an agent that can choose actions from a predefined set of possible choices. The agent receives a sequence of observations from its environment as the basis or input for deciding on actions. Depending on the action chosen the agent is rewarded or punished for it to learn the desired behaviour.

This training is accomplished through defining concepts such as a policy, reward function and value function. A policy is a function that defines the agent’s behaviour, it maps the current observable state to an action and can be either deterministic or stochastic. A Value function estimates the expected return or reward of a certain state given a policy function. This allows the agent to assess different policies in a particular situation. The reward function returns a score based on the agent’s action in the environment’s state (i.e., a state–action pair).

Deep RL is attained when deep neural networks are used to approximate any of the prior mentioned functions [36]. Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C) and Deep Q Networks (DQN) are some examples of popular Deep RL algorithms. RL also sees successful practical use areas such as games, robotic control, finance, recommender systems and load allocation in telecommunications or energy grids [37][38][39].

9. Nearest Neighbour

The Nearest Neighbour (NN) method is a simple algorithm that finds a defined number of samples closest to the new input point [40][41][42]. It is often used as a method for classifying new points based on the closest stored points, where closeness as a metric of similarity can be defined but is usually standard euclidean distance. Computation of the nearest neighbours can be conducted by brute force, or by methods devised to address brute force's shortcomings such as K-D tree or Ball Tree [43][44]. Despite being such a simple method, NN has shown to be effective even for complex problems.

10. Generative Adversarial Networks

Generative Adversarial Networks (GANs) are unsupervised models concerned with recognizing patterns in input data to produce new output samples that would pass as believable members of the original set. The GAN architecture consists of a generator, a DL model for producing new samples, and a discriminator, a DL model for discerning fake samples from real ones. The discriminator receives feedback based on the known labels of which samples are real and the generator receives feedback based on how well the discriminator discerns its output. Thus, the networks are trained in tandem [45]. Despite being the most recent of the discussed methods (first described in 2014), its adoption in real cases is growing rapidly given the high potential usefulness of generating data points to support meaningful problems with limited data availability. Direct applications aside from training data synthesis also include, among others, image processing such as restoration or superresolution, image-to-image translation, generating music and drug discovery [46][47].

References

1. Krose, B.; Smagt, P.V.D. An Introduction to Neural Networks; MIT Press: Cambridge, MA, USA, 2011.
2. Bishop, C.M. Neural networks and their applications. Rev. Sci. Instrum. 1994, 65, 1803–1832.
3. Anderson, J.A. An Introduction to Neural Networks; MIT Press: Cambridge, MA, USA, 1995.
4. Jantzen, J. Introduction to Perceptron Networks; DTU Library: New Delhi, India, 1998.
5. Canziani, A.; Paszke, A.; Culurciello, E. An analysis of deep neural network models for practical applications. arXiv 2016, arXiv:1605.07678.
6. O'Shea, K.; Nash, R. An introduction to convolutional neural networks. arXiv 2015, arXiv:1511.08458.
7. Albawi, S.; Mohammed, T.A.; Al-Zawi, S. Understanding of a convolutional neural network. In Proceedings of the 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 21–23 August 2017; pp. 1–6.
8. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J.; et al. Recent advances in convolutional neural networks. Pattern Recognit. 2018, 77, 354–377.
9. Medsker, L.R.; Jain, L. Recurrent neural networks. Des. Appl. 2001, 5, 64–67.
10. Medsker, L.; Jain, L.C. Recurrent Neural NETWORKS: Design and Applications; CRC Press: Boca Raton, FL, USA, 1999.
11. Hochreiter, S.; Schmidhuber, J. Long short-term memory. Neural Comput. 1997, 9, 1735–1780.
12. Van Houdt, G.; Mosquera, C.; Nápoles, G. A review on the long short-term memory model. Artif. Intell. Rev. 2020, 53, 5929–5955.
13. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv 2014, arXiv:1412.3555.
14. Dey, R.; Salem, F.M. Gate-variants of gated recurrent unit (GRU) neural networks. In Proceedings of the 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, USA, 6–9 August 2017; pp. 1597–1600.
15. Noble, W.S. What is a support vector machine? Nat. Biotechnol. 2006, 24, 1565–1567.
16. Hearst, M.A.; Dumais, S.T.; Osuna, E.; Platt, J.; Scholkopf, B. Support vector machines. IEEE Intell. Syst. Their Appl. 1998, 13, 18–28.

