Uncertainty Quantification of Spatio-Temporal Flows with Deep Learning

Matthew Dixon[1]

Illinois Institute of Technology

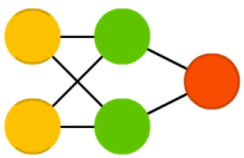
October 30th, 2017
Synthesis of Statistics, Data Mining and Environmental Sciences in Pursuit of Knowledge Discovery
BIRS-CMO

[1] M.F. Dixon, N. Polson and V. Sokolov, Deep Learning for Spatio-Temporal Modeling: Dynamic Traffic Flows and High Frequency Trading, arXiv:1705.09851

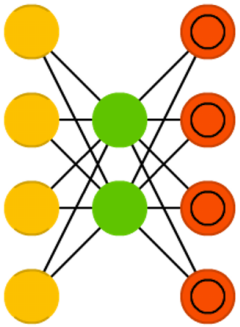
Introduction to Deep Learning

- Machine learning falls into the algorithmic class [Breiman, 2001] of reduced model estimation procedures which treats the data generation process as an unknown.
- Deep learning is a form of machine learning that uses hierarchical layers of abstraction to represent high-dimensional nonlinear predictors.
- Traditional fit metrics, such as R^2, $t-$values, p-values, and the notion of *statistical significance* has been replaced in the machine learning literature by out-of-sample forecasting and understanding the bias-variance trade-off.
- Deep learning is data-driven and focuses on finding structure in large data sets. The main tools for variable or predictor selection are *regularization* and *dropout*.

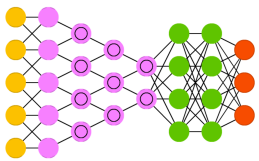
Deep Architectures in TensorFlow



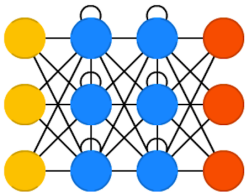
feed forward



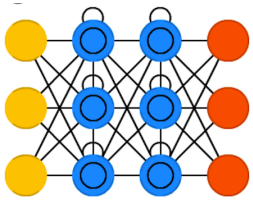
auto-encoder



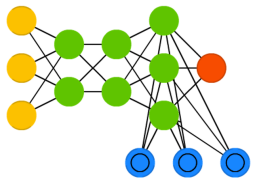
convolution



recurrent



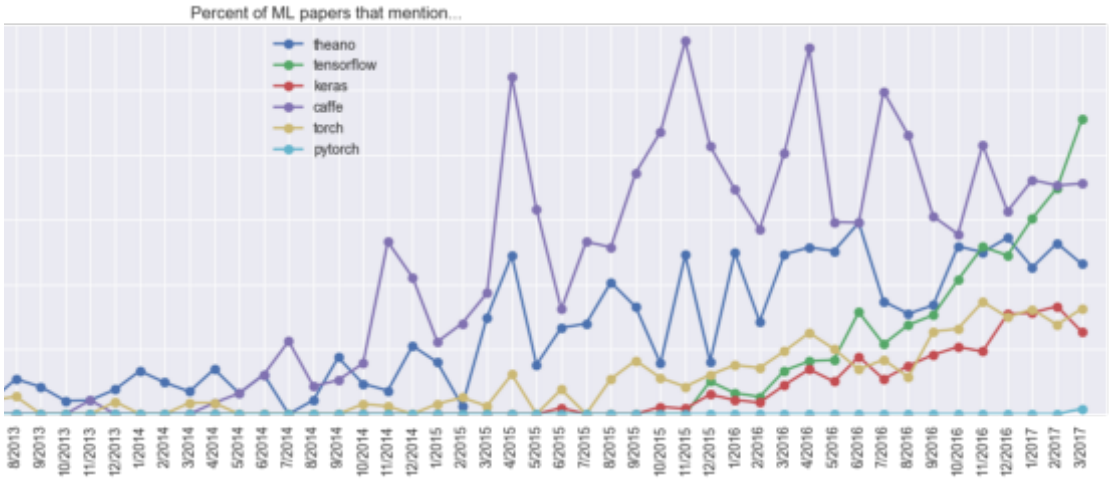
Long / short term memory



neural Turing machines

Figure: Most commonly used deep learning architectures for modeling. Source: `http://www.asimovinstitute.org/neural-network-zoo`

Growth of TensorFlow



% of papers	framework	has been around for (months)
9.1	tensorflow	16
7.1	caffe	37
4.6	theano	54
3.3	torch	37
2.5	keras	19
1.7	matconvnet	26
1.2	lasagne	23
0.5	chainer	16
0.3	mxnet	17
0.3	cntk	13
0.2	pytorch	1
0.1	deeplearning4j	14

Source : Andrej Karpathy's arXiv-sanity database

Tensorflow build for Intel Xeon Phi: https://github.com/tensorflow/tensorflow.git

- Python examples: https://github.com/Quiota/tensorflow
- R examples: 2017 Google Summer of Code Statistical Computing Project in R (with Lan Wei), https://github.com/lweicdsor/OSTSC

Machine Learning

- Machine learning addresses a fundamental prediction problem: Construct a nonlinear predictor, $\hat{Y}(X)$, of an output, Y, given a high dimensional *input matrix* $X = (X_1, \ldots, X_P)$ of P variables.
- Machine learning can be simply viewed as the study and construction of an input-output map of the form

$$Y = F(X) \text{ where } X = (X_1, \ldots, X_P).$$

- The output variable, Y, can be continuous, discrete or mixed.
- For example, in a classification problem, $F : X \to Y$ where $Y \in \{1, \ldots, K\}$ and K is the number of categories. When Y is a continuous vector and f is a semi-affine function, then we recover the linear model

$$Y = AX + b.$$

Deep Predictors

Definition (Deep Predictor)

A deep predictor is a particular class of multivariate function $F(X)$ constructed using a sequence of L layers via a composite map

$$\hat{Y}(X) := F_{\boldsymbol{\theta}}(X) = \left(f^{L}_{W^L, b^L} \ldots \circ f^{1}_{W^1, b^1} \right)(X).$$

