DISTRIBUTION REGRESSION NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce our Distribution Regression Network (DRN) which performs regression from multiple input distributions to multiple output distributions. Through experiments on synthetic and real-world datasets, we show that compared to existing state-of-the-art methods, DRN learns with fewer model parameters. Furthermore, DRN generalizes the conventional multilayer perceptron (MLP). In the framework of MLP, each node encodes a real number, whereas in DRN, each node encodes a probability distribution.

1 INTRODUCTION

The field of regression analysis is largely established with methods ranging from linear least squares to multilayer perceptrons. However, the scope of the regression is mostly limited to real valued input features and output responses (Fiori et al., 2015; Marquardt, 1963). In this paper, we study the task of distribution-to-distribution regression where one regresses from input probability distributions to output probability distributions. Distribution-to-distribution regression (see work by Oliva et al. (2013)) has not been as widely studied compared to the related task of functional regression (Ferraty & Vieu, 2006). Nevertheless, there are many useful applications for regression on distributional objects. For instance, in the field of population study Perotti (1996) suggested that to predict economic growth from income distribution, the entire shape of the distribution is required instead a single real-valued measure like the Gini coefficient.

Variants of the distribution regression task have been explored in literature (Póczos et al., 2013; Oliva et al., 2014). Distribution to distribution regression has been studied by Oliva et al. (2013) where the data is assumed to be obtained through samples from underlying distributions. They proposed an instance-based learning method where a linear smoother estimator (LSE) is applied across the estimated input-output distributions. However at test evaluation, LSE has a complexity linear to the number of instances in the dataset and becomes computationally prohibitive when the data set is large. To that end, Oliva et al. (2015) developed the Triple-Basis Estimator (3BE) where the evaluation complexity is independent of the number of data through the use of basis representations of distributions and Random Kitchen Sink basis functions.

Lampert (2015) addressed a specific task of predicting the future state of a single time-varying probability distribution. Given a series of samples from previous time steps, the method predicts the distribution of the next time step by embedding each probability distribution in a Hilbert space to learn the dynamics.

Our Distribution Regression Network (DRN) is based on a completely different scheme of network learning, motivated by spin models in statistical physics and similar to artificial neural networks. In many variants of the artificial neural network, the network encodes real values in the nodes (Rumelhart et al., 1985; LeCun et al., 1989; Bengio, 2009). DRN generalizes the conventional MLP by encoding each node with a probability distribution. We experimentally demonstrate that compared to existing methods, DRN is able to learn with fewer model parameters and hence gives a more interpretable model. On the challenging task of predicting future stock distributions, DRN significantly outperforms the state-of-the-art methods. To the best of our knowledge, our model is the first to apply network learning to distribution-to-distribution regression where all nodes encode a probability distribution.

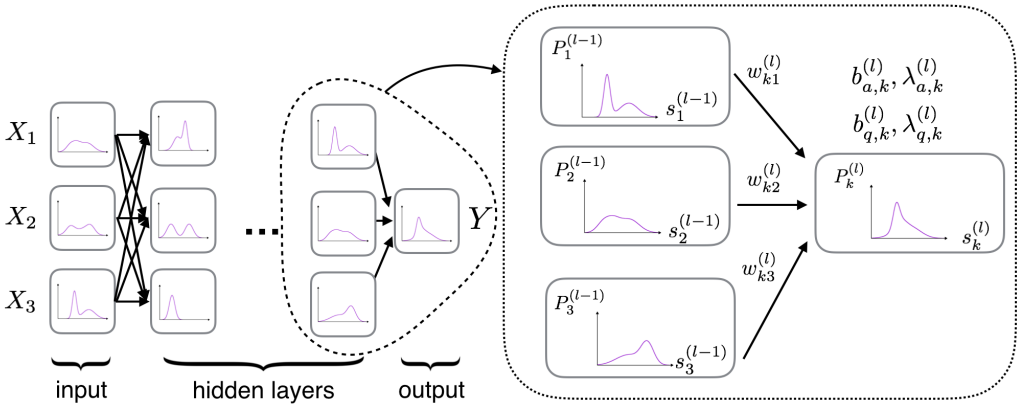


Figure 1: (Left) An example DRN with multiple input probability distributions and multiple hidden layers mapping to an output probability distribution. (Right) A connection unit in the network, with 3 input nodes in layer $l-1$ connecting to a node in layer l. Each node encodes a probability distribution, as illustrated by the pdf $P_k^{(l)}$. The tunable parameters are the connecting weights and the bias parameters at the output node.

2 Distribution Regression Network

Given a training dataset with M data points $\mathcal{D} = \{(X_1^1, \cdots, X_1^K, Y_1), \cdots, (X_M^1, \cdots, X_M^K, Y_M)\}$ where X_i^k and Y_i are univariate continuous distributions with compact support, the regression task is to learn the function f which maps the input distributions to the output distribution.

$$Y_i = f(X_i^1, \cdots, X_i^K) \tag{1}$$

No further assumptions are made on the form of the distribution. It is trivial to generalize our method to regress to multiple output distributions but for simplicity of explanation we shall restrict to single output regressions in the following discussions.

