Research Literature
2019-2020 Masters Thesis Development

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

November 18, 2019

References

[1] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. Motahari-Nezhad, E. Bertino, and S. Dustdar, "Quality control in crowdsourcing systems: Issues and directions," *IEEE Internet Computing*, vol. 17, no. 2, pp. 76–81, 2013.

> This paper focuses on quality control in crowdsourcing platforms like Amazon MTURK focusing on ensuring high quality crowd contributions. The authors introduce a taxonomy for crowdsourcing platforms that records the workers' reputation and expertise. One of the major issues with quality in crowdsourcing platforms is how subjective quality can be. The researchers outline a potential solution for improving quality-control approaches by developing a machine-learning based recommender system that will surface relevant quality control approaches to requestors based on the requester's profile, history, as well as the history of the crowd and the quality requirements of the task. One of the other topics of potential research is the development of crowdsource platform middleware which will solve the problem of having distributed crowds over various crowdsourcing platforms. This would be an interesting area for a machine learning application.

[2] S. Amer-Yahia, E. Gaussier, V. Leroy, J. Pilourdault, R. Borromeo, and M. Toyama, "Task composition in crowdsourcing," 2016.

[3] 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.
Overfitting: When Machine learning models perform worse on new data than on their training data. TODO, add more
Tensor: The primitive datastructure for machine learning. A generalization of matrices to an arbitrary number of dimensions.
Rank: The number of dimensions of a matrix, in the context of ML the number of dimensions of a tensor. *Dimensions* are often called an *axis* in the context of tensors.

[4] M. De Gemmis, L. Iaquinta, P. Lops, C. Musto, F. Narducci, and G. Semeraro, "Preference learning in recommender systems," *Preference Learning*, vol. 41, pp. 41–55, 2009.

The authors outlines the need for reccomender systems to surface relevant content in the internet of information overload. The authors begin by describing the need to collect implicit and explicit user data to develop contextual recommendation systems. The basic architecture for recommender systems outlined in this paper includes a user profile, an algorithm to update the profile given usage/input information,

and an adaptice tool that exploits the profile in order to provide personalization. One of the primary methodologies the authors pointed out was Collaborative Filtering which they defined as system's recommendation based on evaluation from users who share similar interests among them (may apply to my research based on worker and requestor ratings based on historical performance).

[5] A. M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 278–288.

TODO read and annotate

[6] A. Ghorbani and J. Y. Zou, "Data shapley: Equitable valuation of data for machine learning," in *ICML*, 2019.

In this paper the authors applied Shapley techniques from game theory in economics to quantify the value of a data point in a data set used to train a model. This researcher is interesting because it shows quantitative performance improvement of existing machine learning models applied to an existing data set. It was really interesting how the authors were able to apply this concept from game theory to evaluate the contribution of a data point in training data on the model data and the corresponding actions of removing poisoning data or add similar data points was able to considerably increase or decrease the performance of machine learning techniques. This research shines light on two very interesting concepts, the first being how to improve model effectiveness by quantitating the contribution of each data point in the training set–the second being how to evaluate the value of user data to compensate user's for their data. This concept may be used in my research to evaluate the contribution of users or tasks in my training data sets which can be removed or lead to the addition of valuable data based on similar data to improve my algorithm performance.

[7] 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

[8] T. Kaplan, S. Saito, K. Hara, and J. P. Bigham, "Striving to earn more: a survey of work strategies and tool use among crowd workers," in *Sixth AAAI Conference on Human Computation and Crowdsourcing*, 2018.

