Detecting Feature Interactions in Agricultural Trade Data Using A Deep Neural Network [⋆]

Jim O' Donoghue, Mark Roantree, and Andrew McCarren

Insight Centre for Data Analytics, School of Computing, DCU, Collins Ave., Dublin 9
jim.odonoghue@insight-centre.org, {mark.roantree,andrew.mccarren}@dcu.ie

Abstract. Agri-analytics is an emerging sector which uses data mining to inform decision making in the agricultural sector. Machine learning is used to accomplish data mining tasks such as prediction, known as *predictive analytics* in the commercial context. Similar to other domains, hidden trends and events in agri-data can be difficult to detect with traditional machine learning approaches. Deep learning uses architectures made up of many levels of non-linear operations to construct a more holistic model for learning. In this work, we use deep learning for *unsupervised* modelling of commodity price data in agri-datasets. Specifically, we detect how appropriate input signals contribute to, and interact in, complex deep architectures. To achieve this, we provide a novel extension to a method which determines the contribution of each input feature to *shallow, supervised* neural networks. Our generalisation allows us to examine *deep* supervised *and* unsupervised neural networks.

1 Introduction and Motivation

Agri-analytics is an emerging sector which uses data mining and analytics to improve decision making in a market contributing €24 billion to Ireland's economy. In an industry of this size, decision makers in the agricultural (agri) sector require appropriate tools and up-to-date information to make appropriate predictions across a range of products and areas. Despite a high volume of both free-to-use online datasets and highly processed pay-to-use datasets, this is still very difficult. For example, the prediction of future prices for many agri products. Determining how complex input data signals interact and contribute to agri-datasets, would enable us to better *understand* this data and leverage it for prediction. Deep learning refers to a recent breakthrough in machine learning, where *deep architectures*, made up of many levels or *layers* of non-linear operations are used to model data. A central premise behind deep learning is that, like the brain, these algorithms can learn high-level, abstract features from data [1]. These high-level features better represent the outcome or dataset being modelled and correspond to *latent* variables in the dataset. The lower layers correspond to localised, specific features and as the data progresses through the architecture, it is transformed into more abstract, higher level representations.

[⋆] Research funded by Science Foundation Ireland under grant number SFI/12/RC/2289

Motivation. An important task in data mining and machine learning is the detection of how variables interact when predicting an outcome or describing data. These interactions enable us to extract the complex relationships between inputs, a highly important task in argi-analytics. Neural networks, especially deep neural networks have great potential in this regard as they can successfully learn multiple layers of highly complex feature interactions or *feature representations*, but they are notoriously hard to interpret and lack a methodology to interpret these learnt features and thus, are often seen as "black box" solutions.

Contribution. The connection weight method [14] was designed for a single hidden layer, single output *supervised* classification *shallow* neural network. Our contribution is an extension to this. We present the *deep Connection Weight*, or dCW method. The dCW approach gives us a *generic* means to interpret supervised *or* unsupervised *deep* networks and the multiple layers of highly complex features therein. Furthermore, we use the dCW and a Deep Belief Network (DBN) to extract highly complex interactions between the multiple input signals in agri trade data.

Paper Structure. The paper is structured as follows: in Section 2 we discuss related research; in Section 3, we present detailed description of the methods employed in this paper, in terms of the algorithmic optimisation and configuration; in Section 4 we present our novel dCW method for interpreting the feature representation of a deep network; Section 5, describes our dataset, transformations, experiment setup as well as the results and analysis; and finally in Section 6, we present our conclusions.

2 Related Research

Deep and shallow neural networks are often seen as uninterpretable "black boxes". However, there have been studies [6], [14] to compare and examine various techniques to interpret *shallow, supervised* networks. In [6] and [14] they discuss: Garson's algorithm [5]; the connection weight (CW) method; partial derivatives; perturbing inputs; sensitivity analyses; and various forms of stepwise addition, elimination and selection. In [6] they compare many of these methods on a dataset of ecological information. The review of [14] employs a synthetic dataset with known properties. Therefore, the comparisons performed in [14] are more accurate and can be generalised to other datasets. Garson's algorithm [5] was one of the first popular methods for summarising neural networks. However, it only accounts for the cumulative *magnitude* of the weight. The CW method [14] examines the cumulative magnitude *and* sign (positive or negative) of weights. The CW method was also shown to outperform all others [14]. Thus, it was selected as the basis of our approach. The CW method can successfully interpret single classification, single hidden layer *shallow* neural networks but does not generalise to *deep* networks. The research presented here extends this method to encompass these more complicated networks.