2.1 Forward Propagation

Fig. 1 illustrates how the regression in Eq. (1) is realized. DRN generalizes the traditional neural network structure by encoding each node with a continuous probability distribution and connecting the nodes with real-valued weights. The input data consists of one or more probability distributions which are fed into the first layer and propagated layerwise to the upper hidden layers. We emphasize our network is not a Bayesian network even though each node encodes a probability. Unlike bayes net where the conditional probability among variables are learnt by maximizing the likelihood over observed data, DRN regresses probability distributions using a feedforward network, similar to MLP.

At each node in the hidden layer, the probability distribution is computed from the probability distributions of the incoming nodes in the previous layer and the network parameters consisting of the weights and bias parameters (see right figure of Fig. 1). $P_k^{(l)}$ represents the pdf of the kth node in the lth layer and $P_k^{(l)}(s_k^{(l)})$ is the density of the pdf when the node variable is $s_k^{(l)}$.

Before obtaining the probability distribution $P_k^{(l)}$, we first compute its unnormalized form $\tilde{P}_k^{(l)}$. $\tilde{P}_k^{(l)}$ is computed by marginalizing over the product of the unnormalized conditional probability $\tilde{Q}(s_k^{(l)}|s_1^{(l-1)}, \cdots, s_n^{(l-1)})$ and the incoming node probabilities.

$$\tilde{P}_k^{(l)}\left(s_k^{(l)}\right) = \int_{s_1{}^{(l-1)}} \cdots \int_{s_n{}^{(l-1)}} \tilde{Q}\left(s_k^{(l)}|s_1^{(l-1)}, \cdots, s_n^{(l-1)}\right) P_1^{(l-1)}\left(s_1^{(l-1)}\right) \cdots P_n^{(l-1)}\left(s_n^{(l-1)}\right) ds_1^{(l-1)} \cdots ds_n^{(l-1)} \tag{2}$$

$$\tilde{Q}\left(s_k^{(l)}|s_1^{(l-1)},\cdots,s_n^{(l-1)}\right)=\exp\left[-E\left(s_k^{(l)}|s_1^{(l-1)},\cdots,s_n^{(l-1)}\right)\right] \tag{3}$$

$s_1^{(l-1)},\cdots,s_n^{(l-1)}$ represent the variables of the lower layer nodes and E is the energy given a set of node variables, which we define later in Eq. (4). The unnormalized conditional probability has the same form as the Boltzmann distribution in statistical mechanics, except that the partition function is omitted. This omission reduces the computational complexity of our model through factorization, shown later in Eq. (5).

Our energy function formulation is motivated by work on spin models in statistical physics where spin alignment to coupling fields and critical phenomena are studied (Lee et al., 2002; 2003; Katsura, 1962; Wu, 1982). Energy functions are also used in other network models where a scalar energy is associated to each configuration of the nodes (Teh et al., 2003; LeCun et al., 2007). While in energy-based models, the network parameters are learnt such that the observed configurations of the variables have lower energies than unobserved ones, the energy function used in DRN is part of the forward propagation process and is not directly optimized. For a given set of node variables, the energy function is

$$E\left(s_k^{(l)}|s_1^{(l-1)},\cdots,s_n^{(l-1)}\right)=\sum_i^n w_{ki}^{(l)}\left(\frac{s_k^{(l)}-s_i^{(l-1)}}{\Delta}\right)^2+b_{q,k}^{(l)}\left(\frac{s_k^{(l)}-\lambda_{q,k}^{(l)}}{\Delta}\right)^2 + b_{a,k}^{(l)}\left|\frac{s_k^{(l)}-\lambda_{a,k}^{(l)}}{\Delta}\right| \tag{4}$$

$w_{ki}^{(l)}$ is the weight connecting the ith node in the lower layer to the upper layer node. $b_{q,k}^{(l)}$ and $b_{a,k}^{(l)}$ are the values of the quadratic and absolute bias terms which act at the positions $\lambda_{q,k}^{(l)}$ and $\lambda_{a,k}^{(l)}$ respectively. The support length of the distribution is given by Δ. All terms in Eq. (4) are normalized by the support length so that the energy function is invariant with respect to the support. Eq. (2) can be factorized such that instead of having multidimensional integrals, there are n univariate integrals:

$$\begin{aligned}\tilde{P}_k^{(l)}\left(s_k^{(l)}\right)&=\exp\left(B\left(s_k^{(l)}\right)\right)\int_{s_1{}^{(l-1)}}\cdots\int_{s_n^{(l-1)}}P_1^{(l-1)}\left(s_1^{(l-1)}\right)\cdots P_n^{(l-1)}\left(s_n^{(l-1)}\right)\\&\qquad\exp\left[-\sum_i^n w_{ki}^{(l)}\left(\frac{s_k^{(l)}-s_i^{(l-1)}}{\Delta}\right)^2\right]ds_1^{(l-1)}\cdots ds_n^{(l-1)}\\&=\exp\left(B\left(s_k^{(l)}\right)\right)\prod_i^n\left\{\int_{s_i^{(l-1)}}P_i^{(l-1)}\left(s_i^{(l-1)}\right)\exp\left[-w_{ki}^{(l)}\left(\frac{s_k^{(l)}-s_i^{(l-1)}}{\Delta}\right)^2\right]ds_i^{(l-1)}\right\}\end{aligned} \tag{5}$$

where $B(s_k^{(l)})$ captures the bias terms of the energy function in Eq. (4).