This paper outlines survey results collected from 360 crowdsourcing workers on their approach for earning money on crowdsourcing platforms and the tools they used. One of the interesting tools that crowdsource workers use is called Preview and Accept (commonly referred to as PandA) which finds similar tasks and batches them together to decrease the time that workers spend finding tasks. The survey outlines the inequality between workers and requesters and ultimately how requesters have a way to judge workers based on the reputation, workers don't actually have a way to sift through and filter tasks to find tasks that are efficient and feasible. This is directly related to the problem that I am trying to solve, improving efficiency of workers in crowdsourcing platforms. The researchers explain that much of the previous work done on task recommendation in Amazon MTURK deals with information available in the AMT search like task keywords, reward, qualificatoin, in conjunction with the browser extension Turkopticon, a tool used by crowdworkers to rate requesters, to surface wage-efficient tasks. The researchers also discussed their perspective that task recommendation systems should focus on content dependent factors which affect the amount of time it takes to complete a task like the types of media in a task and the amount of inputs outlined in the task. The survey results reported a large number of forums for MTURK crowdworkers which would be really helpful for empathizing common problems that crowdworkers experience. One of the interesting things from the survey was the level of frustration for finding tasks within the platform and that not being

able to find anymore tasks to complete was a high motivation for ending a working session. This is a really important piece of feedback that provides validation for researching approaches for improving the efficiency of crowd workers. An interesting facet of the survey results was the reported priority of crowdworkers on selecting a task in which "Pay per HIT", "Expected Task Completion Time", and "Requester Reputation" were ranked in the order previously described. In the researchers' future work section they outlined several possibilities for applying machine learning to improve crowdworker efficiency. The researchers outlined some potential approaches: develop a machine learning algorithm to surface recommendations based off of task feasibility and the estimated time it would take to complete the task, as well as to use the task content and metadata to help make a prediction.

[9] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, no. 1, pp. 76–80, 2003.

One of the primary drawbacks of collaborative filtering is that it requires online computation and the algorithm performance is linear with regards to the number of customers in the dataset, this isn't scalable for massive retailers like Amazon unless you were to reduce the data sample or the dimensionality of the data set all of which would reduce the quality of the recommendations. An alternative to collaborative filtering is clustering models which perform offline calculations with the primary drawback being reduced quality of recommendations. The approach Amazon took was item-item collaborative filtering where instead of developing recommendations off of similar customer decisions the researchers keyed in only on surfacing items similar to items the customer had either bought or reviewed, drastically reducing the computation and memory concerns of traditional collaborative filtering while still surfacing high quality recommendations.

[10] S. Okura, Y. Tagami, S. Ono, and A. Tajima, "Embedding-based news recommendation for millions of users," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1933–1942.

In this paper the authors utilize a three step process to surface news articles. In the first step of the process they utilize a denoising autencoder, a RNN, and inner product operations to recommend relevant news information quickly. The authors used cosine similarity to reduce duplication of articles before they submit data to the input layer of the RNN, this allows them to reduce the processing time by reducing the level of useless input. This would be an interesting technique that could be applied to reducing duplicate tasks. They also used offline experiments using logs recorded on their server to determine the accuracy of their RNN. The researchers training set was massive, recording over one billion browses. The test set was reduced to a random sample of 500,000 sessions. The researchers compared a GRU (Gated Recurrent Unit) RNN to a variety of techniques and found the GRU RNN to perform best. However, after left in production for over three months the RNN performance had degraded to that of the words based model approach, due in part that it took over a week to train the RNN.

[11] I. Portugal, P. Alencar, and D. Cowan, "The use of machine learning algorithms in recommender systems: A systematic review," *Expert Systems with Applications*, vol. 97, pp. 205–227, 2018.

This study researches many of the primary machine learning algorithms used in real reccommender systems (RS). The purpose of their study was to provide researchers a comprehensive overview of the work being done on RSs and potential areas for further research, specifically targeting the identification of ML algorithms that are used in RSs and to discover open questions in RS development that might be impacted by software engineering. The authors outline that there are three main categories of recommender systems: collaborative, content-based, and hybrid filtering. The collaborative approach pertains to considering user data when processing information for recommendations (recommendations based off other users). The content-based approach has to do with item data the user can access. The hybrid approach combines both collaborative and content based filtering. Many applications collect both explicit and implicit information about the

user and store it on the user's computer using the browser's local storage, accessed using the browser's API. The explicit data that is collected occurs when users are aware that they are providing information about themselves, whereas, during implicit data collection user's are often unaware that their data is being collected (user history on the site, clicks, keystrokes, etc.). The most commonly used algorithms included Bayesian, Decision Tree, Matrix factorization-based, Neighbor-based, Neural Networks, and rule learning. The authors theorized that Bayesian and Decision Tree algorithms appeared most commonly because they are less complex than some of the other approaches. The authors suggested that there is a lot of work that could be done in researching the application of neural networks to RS.