Table 1. Hyper-Parameter Search Space

Hyper-Parameter	Description	Bounds (low, high)
b_size	samples in a mini batch	(856, 8560)
l_hidden	number of hidden layers	(1, 3)
o_nodes	number of hidden layer nodes	(1, 204)
learning_rate α	co-efficient for weight updates	(0.00001, 0.1)
patience	minimum epochs to iterate	(10, 150)
max_epochs	max possible epochs to iterate	(10, 1000)

3 Deep Belief Network Components

In this section we describe our approach to construct and optimise a Deep Belief Network (DBN). Section 3.1 describes the two parameter-types involved and the procedures we adopt for their initialisation and optimisation. Section 3.2 describes how DBNs are configured and relate to the strategies in Section 3.1.

3.1 Parameter Initialisation and Optimisation

Parameter optimisation enables the discovery of the best learner model for pattern extraction. There are two types and thus two levels of optimisation in an experiment, which we define. *Hyper*-parameters are optimised at the *Trial* level and influence algorithm architecture and training. *Model* parameters, optimised at the *Run* level, are used for the predictive or descriptive learning task.

Model Parameter Initialisation and Optimisation. The procedure for *model parameter* initialisation was adopted from the literature [7]. This encompasses initialising all bias terms to zero and for logistic activation units, randomly initialising layer weights between the bounds given in Equation (1). The number of inputs to a layer is given by n_{in} and the number of nodes in a layer by o_{nodes}. Model parameters were optimised with *mini-batch stochastic gradient descent* (MSGD) [4] and *early-stopping* [15].

$$[-4\sqrt{\frac{6}{n_{in}+o_{nodes}}}, 4\sqrt{\frac{6}{n_{in}+o_{nodes}}}] \tag{1}$$

Hyper-Parameter Initialisation and Optimisation. Table 1 shows hyper-parameter initialisation bounds. We discuss the upper and lower bound for each. Batch size (*b_size*) - number of samples in an MSGD data batch - was bounded at roughly 1% and 10% of total dataset samples. Layer counts (*l_hidden*) were bounded at 1 (shallow) and 3 (deep). Node counts in each layer (*o_nodes*) were searched between 1 and $2 \cdot n_{inputs} + n_{inputs}$. Learning rate bounds are typical of the literature [2], with a reduced upper bound to avoid vanishing gradients.

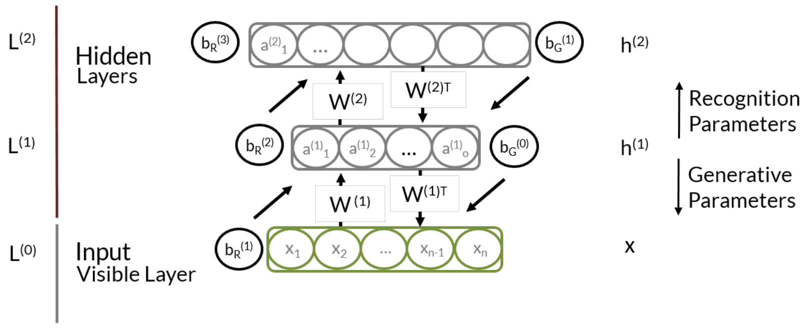

Fig. 1. Unsupervised DBN Configuration

Finally, early stopping parameters *patience* - minimum number of dataset iterations (epochs) - and *max_epochs* - maximum number - had a lower bound of 10 and an upper bound of 150 and 1000, respectively. Random search was used to optimise hyper-parameters [2].

3.2 Architectural Configuration

A Deep Belief Network (DBN) is an algorithm architecture characterised by an unsupervised pre-training phase and an *optional* supervised or unsupervised fine-tuning stage [1]. A DBN is trained in a greedy, unsupervised, layer-wise fashion where each layer is trained as a Restricted Boltzmann Machine (RBM). The first RBM is completely optimised before its hidden values are treated as the visible inputs for the next layer, and so on, until all layers are combined as presented in Figure 1. Values can propagate from $L^{(0)}$ through $L^{(1)}$, to the final hidden layer $L^{(|L|)}$ via relevant parameters and functions and can propagate down to form an accurate *reconstruction* of inputs. When the DBN can accurately reconstruct the data from its hidden layers, it has learnt an abstract representation of the inputs. Thus, a method to examine these abstract representations would allow us to determine how input features interact. To train RBMs we used CD_1, one-step *Contrastive Divergence*. This amounts to 1 step of **Gibbs sampling** at each iteration of MSGD [8]. We refer the reader to [13] for (hidden) activation, energy, cost and derivative functions.