- $f^{l}_{W^l, b^l}(X) := f^{l}(W^l X + b^l)$ *is a semi-affine function, where* f^l *is univariate and continuous.*
- *The parameter set* $\boldsymbol{\theta} = (W, b)$*, where* $W = (W^1, \ldots, W^L)$ *and* $b = (b^1, \ldots, b^L)$ *are weight matrices and offsets respectively.*

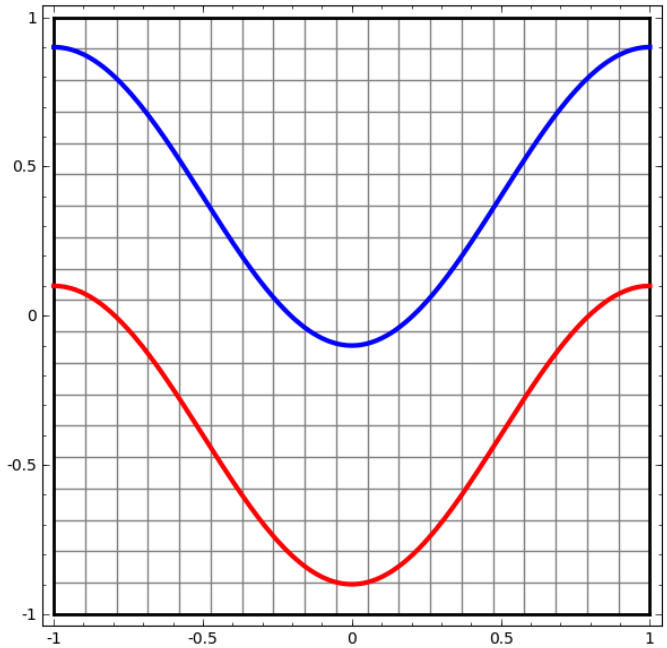
Deep Predictors

- The structure of a deep prediction rule can be written as a hierarchy of $L-1$ unobserved layers, Z^l, given by

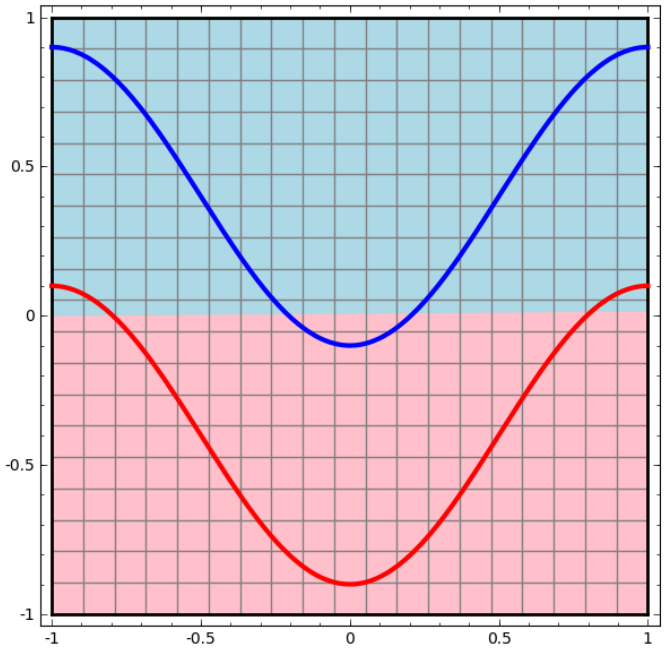
$$\begin{aligned}
\hat{Y}(X) &= f^L(Z^{L-1}), \\
Z^0 &= X, \\
Z^1 &= f^1\left(W^1 Z^0 + b^1\right), \\
Z^2 &= f^2\left(W^2 Z^1 + b^2\right), \\
&\dots \\
Z^{L-1} &= f^{L-1}\left(W^{L-1} Z^{L-2} + b^{L-1}\right).
\end{aligned}$$

- When Y is numeric, the output function $f^L(X)$ is given by the semi-affine function $f^L(X) := f^L_{W^L, b^L}(X)$.
- When Y is categorical, $f^L(X)$ is a softmax function.
- $f(x)$ are 'activation' functions, e.g. $tanh(x)$, rectified linear unit $max(x, 0)$.

Why use hidden layers?



