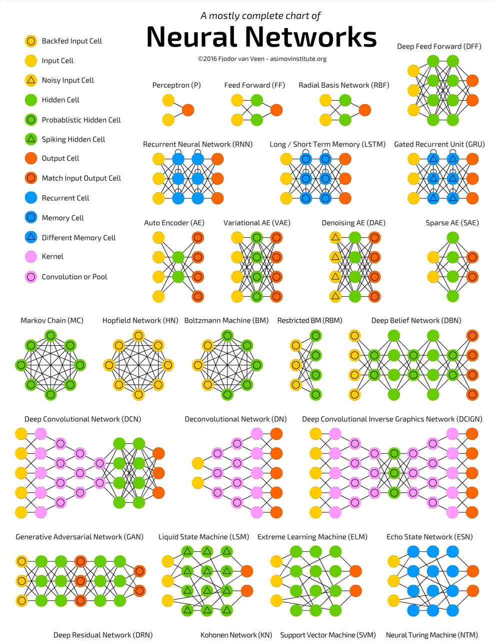
Deep Learning: Theory and Practice in Finance

Matthew Dixon[1]

Illinois Institute of Technology

November 20th, 2017
Department of Finance and Risk Engineering
NYU Tandon

[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
M.F. Dixon, Sequence Classification of the Limit Order Book using Recurrent Neural Networks, to appear in J. Computational Science, Special Issue on Topics in Computational and Algorithmic Finance, arXiv:1707.05642
M.F. Dixon, High Frequency Market Making with Machine Learning, arXiv:1707.05642


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Deep Architectures in TensorFlow

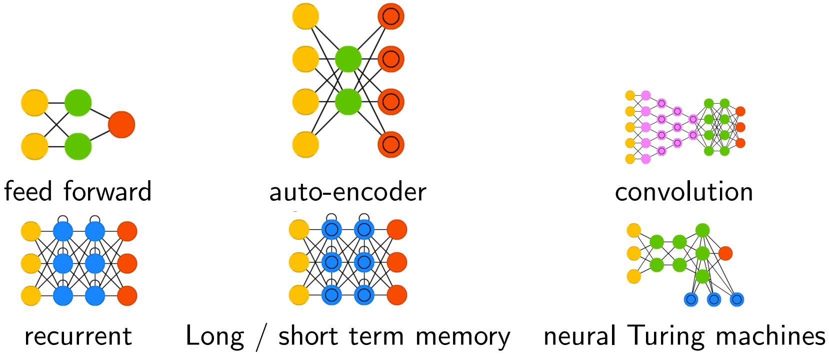
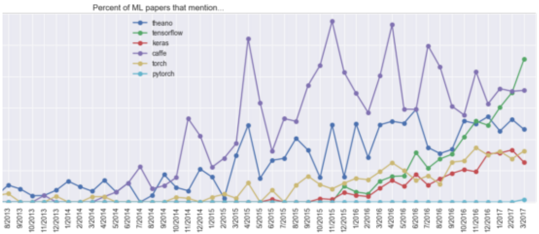
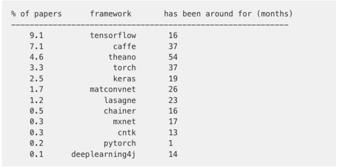


Figure: Most commonly used deep learning architectures for modeling. Source: `http://www.asimovinstitute.org/neural-network-zoo`

Deep Learning Packages[2]





```
% 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

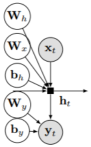
Tensorflow build for Intel Xeon Phi: https://github.com/tensorflow/tensorflow.git

[2] Python examples: https://github.com/Quiota/tensorflow

Bayesian Deep Learning

Edward is a Python library for Bayesian Deep Learning which uses TensorFlow

Edward



```python
def rnn_cell(hprev, xt):
  return tf.tanh(tf.dot(hprev, Wh) + tf.dot(xt, Wx) + bh)