4 An Approach to Interpreting Deep Representations

The CW method was designed for a *single* hidden layer and *single* output *supervised* feed-forward network. Its inputs are: the first hidden layer weight matrix $W^{(1)}$ and the output weight vector $W^{(2)}$. These respectively contain connection weights between: every input to every hidden node; and every hidden node to the classification. Consider the first row in $W^{(1)}$, which maps the first input to each hidden node. Each value in this row is multiplied by each output weight in $W^{(2)}$. This gives a number of values equal to the number of hidden nodes and the contribution of the first input to each input1-hidden-out pathway. Summing these values gives the contribution of this input to the model overall. This process repeats for every row in $W^{(1)}$, to give the contribution of each input. Equation (2) shows the CW method formulated as the dot product between the input and output tensors. This forumlation facilitates our generalisation and the method that follows.

$$CW_{score} = W^{(1)} \cdot W^{(2)} \tag{2}$$

Equation (3) shows the calculation of $\varphi \in \mathbb{R}^{n \times H}$, where n is the number of input features and H is the number of nodes $|N^{(|L|)}|$, in the final layer of a neural network with a set of L layers. We call this the feature *score matrix.* It is the first part of our deep connection weight (dCW) method and generalises the CW approach to networks with *arbitrary numbers* of hidden layers and nodes in the final layer. The *score matrix* is the result of a cumulative dot product of all weight matrices $W^{(1)} \cdot W^{(2)} \ldots W^{(|L|)}$ in the network being examined. The score matrix φ, calculates a single value for the *importance* of each input feature to each node in the top layer of a neural network. This allows us to rank input feature contributions to each node in the final hidden layer nodes and thus, on further analysis examine how they *interact*, which is very useful.

$$\varphi = \dot{\prod}_{i=0}^{|L|} W^{(i)} \tag{3}$$

In some cases, it is necessary to determine the *rank* of each input feature's importance and its contribution to the model as a whole, *or* how much each of the hidden nodes contribute to the overall model. The next steps summarise the score matrix φ and allow us to do this. Equation (4) shows how to determine the overall contribution of each input feature to the learner model. For each feature row vector $f_1, f_2, \ldots f_n$ in the score matrix φ, we sum the value in each column $j \in H$ to obtain the *contribution score* for each *input* feature. Summing over the opposite axis gives φ_N, the contribution of each *node* in the final hidden layer to the entire learner model.

$$\varphi_{f_i} = \sum_{j=1}^{H} \varphi_{ij} \tag{4}$$

5 Experiments

In this section, we present the results of the contributions of inputs and the interactions discovered in the agri dataset. We first describe our experimental setup and the dataset used, before discussing and analysing our results.

Table 2. Input Feature Summary

Cat.Num.	Category	Num.Vars.	Transformation
1	time	2	one-hot
2	countries	2	one-hot
3	product	1	one-hot
4	primary trade info.	2	normalised
5	weight secondary trade info.	6	normalised
6	value secondary trade info.	7	normalised
7	feed price	10	normalised
8	live price	10	normalised
9	slaughter info.	17	normalised
10	meat price	3	normalised
11	currency info.	8	normalised

5.1 Setup

Experimental Environment. Experiments were run on Ubuntu 14.04.4 LTS and a NVIDIA GeForce GT 620 810MhZ Graphical Processing Unit. Code was developed using 64-bit Python 2.7.6 in PyCharm 2016.2.3. IDE. Experiments use the CDN and Deep NoSQL Toolkit [13], [12], Theano 0.7.0 [3], Pandas 0.18.0[10], PyMongo 2.6.3 [11], Mongo 3.2.11. For each trial of 128, bootstrap sampling and nested cross-validation was used.