Problem: classify whether the curve is red or blue

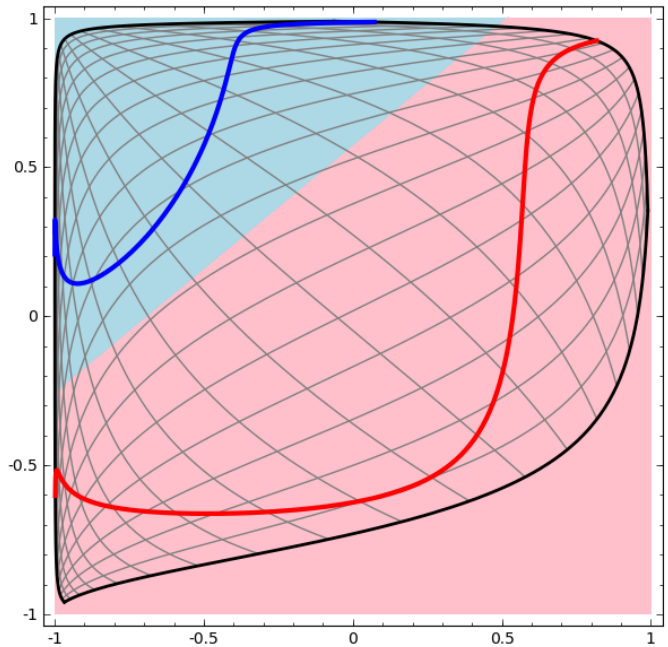


Solution using a linear method

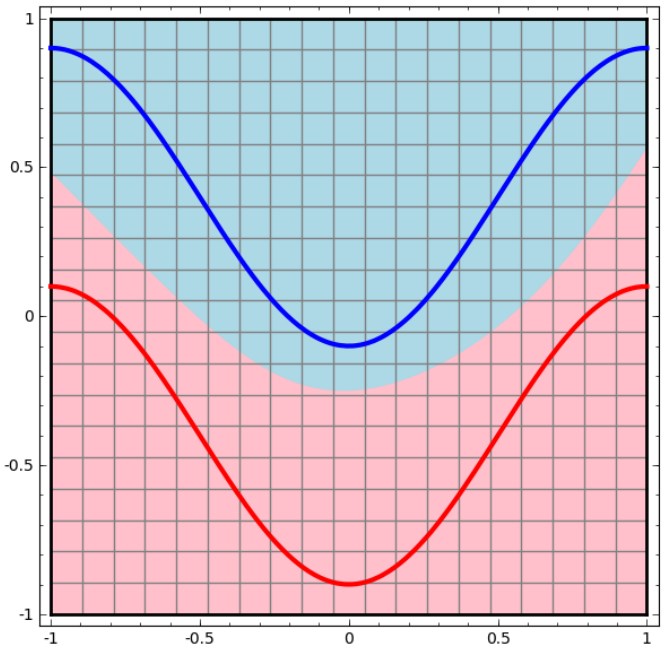
Figure: Image source: Chris Olah, Google Brain.

Why use hidden layers?

Answer: To perform translations of the input space that enable linear separability.



Transformation of the input space using a hidden layer



Result of classification using a hidden layer

Figure: Image source: Chris Olah, Google Brain.

Bayesian Deep Learnining

- Training data of input-output pairs $\mathcal{D} = \{y_i, x_i\}_{i=1}^N$
- The goal is to find the deep predictor $\hat{Y} = F_{\hat{\boldsymbol{\theta}}}(X)$
- The loss function is the negative log probability $\mathcal{L}(Y, \hat{Y}) = -\log\ p(Y \mid \hat{Y})$
- Deep predictors are maximum a posteriori (MAP) estimators

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto p(Y|Y_{\boldsymbol{\theta}}(X))p(\boldsymbol{\theta}).$$

Bayesian Deep Learning

- Training requires the solution of a regularized non-linear optimization problem

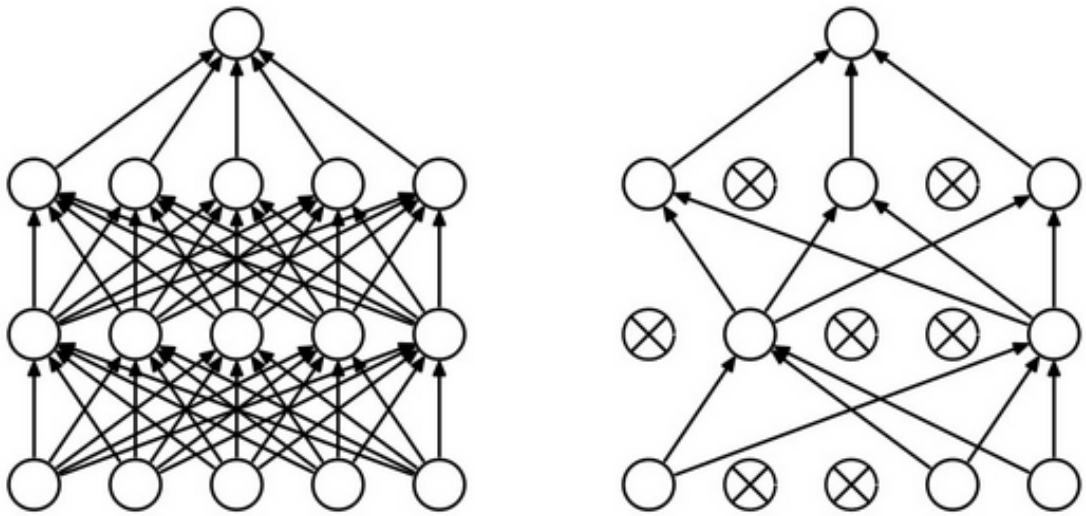
$$\hat{\boldsymbol{\theta}} := \underset{\boldsymbol{\theta}}{\text{argmin}} - log\ p(\boldsymbol{\theta}|\mathcal{D}) \tag{1}$$

$$\approx \sum_{i=1}^{n} \mathcal{L}(y_i, y_{\boldsymbol{\theta}}(x_i)) + \lambda\phi(\boldsymbol{\theta}). \tag{2}$$

- $\phi(\boldsymbol{\theta})$ introduces a *bias-variance tradeoff* controlled by a regularization parameter λ
- The overall objective function is closely related to ridge regression with a g-prior [Polson & Sokolov, 2017]
- See Gal, Uncertainty Quantification in Deep Learning, Ph.D Thesis, University of Cambridge, 2016.

Drop-Out

- Dropout is a model or variable selection technique which randomly removes inputs to a layer with a given probability θ [Srivastava, 2014].



(a) Standard Neural Net (b) After applying dropout.

Drop-Out

- The dropout[2] architecture with stochastic search is

$$
\begin{aligned}
D_i^l &\sim \text{Ber}(\theta), \\
\tilde{Z}^l &= D^l \star Z^l, \ 1 \leq l < L, \\
Z^l &= f^l(W^l \tilde{Z}^{l-1} + b^l).
\end{aligned}
$$

[2] The probability, θ, can be viewed as a further hyper-parameter (like λ) which can be tuned via cross-validation.

Spatial-Temporal Representation

- Gives observations $Y_t = Y(s,t)$ at locations $s := s_1, \ldots, s_N$ and time t
- Cressie and Wikle (2015) provide an introductory overview of spatio-temporal modeling
- In a statistical framework, the non-parametric approach seeks to approximate the unknown map F using a family of spatial basic functions $\Phi(s)$ and random temporal effects $\mathbf{w}(t)$

$$F_t(s) = \sum_{k=1}^{N} w_k(t)\phi_k(s)$$

- Gaussian processes, for spatio-temporal analysis, are computationally intractable and assumes prior knowledge of the covariance function.
- Convolution methods address these issues and are, in fact, a single layer convolution network.