17. Steinwart, I.; Christmann, A. Support Vector Machines; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2008.
18. Myles, A.J.; Feudale, R.N.; Liu, Y.; Woody, N.A.; Brown, S.D. An introduction to decision tree modeling. J. Chemom. J. Chemom. Soc. 2004, 18, 275–285.
19. Song, Y.Y.; Ying, L. Decision tree methods: Applications for classification and prediction. Shanghai Arch. Psychiatry 2015, 27, 130.
20. Biau, G.; Scornet, E. A random forest guided tour. Test 2016, 25, 197–227.
21. Breiman, L. Random forests. Mach. Learn. 2001, 45, 5–32.
22. Breiman, L. Bagging predictors. Mach. Learn. 1996, 24, 123–140.
23. Freund, Y.; Schapire, R.E. Experiments with a new boosting algorithm. In Proceedings of the ICML, Thirteenth International Conference on International Conference on Machine Learning, Bari, Italy, 3–6 July 1996; Volume 96, pp. 148–156.
24. Friedman, J.H. Greedy function approximation: A gradient boosting machine. Ann. Stat. 2001, 29, 1189–1232.
25. Freund, Y.; Schapire, R.E. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. 1997, 55, 119–139.
26. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
27. Liou, C.Y.; Cheng, W.C.; Liou, J.W.; Liou, D.R. Autoencoder for words. Neurocomputing 2014, 139, 84–96.
28. Badrinarayanan, V.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 2017, 39, 2481–2495.
29. Zhou, C.; Paffenroth, R.C. Anomaly detection with robust deep autoencoders. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 665–674.
30. Wold, S.; Esbensen, K.; Geladi, P. Principal component analysis. Chemom. Intell. Lab. Syst. 1987, 2, 37–52.
31. Yousefi-Azar, M.; Varadharajan, V.; Hamey, L.; Tupakula, U. Autoencoder-based feature learning for cyber security applications. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 3854–3861.
32. Gao, S.; Zhang, Y.; Jia, K.; Lu, J.; Zhang, Y. Single sample face recognition via learning deep supervised autoencoders. IEEE Trans. Inf. Forensics Secur. 2015, 10, 2108–2118.
33. Bank, D.; Koenigstein, N.; Giryes, R. Autoencoders. arXiv 2020, arXiv:2003.05991.
34. Sutton, R.S.; Barto, A.G. Reinforcement Learning: An Introduction; MIT Press: Cambridge, MA, USA, 2018.
35. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement learning: A survey. J. Artif. Intell. Res. 1996, 4, 237–285.
36. Li, Y. Deep reinforcement learning: An overview. arXiv 2017, arXiv:1701.07274.
37. Luong, N.C.; Hoang, D.T.; Gong, S.; Niyato, D.; Wang, P.; Liang, Y.C.; Kim, D.I. Applications of deep reinforcement learning in communications and networking: A survey. IEEE Commun. Surv. Tutor. 2019, 21, 3133–3174.
38. Polydoros, A.S.; Nalpantidis, L. Survey of model-based reinforcement learning: Applications on robotics. J. Intell. Robot. Syst. 2017, 86, 153–173.
39. Li, Y. Reinforcement learning applications. arXiv 2019, arXiv:1908.06973.
40. Cover, T.; Hart, P. Nearest neighbor pattern classification. IEEE Trans. Inf. Theory 1967, 13, 21–27.
41. Bhatia, N.; Vandana. Survey of nearest neighbor techniques. arXiv 2010, arXiv:1007.0085.
42. Peterson, L.E. K-nearest neighbor. Scholarpedia 2009, 4, 1883.
43. Bentley, J.L. Multidimensional binary search trees used for associative searching. Commun. ACM 1975, 18, 509–517.
44. Omohundro, S.M. Five Balltree Construction Algorithms; International Computer Science Institute Berkeley: Berkeley, CA, USA, 1989.
45. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. Commun. ACM 2020, 63, 139–144.
46. Gui, J.; Sun, Z.; Wen, Y.; Tao, D.; Ye, J. A review on generative adversarial networks: Algorithms, theory, and applications. IEEE Trans. Knowl. Data Eng. 2021.

47. Aggarwal, A.; Mittal, M.; Battineni, G. Generative adversarial network: An overview of theory and applications. Int. J. Inf. Manag. Data Insights 2021, 1, 100004.