$$B\left(s_k^{(l)}\right)=-b_{q,k}^{(l)}\left(\frac{s_k^{(l)}-\lambda_{q,k}^{(l)}}{\Delta}\right)^2-b_{a,k}^{(l)}\left|\frac{s_k^{(l)}-\lambda_{a,k}^{(l)}}{\Delta}\right| \tag{6}$$

Finally, the probability distribution from Eq. (2) is normalized.

$$P_k^{(l)}\left(s_k^{(l)}\right)=\frac{\tilde{P}_k^{(l)}\left(s_k^{(l)}\right)}{\int_{s_k^{(l)\prime}}\tilde{P}_k^{(l)}\left(s_k^{(l)\prime}\right)ds_k^{(l)\prime}} \tag{7}$$

The propagation of probability distributions within a connection unit forms the basis for forward propagation. Forward propagation is performed layerwise from the input layer using Eq. (2) to (7).

2.1.1 Propagation Properties

The forward propagation in DRN has some important properties. Fig. 2 illustrates the propagation behavior for a connection unit with one input node where the bias values $b_{a,k}^{(l)}$ and $b_{q,k}^{(l)}$ are set as zero.

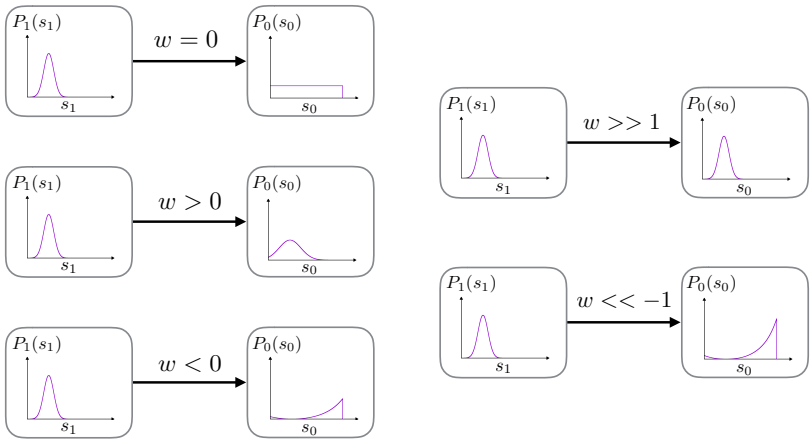


Figure 2: Propagation behavior for a connection unit with one input node. The biases are set as zero in these examples. When weight is zero, the output distribution is flat. Positive weights causes the output distribution to have the same peak position as the input distribution while negative weights causes the output pdf to 'repel' away from the input peak. When the weight is a sufficiently large positive number, the propagation tends towards the identity mapping.

When the weight is zero, the output distribution is flat and the output distribution is independent of the input. With a positive weight, the output distribution is 'attracted' to the peak of the input distribution whereas a negative weight causes the output distribution to be 'repelled' away from the input peak. In addition, the weight magnitude represents the strength of the 'attraction' or 'repulsion'. When the weight is a sufficiently large positive number, the propagation tends towards the identity mapping (top right example in Fig. 2). The implication is that like in neural networks, a deeper network should have at least the same complexity as a shallow one, as the added layers can produce the identity function. Conversely, a small positive weight causes the output peak to be at the same position as the input peak, but with more spread (second example on left column of Fig. 2).

The remaining absolute and quadratic bias terms in Eq. (4) have a similar role as the bias in a traditional neural network. Depending on the bias values $b_{a,k}^{(l)}$ and $b_{q,k}^{(l)}$, the bias terms act as attractors or repellers from the positions defined by $\lambda_{a,k}^{(l)}$ and $\lambda_{q,k}^{(l)}$ respectively. The weight and bias values play a similar role as the inverse temperature in the Boltzmann distribution in statistical physics (Lee et al., 2002; Wu, 1982).

2.2 Network cost function

The cost function of the network given a set network parameters is measured by the Jensen-Shannon (JS) divergence between the label and output distributions. The JS divergence is given by $D_{JS}(Y||\hat{Y}) = \frac{1}{2}D_{KL}(Y||W) + \frac{1}{2}D_{KL}(\hat{Y}||W)$, where $W = \frac{1}{2}(Y + \hat{Y})$ and D_{KL} is the Kullback-Liebler divergence. The Jensen-Shannon divergence is a suitable cost function as it is symmetric and is bounded. The network cost function C_{net} is the average D_{JS} over all M training data: $C_{net} = \frac{1}{M}\sum_i^M D_{JS}(Y_i||\hat{Y}_i)$.

2.3 Discretization of Probability Distributions

In our experiments, the integrals in Eq. (5) and (7) are performed numerically. This is done through discretization from continuous probability density functions (pdf) to discrete probability mass functions (pmf). Given a continuous pdf with finite support, the range of the continuous variable is partitioned into q equal widths and the probability distribution is binned into the q states. The estimation error arising from the discretization step will decrease with larger q.