Space-Temporal Representation

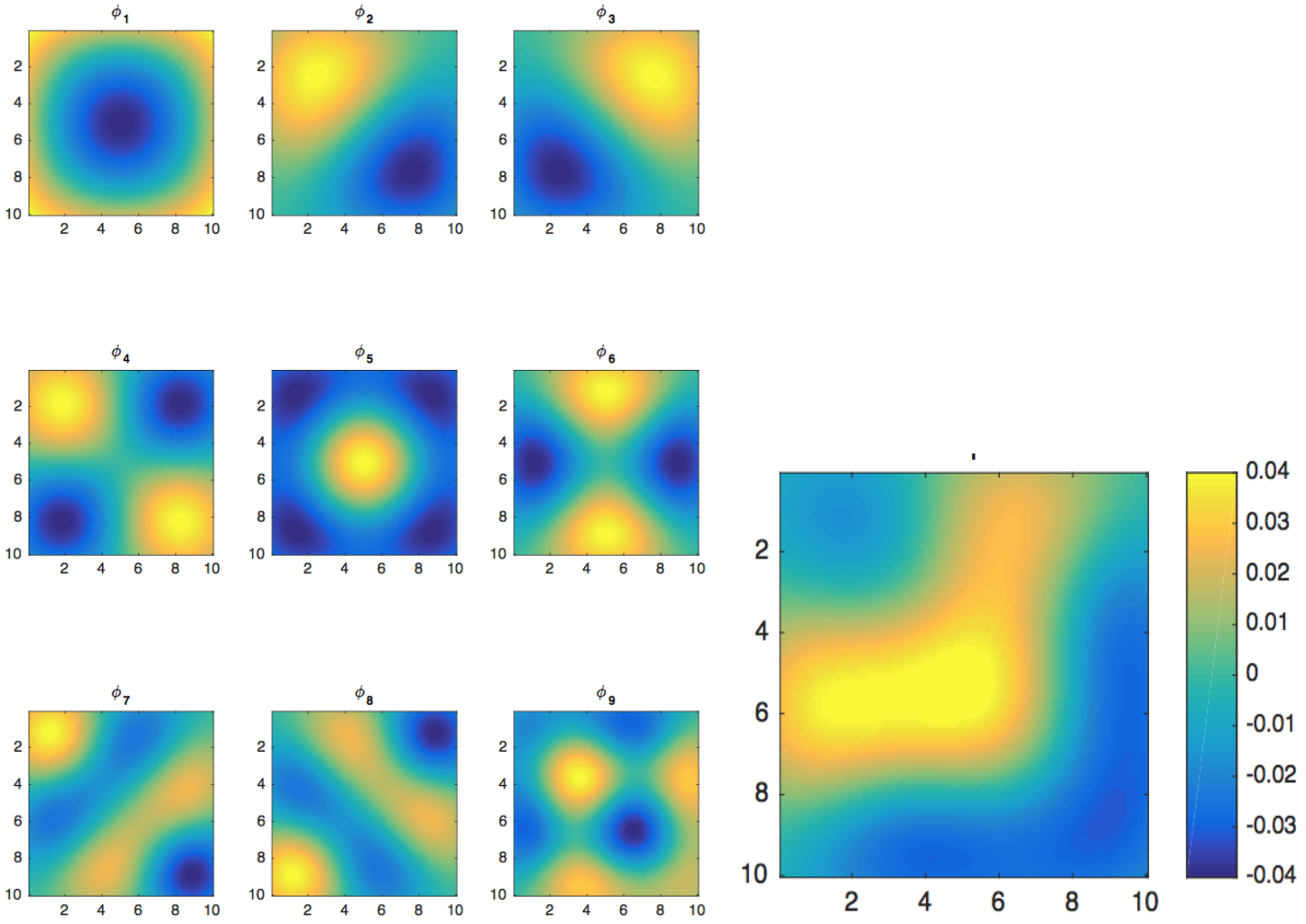


Figure: (left) Spatial basis functions on $\mathbb{R}^2$. (right) $Y = F_t(s)$ at time t_0.

Space-Temporal Prediction with Deep Learning

- The spatio-temporal prediction can be written as

$$Y = x_{t+h}^{t} := \begin{pmatrix} x_{1,t+h} \\ \vdots \\ x_{n,t+h} \end{pmatrix},$$

- x_{t+h}^{t} is the forecast of the random field at time $t+h$, given measurements of x up to time t.
- The predictors are

$$x \equiv x^{t} = \mathrm{vec} \begin{pmatrix} x_{1,t-k} & \cdots & x_{1,t} \\ \vdots & \vdots & \vdots \\ x_{n,t-k} & \cdots & x_{n,t} \end{pmatrix}.$$