[12] C. Sun, N. Rampalli, F. Yang, and A. Doan, "Chimera: Large-scale classification using machine learning, rules, and crowdsourcing," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1529–1540, 2014.

In this paper the team at Walmart Labs developed a tool referred to as Chimera which implements a combination of learning, rules, and crowdsourcing to classify products by their product description. Their solution demonstrates that classical assumptions that are applied to classification problems break down at the scale that they deal with (5000+ categories) and that a combination of approaches is required to solve the problem.

[13] 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.

[14] X. Wang, L. Yu, K. Ren, G. Tao, W. Zhang, Y. Yu, and J. Wang, "Dynamic attention deep model for article recommendation by learning human editors' demonstration," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 2051–2059.

In this article the researchers utilize deep learning to learn the relationship between articles and there meta data in conjunction with a deep learning model to learn the underlying preference of the article reviewers which cannot be predicted using bag of word techniques since article reviewer preferences depends more on the writing quality vs. specific keywords found in the metadata or the article. This specific approach was focused on surfacing a subset articles for content reviewers for media companines like Buzzfeed to reduces their workload. These reviewers then forward interesting/appropriate articles to machine learning models that will match these articles to consumers, an area that has drawn much more attention from researchers. The researchers touched on two traditional recommender techniques in their related work sections stating that collaborative filtering runs into issues with sparse datasets and thus faces the cold-start problem, and also touched on the content-based recommendation approach which deals with relationships between items in the dataset and won't run into issues with a cold start since they're reliant on the relationship between items, however, the content-based approach runs into issues with providing personalized reccomendations to the user, a crucial component of an ef-

fective recommendation system. The focus of the researchers was to create models that learned from the textual content of the articles as well as the articles' metadata like the author, original media, author city, and various other contextual information to provide relevant predictions. The authors' problem is unique to other reccommender systems in that the data is sparse becuase news articles become outdated quickly and thus the data they need to perform recommendations on won't have a lot of user interaction, rendering collaborative filtering essentially useless. Encouraging the researchers to focus on a content based recommendation system. For their CNN the researchers used character level neural network language models (NNLM) for low-level textual feature learning. The researchers found that the deep learning CNN technique had similar but better results that a logistic regression technique.

[15] S. Wu, W. Ren, C. Yu, G. Chen, D. Zhang, and J. Zhu, "Personal recommendation using deep recurrent neural networks in netease," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 2016, pp. 1218–1229.

In this experiment the researchers forged a feedforward neural network which simulated collaborative filtering (accepted user purchase history as input to generate recommendation) and a conventional RNN to generate realtime reccomendations based of the user's browsing session. The researchers explain that an RNN is a suitable choice of architercture for their problem because an RNN with on inner layer can determine the order of page access and an RNN with multiple layers can determine the combination of page accesses. One of the keypoints that the researchers make concerning their use of deep learning vs. traditional collaborative filtering is that collaborative filtering is effective at making recommendations based off of previous user choices whereas their deep learning model is able to make a new prediction based off of the given information of a user. To train their model the researchers used Stochastic Gradient Descent. Something that was really interesting about the researchers' approach is that they would iteratively retrain the model in real-time to give

the most up to date predictions of how users interacted with the ecosystem. Another aspect that was interesting about the researchers' approach was their use of a "history state" that they used to consilidate decisions that dated outside of their sliding window of sessions concept. The researchers showed that their RNN approach significantly outperformed a collaborative filtering approach.