2.4 OPTIMIZATION BY BACKPROPAGATION

The network cost is a differentiable function over the network parameters. We derive the cost gradients similar to backpropagation in neural networks (Rumelhart et al., 1988). We use chain rule to derive at each node a q-by-q matrix which denotes the derivative of the final layer node distribution with respect to the current node distribution.

$$\frac{\partial P_1^{(L)}\left(s_1^{(L)}\right)}{\partial P_k^{(l)}\left(s_k^{(l)}\right)} = \sum_i^n \frac{\partial P_1^{(L)}\left(s_1^{(L)}\right)}{\partial P_i^{(l+1)}\left(s_i^{(l+1)}\right)} \frac{\partial P_i^{(l+1)}\left(s_i^{(l+1)}\right)}{\partial P_k^{(l)}\left(s_k^{(l)}\right)} \tag{8}$$

where $P_1^{(L)}\left(s_1^{(L)}\right)$ is the final layer output probability distribution. With the derivative $\frac{\partial P_1^{(L)}\left(s_1^{(L)}\right)}{\partial P_k^{(l)}\left(s_k^{(l)}\right)}$, the cost gradients for all network parameters can be obtained. The network weights $w_{ki}^{(l)}$ and bias magnitudes $b_{a,k}^{(l)}$, $b_{q,k}^{(l)}$ are randomly initialized with a uniform distribution, though other initialization methods are also feasible. The bias positions $\lambda_{a,k}^{(l)}$ and $\lambda_{q,k}^{(l)}$ are uniformly sampled from the range corresponding to the support of the distributions.

3 EXPERIMENTS

We evaluate DRN on synthetic and real-world datasets and compare its performance to the state-of-the-art 3BE method and a fully-connected multilayer perceptron (MLP). For each of the datasets, DRN achieves similar or higher accuracy with significantly fewer model parameters, implying that the parameters are encoding the distribution regression task more efficiently. Additionally, we show for the second dataset that the model parameters learnt by DRN can be interpreted to give insights on the regression task.

In MLP, each discretized probability mass function is represented by q nodes. The MLP consists of fully connected hidden layers with ReLU units and a softmax final layer, and is optimized with mean squared error using Adam. Unlike DRN and MLP where the distribution pdfs are directly used by the methods, 3BE assumes the input and output distributions are observed indirectly through i.i.d. samples. Hence, for the first two datasets we provide 3BE with sufficient samples from the underlying distribution such that errors from density estimation are minimal.

3.1 SYNTHETIC DATA

The first experiment involves a synthetic dataset similar to the one used by Oliva et al. (2013) but with increased complexity. We first generate two truncated gaussians by sampling their means $\mu_1 \sim \text{Unif}[0.1, 0.4]$, $\mu_2 \sim \text{Unif}[0.6, 0.9]$, standard deviations $\sigma_1, \sigma_2 \sim \text{Unif}[0.05, 0.1]$. The input pdf is $X(s) = \gamma g(s; \mu_1, \sigma_1) + (1-\gamma) g(s; \mu_2, \sigma_2)$ and the output pdf is $Y(s) = \gamma g(s; h(\mu_1, 0.1, 0.4), h(\sigma_1, 0.05, 0.1)) + (1-\gamma) g(s; h(\mu_2, 0.6, 0.9), h(\sigma_2, 0.05, 0.1))$ where g is the truncated normal pdf with support of [0,1], $\gamma \sim \text{Unif}[0, 1]$, and h is

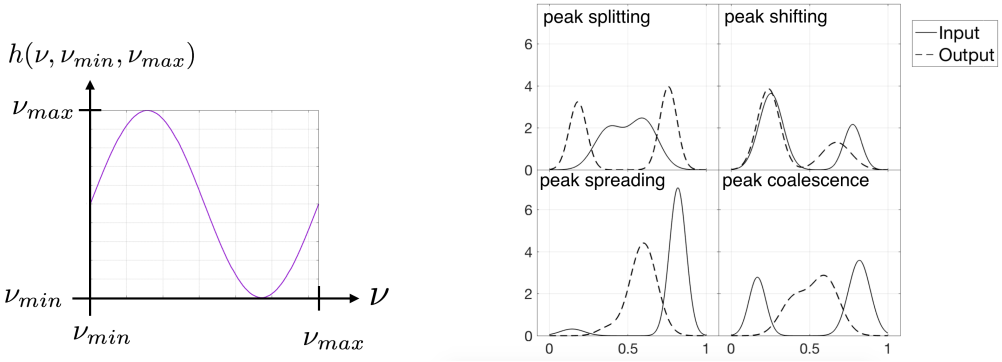
$$h(\nu, \nu_{min}, \nu_{max}) = \nu_{min} + \frac{\sin\left(2\pi \frac{\nu - \nu_{min}}{\nu_{max} - \nu_{min}}\right) + 1}{2} \times (\nu_{max} - \nu_{min}) \tag{9}$$

The function h transforms the means and standard deviations using the non-linear function shown in Fig. 3a. The transformation is such that the two gaussian means will remain in their respective ranges. The sample input-output data pairs in Fig. 3b shows the complexity of the regression task with various effects like peak splitting and peak spreading. 1000 training data and 1000 testing data were created to evaluate the regression methods.