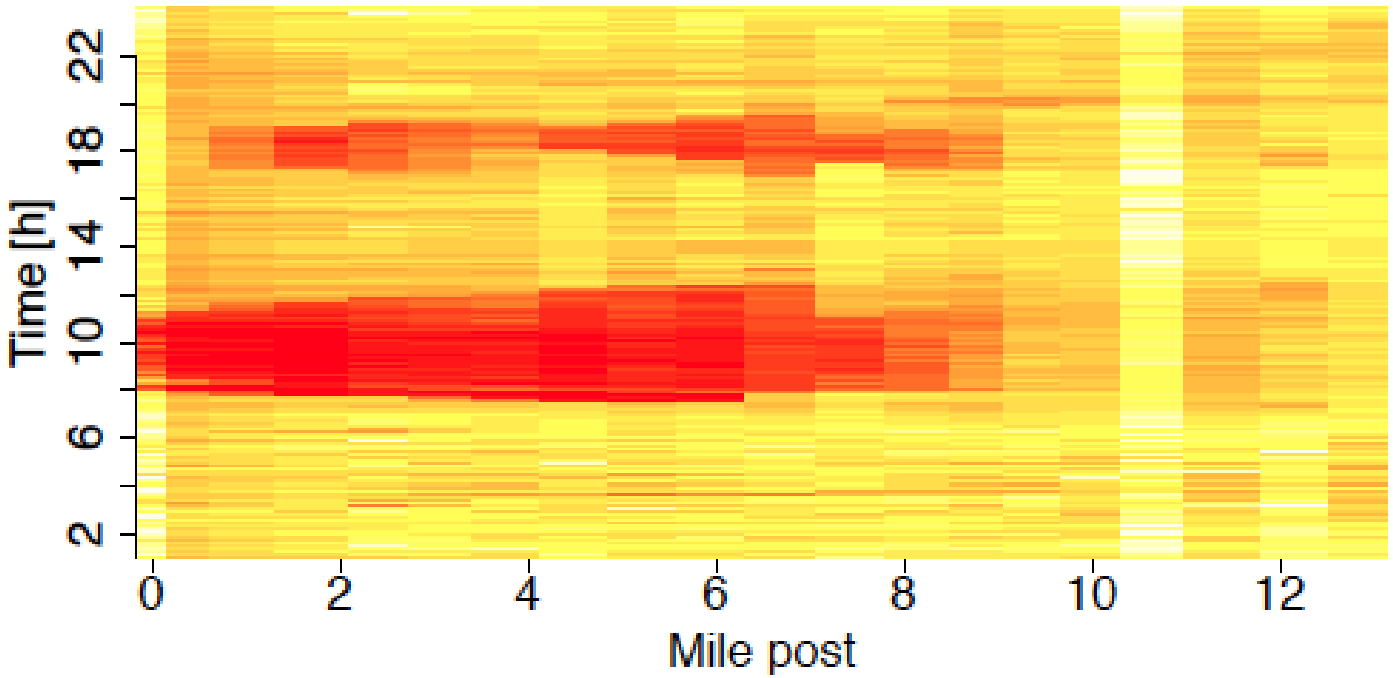
Spatial-Temporal Neural Networks

- Construct layers as a time "filter" given by

$$z_i^{l+1} = f\left(\sum_{i=1}^{N_l}(w_i^{l+1}z_i^l + b_i^{l+1})\right), \mathbf{z}^0 = x^t$$

- f is the activation function and N_l is the number of neurons in layer l.

Space-Time Diagram of Traffic



Traffic Prediction

- Predict the traffic flow speeds at loop detector locations:

$$Y = x_{t+h}^{t} = \begin{pmatrix} x_{1,t+h} \\ \vdots \\ x_{n,t+h} \end{pmatrix}$$

- x_{t+h}^{t} is the forecast of traffic flow speeds at time $t+h$, given measurements up to time t.
- n is the number of locations on the network (loop detectors) and
- $x_{i,t}$ is the cross-section traffic flow speed at location i at time t

Spatial-Temporal Representation in HFT

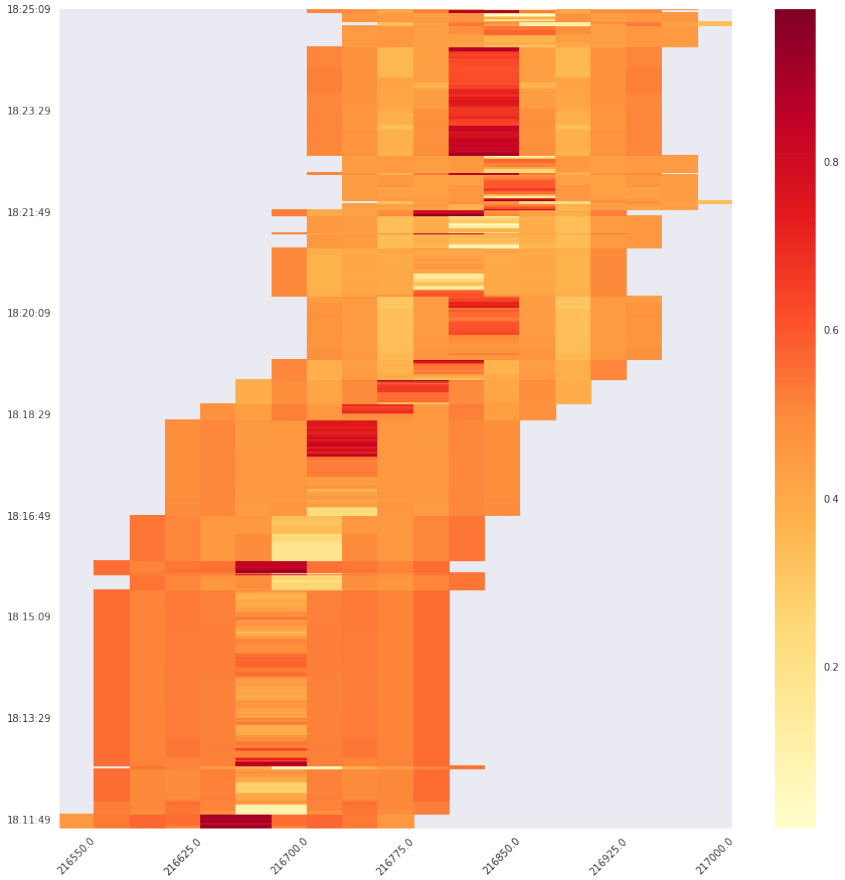


Figure: A space-time diagram showing the double auction The contemporaneous liquidities at each price level, $x_{i,t}$, are represented by the color scale: red denotes a high value of liquidity imbalance and yellow the converse. The auction book are observed to polarize prior to a price movement.

