Research Literature
2019-2020 Masters Thesis Development

Riley Miller (rileymiller@mymail.mines.edu)
Colorado School of Mines

November 1, 2019

References

[1] F. Chollet, *Deep Learning With Python.* Shelter Island, NY: Manning, 2018.

> Deep learning is a mathematical framework for learning representations from data. The author describes deep learning as a multistage way too learn data representations. Deep learning has become widely popular because it automates the most crucial step in the machine learning workflow, *feature engineering*. The automation of this process has simplified many complex data pipelines used for *shallow learning* algorithms by learning all of the features in one end-to-end pass through a deep learning model. The author describes the process by which deep learning learns as *the incremental, layer-by-layer way in which increasingly complex representations are developed and the fact that these intermediate representations are learned jointly.* One of the biggest breakthroughs that deep learning introduces is hte ability for a model to learn all layers of representation *jointly*, at the same time, instead of in succession like the approach used for *feature engineering* in shallow models. The author outlines that the two most effective techniques in Machine Learning are *gradient boosting* for shallow learning where structured data is available and deep learning for perceptual problems. Machine Learning is finally accessible due to advances in the internet, providing much larger data sets, and graphics

chips that were developed for gaming. The gaming market effectively subsidized supercomputing for the next wave of machine learning, specifically deep learning.

Definitions

Weights: The specification of what a layer does to it's input data is stored in the layer's *weights*.

Parameterized: The transformation implemented by a layer is *parameterized* by its weights.

Learning: *Learning* means finding a set of values for the weights of all layers in a networkm such that the network will correctly map example inputs to their associated targets.

Loss function: The *loss function* in the context of Deep Learning is the method used to measure the output of a neural network and measure how far this output is from what was expected.

Optimizer: Uses *backpropagation* to make the adjustment of weights based off the score of the *loss function* in a manner that will lower the loss score for the current example.

Backpropagation: The central algorithm in deep learning. TODO add more here

Training Loop: At the beginning of training, the network weight's are assigned random values, thus, the network performs a series of random transformations and the loss score is correspondingly high. Then with every example the neural network processes the weights are adjusted a little each time in the correct direction, causing the loss score to decrease.

Probabilistic Modeling: One of the earliest forms of machine learning, Naive Bayes is an example. A closely related model is the *logistic regression*, a classification algorithm. Logist regression is usually one of the first algorithms a data scientist will try on a data set to get a handle on the dataset.

Support Vector Machines (SVM): aim to solve classification problems by finding good *decision boundaries* between two sets of points belonging to two different categories. SVMs find these boundaries in two steps: mapping data to a new high-dimensional representation where the *decision boundary* can be expressed as a hyperplane and trying to maximize the distance between the hyperplane and the closest data points from each class in a process called *maximizing the margin.* SVMs use the *kernel trick* to find good decision hyperplanes in the new representation space. This happens by only computing the distance between the pairs of points in that space, which can be done using a *kernel function.* A *kernel function* is a computationally tractable operatoin that maps any two points in the initial space to the distance between these points in the targer representation space, bypassing the explicit computation of the new representation. SVMs were widely popular for a long time but were hard to scale to large data sets and perceptual problems. Since SVMs are a flavor of *shallow learning*, applying SVMs to perceptual problems like image classification requires extracting useful representations manually (*feature engineering*) which is difficult and brittle.

Decision Trees: A flowchart-like structure that enables the classification of input data points or to predict output values given inputs. Preferred to kernel methods by 2010.

Random Forest: An algorithm based on *decision trees* which builds a large number of specialized decision tres and then ensembles their outputs. Almost always the second-best algorith for shallow machine-learning tasks.

Gradient Boosting Machines: Considered one of, if not *the* best shallow learning algorithm. Gradient boosting machines is a technique based on ensembling weak prediction models, generally *decision trees.* It uses *gradient boosting* to improve any machine-learning model by iteratively training new models that specialize in addressing the weak points of the previous models. When applied to *decision trees* the resulting models outperform *decision trees* most of the time.

CNN: Deep convolutional neural networks, *convnets*, became the goto algorithm for computer vision problems after becoming the goto technique for the ImageNet competition. Has completely replaced SVMs and decision tress in a wide range of applications.

[2] H. S. H. Cheng L. Koc J. Harmsen T. Shaked T. Chandra H. Aradhye G. Anderson G. Corrado W. Chai M. Ispir R. Anil Z. Haque L. Hong V. Jain X. Liu, “Wide & deep learning for recommender systems,” in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, September 2016.

Cheng et al. implemented a combonation of wide and deep learning to surface relevant content to users on the Google Play store. They used the two different techniques to combat some of the model fallacies that can appear in recommender systems. The team of researchers tackled the two primary tradeoffs for recommender systems by using wide and deep learning to optimize for memorization and generalization. Memorization in recommender systems "can be defined as learning the frequent concurrence of items and features and exploiting the correlation in historical data". On the other hand, generalization is based on transitivity of correlation and explores new and rare feature combination, this approach leads to greater diversity of recommendations. On a practical basis, the researchers utilized wide learning to help with the memorization aspect, many recommender systems utilize simple logistic regression algorithms to surface relevant content. However, this approach falls short when trying to query item feature pairs that haven't appeared in the training data. On the other hand, many researchers have tried to use deep learning neural networks to generate recommendations since they're able to generate feature pairs that haven't appeared in the training set, however, these deep learning approaches can over generalize and make less relevant reccomendations. These researchers made use of joint training which simultaneously optimzes the parameters of the wide and deep models at training time. The teams data pipeline consisted of three stages: data generation, model training, and model serving. During training their model input layer took in training data and vocabulary and generated sparse and dense features with labels. The wide learning component consisted of the cross product transformation of user installed apps and impression apps. The deep learning model consisted of a 32 dimension embedding vector which is learned for each categorical feature. These embeddings are then concatenated together with dense features, resulting in a dense vector of approximately 1200 dimensions. These models were trained on over 500 billion examples. To optimize for performance, the researchers optimized the performance by using multithreading parallelism by running smaller batches in parallel instead of running all of the candidate inferences through the model at the same time.

The recommender servers recommend over 10 million apps a second. https://dl.acm.org/citation.cfm?id=2988454

[3] H. W. N. Wang and D.-Y. Yeung, "Collaborative deep learning for recommender systems," *Proc. KDD*, pp. 1235–1244, 2015.

These researchers introduced a concept they called collaborative deep learning (CDL) which is a hierarchical Bayesian model which combines deep representation learning for the content information and collaborative filtering for the ratings. The paper introduces three main approaches for recommender systems: content based models, collaborative filtering models, and hybrid methods. The content based methods make use of user profiles and product descriptions to make recommendations. CF approaches use history such as past activities or preferences without using user or product info. While hybrid methods utilize a mixture of both content and CF approaches. Limitations for content models are user privacy while CF approaches stumble when ratings on products are sparse. The researchers highlight that deep learning models are state of the art in other fields such as computer vision and NLP and that they are able to learn features automatically, yet they are inferior to shallow models such as CF when it comes to learning similarity and relationships between items. To construct their CDL model, the researchers utilized Stacked Denoising Autoencoders (SDAE) which is a feedforward neural network for learning representations.