For DRN and MLP, the pdfs are discretized into $q = 100$ states and for 3BE, 10,000 samples from each data distribution are generated. While 3BE gives a continuous distribution as the output, DRN and MLP output the discrete pmf and require conversion to continuous pdf. Following Oliva et al. (2014), the regression performance on the test set is measured by the $L2$ loss between the continuous

predicted distribution, $\hat{Y}(s)$ and the true distribution $Y(s)$:

$$||\hat{Y}(s) - Y(s)||_2 = \sqrt{\int_s \left(\hat{Y}(s) - Y(s)\right)^2} \tag{10}$$



(a) (b)

Figure 3: (a) Nonlinear transformation of the input means and standard deviations of gaussians for the synthetic dataset. (b) Example input-output pairs from the synthetic data, illustrating the complexity of the regression task.

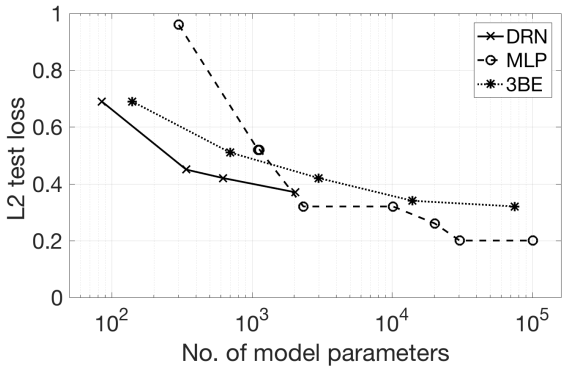


Figure 4: Comparison of $L2$ loss on the synthetic data test set. Note that the x-axis denotes the number of model parameters using the log scale.

We study how the regression accuracy varies with respect to the number of model parameters. For DRN and MLP, the number of parameters are varied using different depths and widths of the networks and for 3BE, we vary the number of Random Kitchen Sink features. Fig. 4 shows the $L2$ loss on the test set as we vary the number of model parameters. Note that the x-axis is presented on the log scale. DRN's test performance is comparable to the other methods, though not as well as MLP.

3.2 Ornstein-Uhlenbeck Process

For the next experiment, we demonstrate the interpretability of the model learnt by DRN. Because of its rich propagation behavior (ref. Fig. 2), DRN uses much fewer parameters than the baseline methods and provides an explanable model. We evaluate our model on data generated from the stochastic Ornstein-Uhlenbeck (OU) process (Uhlenbeck & Ornstein, 1930) which combines the notion of random walk with a drift towards the long-term mean. The OU process has wide-ranging applications. In the commodity market, prices exhibit mean-reverting patterns due to market forces and hence modelling the prices with the OU process helps form valuation strategies (Schwartz &

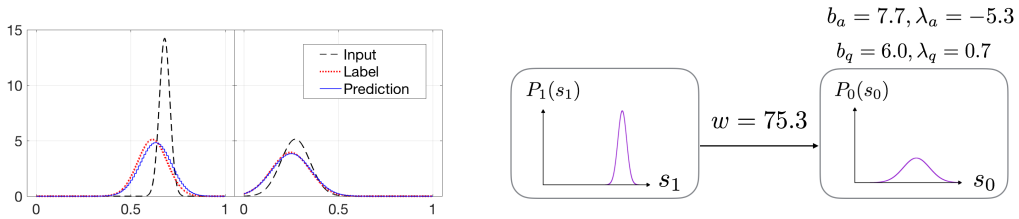
Smith, 2000; Zhang et al., 2012). It is also used in quantitative biology to model phenotypic traits evolution (Bartoszek et al., 2016).

The OU process is described by a time-varying gaussian pdf: With the long-term mean set at zero, the pdf has a mean of $\mu(t) = y\exp(-\theta t)$ and variance of $\sigma^2(t) = \frac{D(1-e^{-2\theta t})}{\theta}$. t represents time, y is the initial point mass position, and D and θ are the diffusion and drift coefficients respectively. The regression task is to map from an initial gaussian distribution at t_{init} to the resulting distribution after some time step Δt. The gaussian distributions are truncated with support of $[0, 1]$. With different sampled values for $y \in [0.3, 0.9]$ and $t_{init} \in [0.01, 2]$, pairs of distributions are created for $\Delta t = 1$, $D = 0.003$ and $\theta = 0.1$. For DRN and MLP, $q = 100$ was used for discretization of the pdfs while 10,000 samples were taken for each distribution to train 3BE.

Table 1: Comparison of $L2$ test loss and the number of model parameters used for the Ornstein-Uhlenbeck data.

	$L2$ test loss	Model description	No. of parameters
DRN	0.1441 ± 0.0010	No hidden layer	4
MLP	0.1475 ± 0.0005	1 hidden layer of 3 nodes	703
3BE	0.1255 ± 0.0083	16 projection coefficients, 17 Random Kitchen Sink features	272

We compare the number of model parameters required to achieve a small $L2$ test loss with 100 training data. Table 1 shows that a single-layer DRN with 4 parameters (illustrated in Fig. 5b) performs similarly as MLP and 3BE. MLP requires 1 fully-connected hidden layer with 3 nodes, with a total of 703 network parameters. 3BE requires 64 projection coefficients for both input and output distributions and 17 Random Kitchen Sink features, resulting in 272 model parameters.