HFT Prediction

- Predict the prices at different levels in a (double) auction:

$$Y = p^t_{t+h} = \begin{pmatrix} p_{1,t+h} \\ \vdots \\ p_{n,t+h} \end{pmatrix}$$

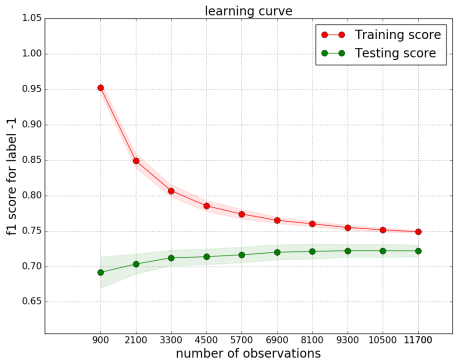
- p^t_{t+h} is the forecast of price changes at time $t + h$, given measurements of up to time t.
- n is the number of price levels and
- $x_{i,t}$ is the cross-section liquidity at level i at time t

Model Configuration[3]

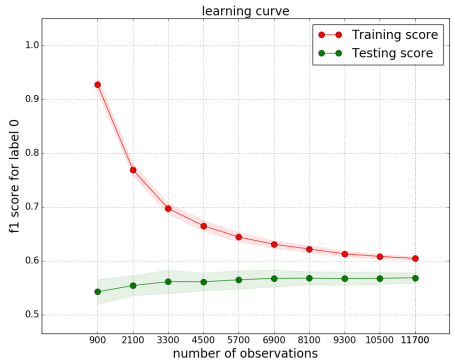
$$\begin{aligned}
\text{Activation function: } & f \in \{\text{ReLU}(x),\ \text{softmax}(x)\} \\
\text{Number of hidden layers: } & L \in \{3, \ldots, 7\} \\
\text{Number of nodes in each layer: } & N_l \in \{50, \ldots, 200\} \\
L_1 \text{ regularization: } & \lambda_1 \in \{10^{-3}, 10^{-2}, 10^{-1}\} \\
L_2 \text{ regularization: } & \lambda_2 \in \{10^{-3}, 10^{-2}, 10^{-1}\} \\
\text{Learning rate: } & \gamma \in \{10^{-4}, 10^{-3}, 10^{-2}\}
\end{aligned}$$

[3] Times series cross-validation is performed using an unbalanced validation and test set, each of size 2×10^5 observations. Each experiment is run for 2500 epochs with a mini-batch size of 32 drawn from the training set of 298,062 observations, containing 411 variables chosen from the elastic-net method.

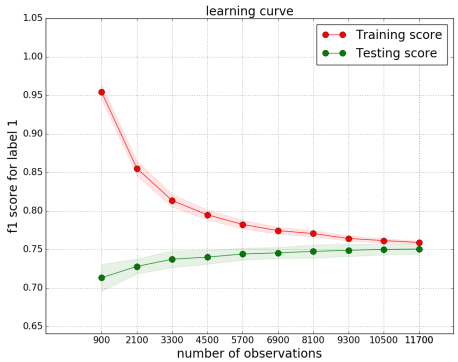
The Bias-Variance Tradeoff



(a) DNN F1-score of $\hat{Y} = 1$



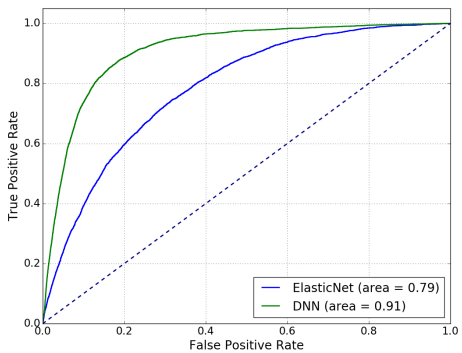
(b) DNN F1-score of $\hat{Y} = 0$



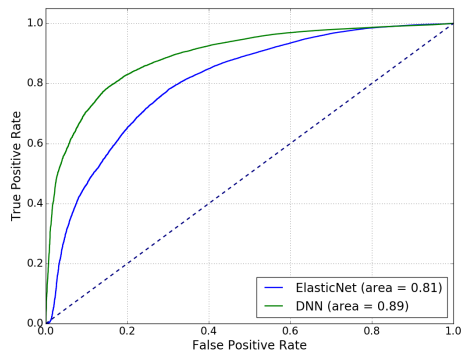
(b) DNN F1-score of $\hat{Y} = -1$.

Table: The learning curves of the deep learner are used to assess the bias-variance tradeoff and are shown for (left) downward, (middle) neutral, or (right) upward price prediction. The variance is observed to reduce with an increased training set size and shows that the deep learning is not-overfitting. The bias on the test set is also observed to reduce with increased training set size.

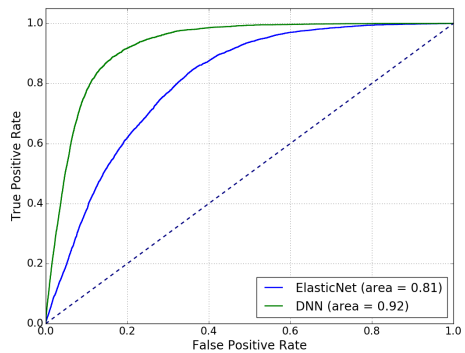
Receiver Operator Characteristics



(a) ROC curves of $\hat{Y} = 1$



(b) ROC curves of $\hat{Y} = 0$



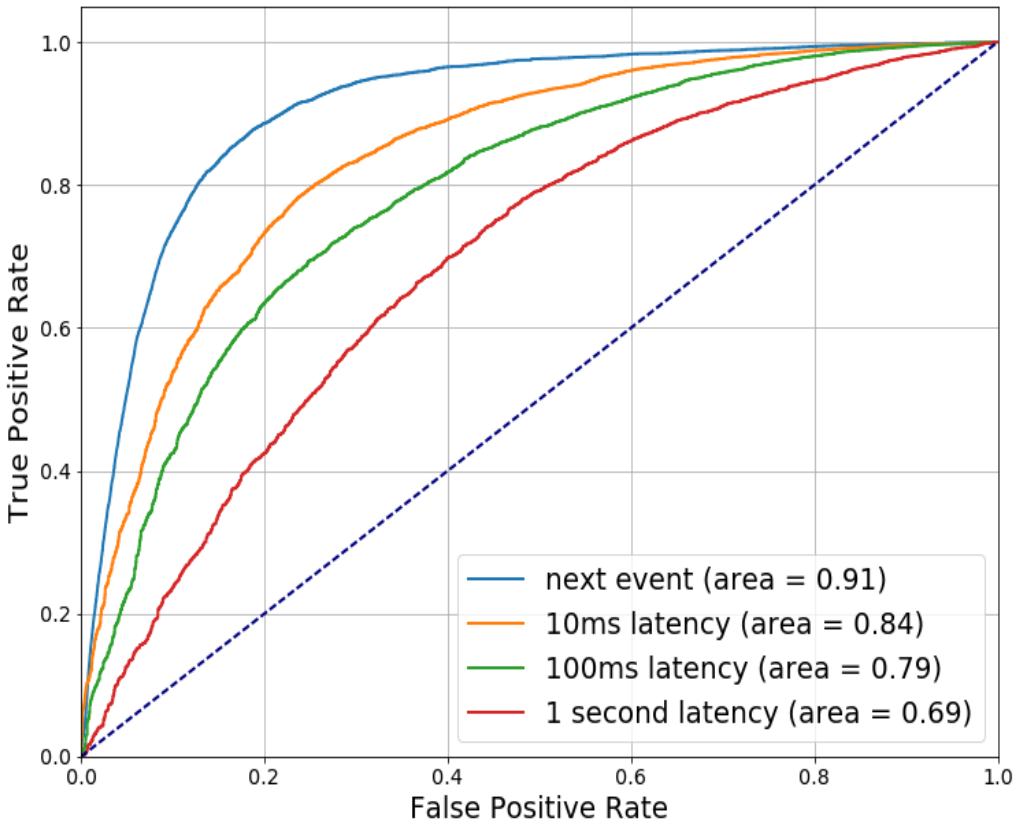
(b) ROC curves of $\hat{Y} = -1$.