Dataset. The dataset was generated from 5 fact tables in an agri data-warehouse [9]. European Union trade data 2010 - 2014 (inclusive) was used to eliminate missing values, resulting in 85,602 samples. Table 2 shows a summary of the 68 input features and data transformations performed on each. For description we have divided them into 11 categories. *Category* provides a label for each category, *Num.Vars.* is the number of variables each contains and *Transformation* describes how the data was processed. Variables in categories 1-3 were one-hot encoded. Other variables were normalised between 0 and 1. The *time* category relates to month and year. *Countries* relates to the country exporting (27 possible) and importing (14 possible). *Product* describes the *type* of pork cut being exported. *Primary trade info.* relates to the volume of product being exported in kilograms, and its value in Euro. *Secondary trade info* (weight and value) relates to the trade volumes in kgs and value in euro from other major exporting regions in the world in the same month, as well as the entire EU. The value, but not the weight of Russian trade was available as secondary trade information, hence the 6:7 discrepancy. *Live price* contains the price per kg of live animal for countries where this was available. *Feed price* contains all available information for the price of pig feed in the 5 regions studied. *Slaughter info.* contains available information on how many pigs were slaughtered in each region (supply of product). *Meat price* relates to the price per kg of pork product in Europe, the US and Canada. Finally, *currency info.* contains exchange rates for a wide range of currencies.

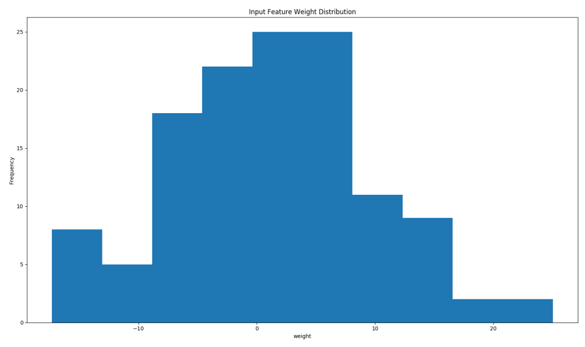

Fig. 2. Histogram of Input Node Contributions

Table 3. Top 5 Error Scores and Architecture Hyper-Parameters

rank	trial_id	l_hidden	o_nodes1	o_nodes2	o_nodes3	t_scores	v_scores	tst_scores
1	52	**2**	**5**	**130**	na	-0.0163	**-0.0018**	**-0.001**
2	74	3	169	63	60	74	0.0978	-0.0025
3	126	2	29	79	na	**-0.005**	-0.0035	-0.0197
4	31	2	149	27	na	0.0318	0.0082	0.0661
5	94	2	118	46	na	0.0255	0.0087	-0.0697

5.2 Results and Analysis

We first explore top architectural hyper-parameters and associated error rates. These provide insight into interaction analyses that follow, but are not central to the paper. Due to space restrictions, we omit training hyper-parameters and in-depth analysis of these results. Table 3 shows the the top 5 training (t_score), validation (v_score) and test (tst_score) reconstruction cross entropy scores and associated hyper-parameters of the 128 trials. Error scores closer to zero are best and are ranked according to validation performance. Trial 52 obtained the smallest validation *and* test generalisation scores at -0.0018 and -0.001 respectively. Trial 126 obtained the lowest training score of -0.005. All configurations are deep - four have 2 hidden layers and one has 3. The top performing - Trial 52 - has 5 nodes in its first hidden layer and 130 in its second. The next best has 169 nodes in the first, 63 in the second and 60 in the third hidden layer. For all 5, there is roughly a ratio of at least 3:1 nodes between input-hidden or hidden-input layers suggesting a combination of data compression or summary and then expansion or separation of signals, best extracts the interactions between inputs.

Feature Representation and Interaction Results. For analysis of features and interactions we focus on φ_f and top ranked contributors therein. Figure 2 shows the distribution of cumulative weight scores in φ_f. The x-axis shows the cumulative weight score and the y-axis shows the number of features with that score. It highlights that a small proportion of inputs has the largest contribution to the model and that contributions are normally distributed. Figure 2 shows roughly: 50 to 60 features have scores in the range 0 to 10; fifteen have scores in the range 10 to 30; forty are in the range 0 to -10; and about 12 or 13 have scores lower than -10. Table 4 shows these scores at a greater granularity. IrishKill and NIKill (animals slaughtered in Ireland and Northern Ireland) account for the largest weight contribution at 5.47%. FranMill (price of millet in France) accounts for 2.05%; partner1 (country exporting to Australia) contributes 2.03%; and GerRap (price of rapeseed meal in Germany) contributes 1.99% to the model. The final contributors are: the weight of monthly exports from Brazil (BrazilKg); February (month1); the amount the reporter exported in kg (ReporterKg); whether a country is exporting to Mexico; and 2014 (year4). For interactions, cumulative weight is used, similar to regression co-efficients.