(a) (b)

Figure 5: (a) The regression by DRN on two test samples. (b) The learnt parameters for the trained DRN gives insight on the OU process. The positive weight of 75.3 reflects the positive correlation between input and output peak positions and that the peak spreads out over time. The negative position of the absolute bias (λ_a) shows that the output peak is displaced leftwards of the input peak.

The regression by DRN on two random test samples are shown in Fig. 5a and we see that DRN is able to demonstrate the OU process. Fig. 5b shows the DRN network parameters after training and they can be interpreted to give insights on the regression task. The weight parameter is positive, hence the output peak position is positively correlated to the input peak position. Moreover, $w = 75.3$ is such that the network mimics the diffusion property of the OU process. The bias position λ_a is negative and its magnitude is 5 times the distribution support, causing the output peak to be displaced leftwards of the input peak. These two observations reflect the random walk and mean-reverting properties of the OU process. In contrast, 3BE and MLP use much more model parameters which are difficult to interpret.

3.3 Stock Data

Lastly, we demonstrate that DRN can be useful for an important real-world problem and overperforms 3BE and MLP in terms of prediction accuracy. With greater integration of the global stock markets, there is significant co-movement of stock indices (Hamao et al., 1990; Chong et al., 2008).

Table 2: Comparison of log-likelihood on the stock data and the number of model parameters.

	Log-likelihood on test set	Model description	No. of parameters
DRN	474.1 ± 3.2	No hidden layer (Fig. 6)	7
MLP	471.4 ± 3.5	1 hidden layer of 10 nodes	4110
3BE	446.2 ± 3.9	1009 projection coefficients, 10 Random Kitchen Sink features	10090

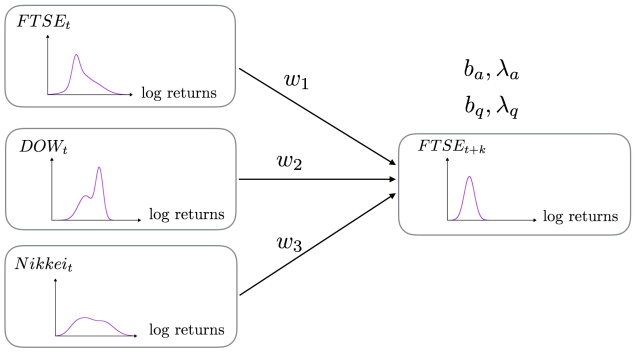


Figure 6: Single-layer network used in DRN for the stock dataset with 7 model parameters (3 weights, 4 bias parameters).

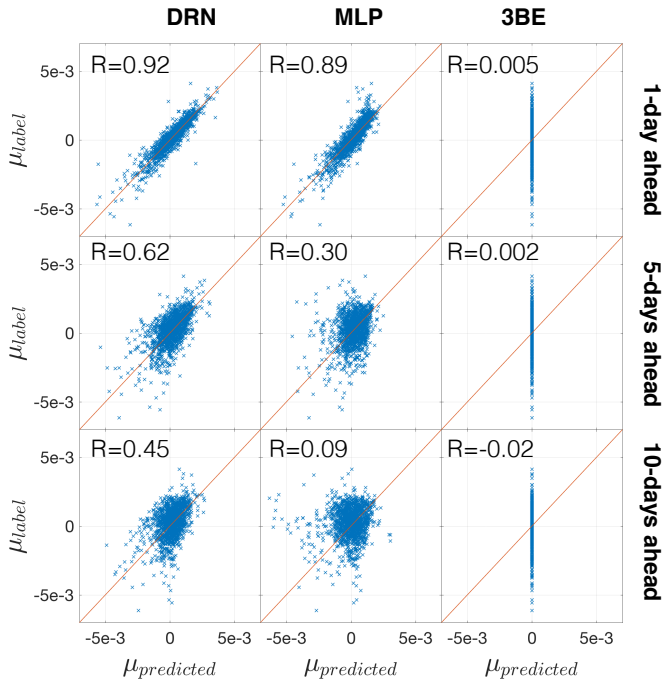
In a study by Vega & Smolarski (2012), it was found that stock market indices show significant correlation. Specifically, the previous day stock returns of the Nikkei and Dow Jones Industrial Average (Dow) are good predictors of the FTSE return. Modelling the co-movement of global stock indices has its value as it facilitates investment decisions.

While existing research has primarily focused on the movement of stock returns of the various indices, we predict the future distribution of returns over the constituent companies as it provides more information than just the mean and variance. Our regression task is as follows. Given the current day's distribution of returns of the FTSE, Dow and Nikkei, predict the distribution of returns for FTSE k days later: $FTSE_{t+k} = f(FTSE_t, Dow_t, Nikkei_t)$, where $FTSE_t$, Dow_t and $Nikkei_t$ represent the distribution of logarithmic returns of the respective indices at day t. Let V_t and V_{t-1} represent the closing price of a company's stock at day t and $t-1$ respectively. The logarithmic return for the company's stock at day t is given by $\ln(V_t/V_{t-1})$. The number of companies for the FTSE, DOW and NIKKEI are 100, 30 and 225 respectively.