Table: The Receiver Operator Characteristic (ROC) curves of the deep learner and the elastic net method are shown for (left) downward, (middle) neutral, or (right) upward next price movement prediction.

Model Sensitivity

Hidden layers	DNN	EL-DNN
1	0.5057967179	0.5691572606
2	0.5340439642	0.5555855057
3	0.5724887077	0.578907192
4	0.5819864454	0.6474221372
5	0.5794411575	0.65784692

Prediction Example



Review of Bayesian Inference

- Given a set of models $\{\mathcal{M}_k\}_{k=1}^K$ to explain the data $\mathcal{D}$, which model is 'best'?
- Estimate the posterior distribution over models:

$$p(\mathcal{M}_k|\mathcal{D}) = \frac{\int_{\boldsymbol{\theta}_k \in \boldsymbol{\Theta}_k} p(\mathcal{D}|\boldsymbol{\theta}_k, \mathcal{M}_k) p(\boldsymbol{\theta}_k|\mathcal{M}_k) d\boldsymbol{\theta}_k p(\mathcal{M}_k)}{\sum_j p(\mathcal{D}|\mathcal{M}_j) p(\mathcal{M}_j)}.$$

- *Model evidence* is a marginal likelihood function over the space of models

$$p(\mathcal{D}|\mathcal{M}_k) = \int_{\boldsymbol{\theta}_k \in \boldsymbol{\Theta}_k} p(\mathcal{D}|\boldsymbol{\theta}_k, \mathcal{M}_k) p(\boldsymbol{\theta}_k|\mathcal{M}_k) d\boldsymbol{\theta}_k$$

- We maintain a belief over which parameters in the model we consider plausible by reasoning with the posterior

$$p(\boldsymbol{\theta}_k|\mathcal{D}, \mathcal{M}_k) = \frac{p(\mathcal{D}|\boldsymbol{\theta}_k, \mathcal{M}_k) p(\boldsymbol{\theta}_k|\mathcal{M}_k)}{p(\mathcal{D}|\mathcal{M}_k)}.$$

- Choose the parameter value which maximizes the posterior distribution (MAP).

Occam's Razor

- The model evidence performs a vital role in the prevention of model overfitting.
- Bayesian inference therefore automates the determination of model complexity using the training data $\mathcal{D}$ alone.

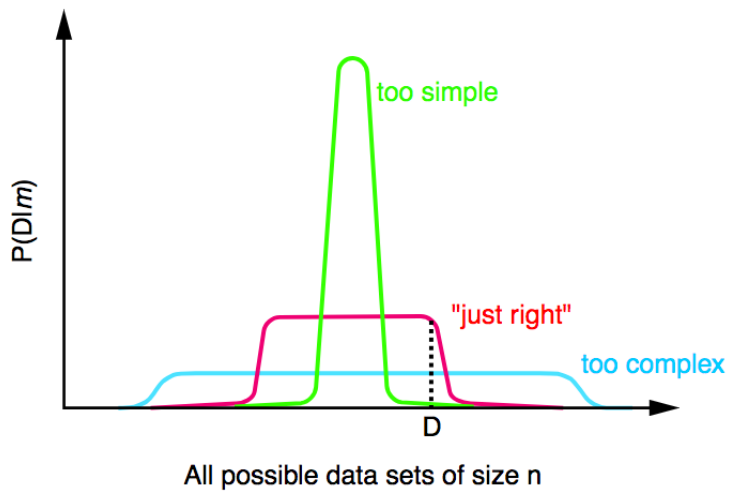


Figure: *The model evidence $p(\mathcal{D}|m)$ performs a vital role in the prevention of model overfitting. Models that are too simple are unlikely to generate the data set. Models that are too complex can generate many possible data sets, but they are unlikely to generate any particular data set at random.*

Model Selection and Averaging

- We can compare any two models via the *posterior odds*. If there are two models $\mathcal{M}_1$ and $\mathcal{M}_2$ then this ratio is given by

$$\frac{p(\mathcal{M}_1|\mathcal{D})}{p(\mathcal{M}_2|\mathcal{D})} = \frac{p(\mathcal{M}_1)p(\mathcal{D}|\mathcal{M}_1)}{p(\mathcal{M}_2)p(\mathcal{D}|\mathcal{M}_2)}$$

- Prior odds multiplied by the ratio of the evidence for each model (*Bayes factor* for $\mathcal{M}_1$).
- Or Bayesian model averaging (BMA) over all models

$$p(\boldsymbol{\theta}_k|\mathcal{D}) = \sum_{k=1}^{K} p(\mathcal{M}_k|\mathcal{D})p(\boldsymbol{\theta}_k|\mathcal{D}, \mathcal{M}_k)$$

Online learning

- An important aspect of Bayesian learning is the capacity to update the posterior in response to the arrival of new data $\mathcal{D}'$.
- The posterior over $\mathcal{D}$ now becomes the prior, and the new posterior is updated to

$$p(\boldsymbol{\theta}_k|\mathcal{D}', \mathcal{D}, \mathcal{M}_k) = \frac{p(\mathcal{D}'|\boldsymbol{\theta}_k, \mathcal{M}_k)p(\boldsymbol{\theta}_k|\mathcal{D}, \mathcal{M}_k)}{\int_{\boldsymbol{\theta}_k \in \boldsymbol{\Theta}_k} p(\mathcal{D}'|\boldsymbol{\theta}_k, \mathcal{M}_k)p(\boldsymbol{\theta}_k|\mathcal{D}, \mathcal{M}_k)d\boldsymbol{\theta}}.$$

- This mechanism for updating our beliefs in response to new data is often referred to as *online learning*.