Table 4. Top 10 Contributing Features

rank	feature	cumulative weight	contribution %
1	IrishKill	25.03578609	2.94
2	NIKill	21.53403839	2.53
3	FranMill	17.48791526	2.05
4	partner1	-17.31823953	2.03
5	GerRap	16.97575512	1.99
6	BrazilKg	16.40203897	1.93
7	month1	-16.19787237	1.90
8	ReporterKg	16.14323915	1.90
9	partner8	16.13083378	1.89
10	year4	-15.68426263	1.84

Feature Representation and Interaction Analysis. The above findings would suggest that *in our dataset*, that the supply coming from Ireland has the largest impact on trade. Perhaps this is the case in Europe, or perhaps we have richer data from Ireland. Further investigation is necessary. The price of feed in Germany and France stands to reason as having a large effect, as these are some of the largest countries in Europe. Finally, the amount in Kg the reporter is exporting having a large impact also stands to reason as this is what the model is developed to describe. More unusual findings were a particular year and month having a large impact. Seasonality effects trade but further investigations will have to be made into these time-points. Interactions also require further investigation as they are somewhat unintuitive, describing impact on the final hidden representation.

6 Conclusions and Future Work

Agri-analytics plays a significant role in decision making for today's agri sector. Determining multi-variate, unsupervised, feature interactions is extremely difficult, but can greatly aid understanding of agri data. Deep neural networks can extract complex multi-variate interactions via learned representations, but no generic method exists to interpret these representations. We presented the dCW method for summarisation of complex *deep* representations. Furthermore, we successfully applied our method to complex, *unsupervised* agri trade data and determined the most relevant input signals and interactions. Future work will extend dCW to discern each input's effect on individual variable reconstructions.

References

1. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al.: Greedy layer-wise training of deep networks. Advances in neural information processing systems 19, 153–160 (2007)
2. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13, 281–305 (Feb 2012)
3. Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., Bengio, Y.: Theano: a CPU and GPU math expression compiler. In: Proceedings of the Python for Scientific Computing Conference (SciPy) (Jun 2010), oral Presentation
4. Bottou, L.: Stochastic gradient learning in neural networks. Proceedings of Neuro-Nımes 91(8) (1991)
5. Garson, G.D.: Interpreting neural-network connection weights. AI Expert 6(4), 46–51 (Apr 1991), http://dl.acm.org/citation.cfm?id=129449.129452
6. Gevrey, M., Dimopoulos, I., Lek, S.: Review and comparison of methods to study the contribution of variables in artificial neural network models. Ecological modelling 160(3), 249–264 (2003)
7. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: International conference on artificial intelligence and statistics. pp. 249–256 (2010)
8. Hinton, G.: A practical guide to training restricted boltzmann machines. Momentum 9(1), 926 (2010)
9. McCarren, A., McCarthy, S., Sullivan, C.O., Roantree, M.: Anomaly detection in agri warehouse construction. In: Proceedings of the Australasian Computer Science Week Multiconference. pp. 17:1–17:10. ACSW '17, ACM (2017)
10. McKinney, W.: Data structures for statistical computing in python. In: van der Walt, S., Millman, J. (eds.) Proceedings of the 9th Python in Science Conference. pp. 51 – 56 (2010)
11. MongoDB: Pymongo (2016), [Online; accessed 12-December-2016]
12. O'Donoghue, J., Roantree, M.: A toolkit for analysis of deep learning experiments. In: International Symposium on Intelligent Data Analysis. pp. 134–145. Springer (2016)
13. O'Donoghue, J., Roantree, M., Boxtel, M.V.: A configurable deep network for high-dimensional clinical trial data. In: Neural Networks (IJCNN), 2014 International Joint Conference on. IEEE (July 2015)
14. Olden, J.D., Joy, M.K., Death, R.G.: An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. Ecological Modelling 178(3), 389–397 (2004)
15. Prechelt, L.: Early stopping-but when? In: Neural Networks: Tricks of the trade, pp. 55–69. Springer (1998)