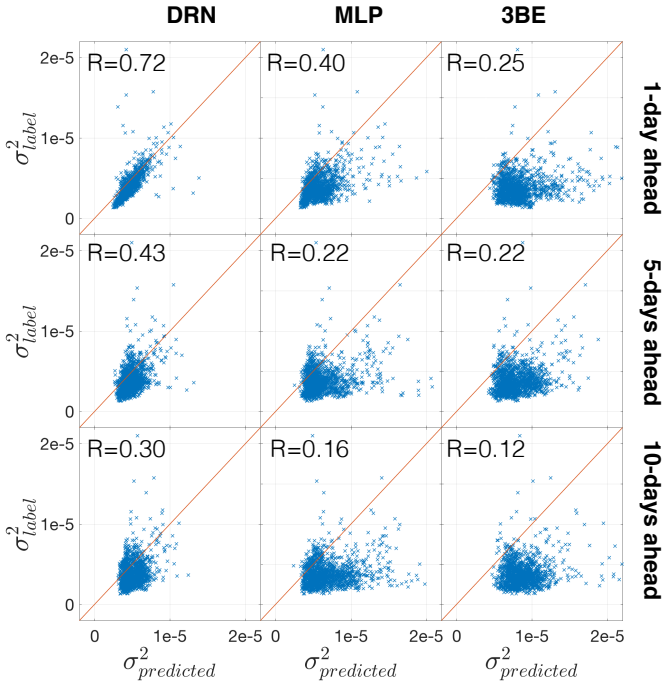
The stock data consists of 9 years of daily returns from January 2007 to December 2015. To adapt to changing market conditions, we use a sliding-window training scheme where the data is split into windows of training, validation and test sets and moved foward in time Kaastra & Boyd (1996). A new window is created and the network is retrained after every 300 days (ie. the size of test set). For each test set, the previous 500 and 100 days were used for training and validation. To reduce the noise in the data, we performed exponential window averaging on the price series for each stock with a window of 50 days, following common practice (Murphy, 1999). The logarithmic returns are then calculated using the smoothed price series. The logarithmic returns of the constituent company stocks form the samples for the distributions of the returns. For DRN and MLP, the pdf is estimated using kernel density estimation with a Gaussian kernel function with bandwidth of 0.001 and $q = 100$ was used for discretization of the pdf. 3BE was originally formulated for single input distributions, so we concatenated the basis coefficients obtained from the three input distributions.

As we do not have the underlying true pdf for this real-world dataset, the regression performance is measured by the log-likelihood of the test set samples. Table 2 shows the mean test log-likelihoods, where higher log-likelihood is favorable. Interestingly, the single-layer network in DRN (see Fig. 6) was sufficient to perform well, using just 7 network parameters. In comparison, MLP and 3BE require 4110 and 10090 parameters respectively.

To visualize the regression results on the test set, we compare for each day the first two moments (mean and variance) of the predicted output distribution and the ground truth label (see 1-day ahead



(a)



(b)

Figure 7: Comparison of the (a) mean and (b) variance of the label distributions and predicted distributions on the test set, for various k-days ahead predictions. The diagonal line represents a perfect fit where the predicted and labelled moments are equal. DRN outperforms the rest as its data points are closest to the diagonal line and it has the highest correlation coefficient (R) for all experiments.

panels of Fig. 7a and Fig. 7b). Each point on the plots represents one test data and we show the Pearson correlation coefficients between the predicted and labelled moments at the top-left corner. DRN has the best regression performance as the points lie close to the target diagonal line where the output and label moments are equal, and the correlation values are highest. 3BE's predicted distributions have very similar means regardless of the input data, showing that the regression fit is poor.

3.3.1 PREDICTING SEVERAL DAYS AHEAD

As an extension, instead of predicting next-day distributions, we want to predict the FTSE returns distribution several days ahead. The second and third horizontal panels of Fig. 7a and Fig. 7b show the moment plots for 5 and 10 days ahead respectively. Expectedly, the performance deterioriates as the number of days ahead increases. Still, DRN outperforms the rest as shown by the mean and variance plots and the correlation values. Fig. 8 summarizes the results by showing the average absolute error of the mean and variance as the number of days-ahead increases. For all experiments, DRN consistently has the lowest error and it is noteworthy that it achieves this performance with just 7 network parameters.