Bayesian Model Averaging

- Bayesian Model Averaging (BMA) approach for climate model ensemble output [Leamer,1978; Raftery et al., 2005; Berrocal et al., 2007, Bhat el al. 2011]
- $Y \equiv Y(s,t)$ is the projection quantity
- $f_k \equiv f_k(s,t)$ and $h_k \equiv h_k(s,t)$ are model forecasts and hindcasts for the projection quantity
- X is the historical spatial data used as an input to the model

Bayesian Model Averaging: simple case

- Model average distribution of Y conditional on the model forecasts

$$p(Y|f_1, \ldots, f_K) = \sum_{k=1}^{K} w_k g_k(Y|f_k)$$

- The model weights $\sum_{k=1}^{K} w_k = \sum_{k=1}^{K} p(\mathcal{M}_k|X) = 1$
- Assume that $g_k(Y|f_k)$ is Gaussian

$$y|f_k \sim N(a_k + b_k f_k, \sigma_k^2)$$

- Mean of $p(Y|f_1, \ldots, f_K)$:

$$\mathbb{E}[Y|f_1, \ldots, f_K] = \sum_{k=1}^{K} w_k(a_k + b_k f_k)$$

Bayesian Model Averaging: Gaussian Processes

- Bhat et al. (2011)[4] use a linear model to regress Y for each hindcast h_k over all (s, t)

$$Y = a_k + b_k h_k + \delta_k + \epsilon_k$$

- ϵ_k represents non-spatial error and i.i.d. Gaussian for model $\mathcal{M}_k$.
- δ_k represents space-time dependence for model $\mathcal{M}_k$.
- Model δ_k as a zero mean linear Gaussian process $\delta_k | \phi_s^k, \phi_t^k, \kappa^k \sim N(0, \mathcal{K}_k \mathcal{K}_k^T)$

$$\mathcal{K}_k(i,j) = \sqrt{\kappa^k} exp\{-\frac{||s_i - u_j||^2}{(\phi_s^k)^2}\} \cdot exp\{-\frac{||t_i - v_j||^2}{(\phi_t^k)^2}\}$$

[4] Bhat, K. S., Haran, M., Terando, A. and Keller, K., Climate Projections Using Bayesian Model Averaging and Space–Time Dependence, Journal of Agricultural, Biological, and Environmental Statistics, Dec 2010.

Bhat et al. Results

- Ensemble of 20 AR4 GCM hindcasts and forecasts
- GISS historical data gridded to 2 Deg by 2 Deg cells referenced annually between 1900 and 2000
- 1,225,332 space-time locations

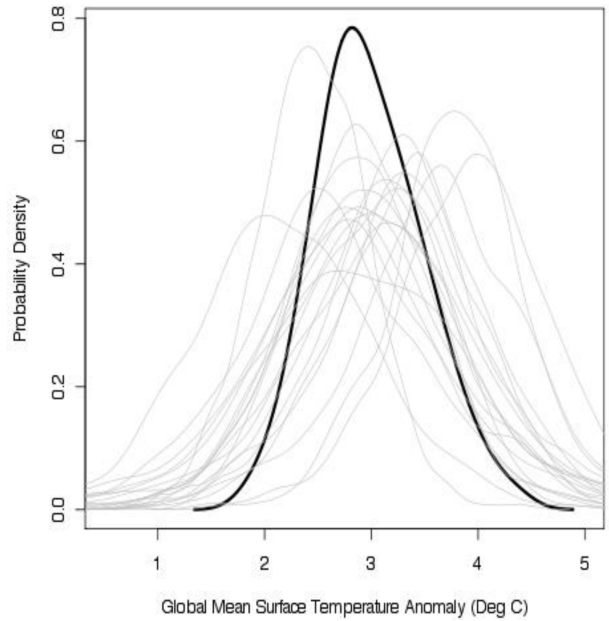


Figure: BMA predictive pdf for global mean surface temperature anomaly in 2100 (solid black line) and the twenty components for each GCM forecast (gray lines) under the A1B scenario. Scenario A1B assumes strong economic growth, a globalized economy with converging income levels between nations, a global population of 9 million in 2050 but stable or decreasing afterwards, and reliance on both fossil-fuels and non-fossil energy sources.

Deep Learning Approach: A potential direction

- For each dynamic model, find the map over all (s, t) pairs.

$$Y = F_{\boldsymbol{\theta}}(h_k)$$

- We arrive at a deep learner $\hat{Y}^k_{\hat{\boldsymbol{\theta}}}(x)$ for mapping all dynamic model outputs to historical observations Y
- Distribution of $g_k(Y|f_k, \hat{\boldsymbol{\theta}})$ given by DL as a MAP estimator
- Able to express high degree of non-linearity between dynamic model outputs and the response
- Spatial-temporal structure captured without assuming a data generation process for the error and spatial covariance structure

Summary

- Predicting spatio-temporal flows is a challenging problem as dynamic spatio-temporal data possess underlying complex interactions and nonlinearities
- Deep learning applies layers of hierarchical hidden variables to capture these interactions and nonlinearities without using a data generating process.
- Deep learning could be embedded in a Bayesian model averaged forecast:
  - Apply non-linear weighted GCM models
  - Weights are represented by posterior probabilities and can be updated through online learning
  - No reliance on a data generation process and model spatial-structure