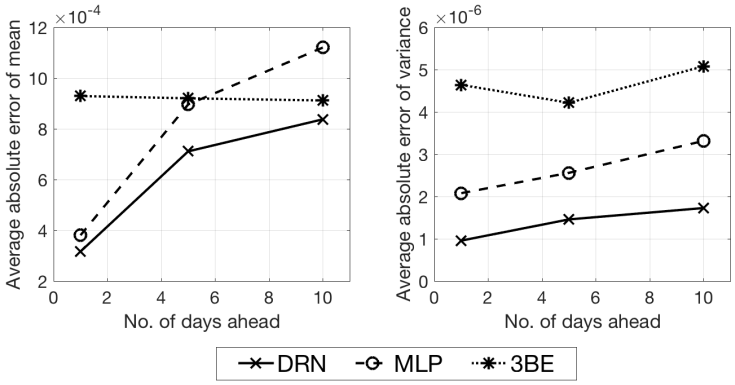


Figure 8: The average absolute error of mean and variance across the three methods for prediction with varying number of days-ahead. DRN's error is consistently the lowest compared to the benchmark methods. The standard errors are smaller than the data point symbols.

4 CONCLUSION

The distribution-to-distribution regression task has not been widely studied in literature but is relevant has many potential applications ranging from population studies to stock market prediction. In this paper, we introduce our Distribution Regression Network which generalizes the MLP framework by encoding a probability distribution in each node. Through experiments on synthetic and real-world datasets, we show that DRN performs comparable or better than the state-of-the-art methods with fewer model parameters. This implies that DRN's parameters are able to encode the distribution task more efficiently. Additionally, using few parameters allows DRN to provide interpretable models that reflect properties of the regression task. For future work, we would like to study what class of problems DRN can solve. Extensions may also be made for regressing multivariate distributions.

REFERENCES

Krzysztof Bartoszek, Sylvain Glémin, Ingemar Kaj, and Martin Lascoux. The ornstein-uhlenbeck process with migration: evolution with interactions. *arXiv preprint arXiv:1607.07970*, 2016.

Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

Terence Tai-Leung Chong, Ying-Chiu Wong, and Isabel Kit-Ming Yan. International linkages of the japanese stock market. *Japan and the World Economy*, 20(4):601–621, 2008.

Frédéric Ferraty and Philippe Vieu. *Nonparametric functional data analysis: theory and practice*. Springer Science & Business Media, 2006.

Simone Fiori, Tianxia Gong, and Hwee Kuan Lee. Bivariate nonisotonic statistical regression by a lookup table neural system. *Cognitive Computation*, 7(6):715–730, 2015.

Yasushi Hamao, Ronald W Masulis, and Victor Ng. Correlations in price changes and volatility across international stock markets. *Review of Financial studies*, 3(2):281–307, 1990.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Iebeling Kaastra and Milton Boyd. Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, 10(3):215–236, 1996.

Shigetoshi Katsura. Statistical mechanics of the anisotropic linear heisenberg model. *Physical Review*, 127(5):1508, 1962.

Christoph H Lampert. Predicting the future behavior of a time-varying probability distribution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 942–950, 2015.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Yann LeCun, Sumit Chopra, Marc'Aurelio Ranzato, and Fu Jie Huang. Energy-based models in document recognition and computer vision. In *icdar*, volume 7, pp. 337–341, 2007.

HK Lee, DP Landau, and TC Schulthess. Monte carlo simulations of phase transitions in rb 2 mnf 4. *Journal of applied physics*, 93(10):7643–7645, 2003.

Hwee Kuan Lee, TC Schulthess, DP Landau, Gregory Brown, John Philip Pierce, Z Gai, GA Farnan, and J Shen. Monte carlo simulations of interacting magnetic nanoparticles. *Journal of applied physics*, 91(10):6926–6928, 2002.

Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

John J Murphy. Technical analysis of the futures markets: A comprehensive guide to trading methods and applications, new york institute of finance, 1999.

Junier Oliva, William Neiswanger, Barnabás Póczos, Eric Xing, Hy Trac, Shirley Ho, and Jeff Schneider. Fast function to function regression. In *Artificial Intelligence and Statistics*, pp. 717–725, 2015.

Junier B Oliva, Barnabás Póczos, and Jeff G Schneider. Distribution to distribution regression. In *ICML (3)*, pp. 1049–1057, 2013.

Junier B Oliva, Willie Neiswanger, Barnabás Póczos, Jeff G Schneider, and Eric P Xing. Fast distribution to real regression. In *AISTATS*, pp. 706–714, 2014.

Roberto Perotti. Growth, income distribution, and democracy: what the data say. *Journal of Economic growth*, 1(2):149–187, 1996.

Barnabás Póczos, Aarti Singh, Alessandro Rinaldo, and Larry A Wasserman. Distribution-free distribution regression. In *AISTATS*, pp. 507–515, 2013.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

Eduardo Schwartz and James E Smith. Short-term variations and long-term dynamics in commodity prices. *Management Science*, 46(7):893–911, 2000.

Yee Whye Teh, Max Welling, Simon Osindero, and Geoffrey E Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4(Dec):1235–1260, 2003.

George E Uhlenbeck and Leonard S Ornstein. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

Jose G Vega and Jan M Smolarski. Forecasting ftse index using global stock markets. *International Journal of Economics and Finance*, 4(4):3, 2012.

Fa-Yueh Wu. The potts model. *Reviews of modern physics*, 54(1):235, 1982.

Bowen Zhang, Lech Aleksander Grzelak, and Cornelis Willebrordus Oosterlee. Efficient pricing of commodity options with early-exercise under the ornstein–uhlenbeck process. *Applied Numerical Mathematics*, 62(2):91–111, 2012.