Wh = Normal(mu=tf.zeros([H, H]), sigma=tf.ones([H, H]))
Wx = Normal(mu=tf.zeros([D, H]), sigma=tf.ones([D, H]))
Wy = Normal(mu=tf.zeros([H, 1]), sigma=tf.ones([H, 1]))
bh = Normal(mu=tf.zeros(H), sigma=tf.ones(H))
by = Normal(mu=tf.zeros(1), sigma=tf.ones(1))

x = tf.placeholder(tf.float32, [None, D])
h = tf.scan(rnn_cell, x, initializer=tf.zeros(H))
y = Normal(mu=tf.matmul(h, Wy) + by, sigma=1.0)
```

Foundations of 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*.

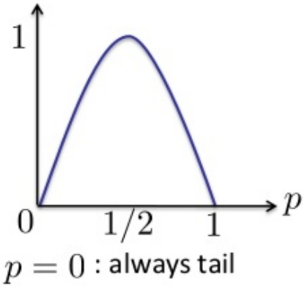
Entropy as a measure of uncertainty

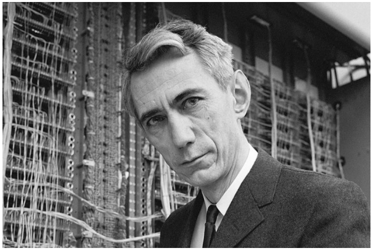
Coin-flip example (Klein & Manning)

- Consider a non-fair coin toss.
- There are two outcomes, $\Omega = \{H, T\}$.
- Y is the number of heads with density $f(Y = 1) = P\{H\} = p$ and $f(Y = 0) = P\{T\} = 1 - p$
- The binary entropy of Y under f is

$$\mathcal{H}(f) = -p \log_2 p - (1 - p) log_2(1 - p) \leq 1\text{bit}$$

Entropy as a measure of uncertainty





C. Shannon, A Mathematical Theory of Communication, The Bell System Technical Journal, Vol. 27, pp. 379-423, 623-656, July, October, 1948.

Entropy and parameterized mass

- $Y : \Omega \to \mathcal{Y} \subset \mathbb{R}$ is an i.i.d. discrete random variable
- $g(y|\theta) = P\{\omega \in \Omega; Y(\omega) = y\}$ is a parameterized probability mass function for Y
- The entropy is

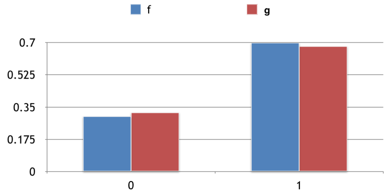
$$\mathcal{H}(g) := -\mathbb{E}_g[log_2 g] = -\sum_{y \in \mathcal{Y}} g(y|\theta) log_2 g(y|\theta)$$

Entropy: example

Coin-flip example (Klein & Manning)

Suppose that there is a model $g(y|\theta)$ of the non-fair coin with $g(Y=1|\theta)=p_\theta$, $g(Y=0|\theta)=1-p_\theta$. The cross-entropy is

$$\mathcal{H}(g) = -p_\theta \log_2 p_\theta - (1-p_\theta) log_2(1-p_\theta).$$



Cross-Entropy

- We can measure how different $g(y|\theta)$ is from the true density $f(y)$ using the cross-entropy

$$\mathcal{H}(f,g) := -\mathbb{E}_f\left[\log_2 g\right] = \sum_{y\in\mathcal{Y}} f(y)\log_2 g(y|\theta) \geq H(f)$$

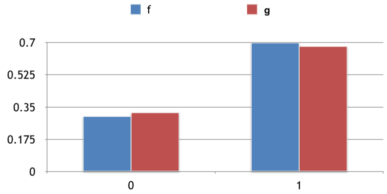
  so that $\mathcal{H}(f,f) = H(f)$.
- *Goal*: minimize the cross-entropy with respect to θ.

Cross-Entropy: example

Coin-flip example (Klein & Manning)

Suppose that there is a model $g(y|\theta)$ of the non-fair coin with $g(Y=1|\theta)=p_\theta$, $g(Y=0|\theta)=1-p_\theta$. The cross-entropy is

$$\mathcal{H}(f,g) = -p\log_2 p_\theta - (1-p) log_2(1-p_\theta) \geq -p\log_2 p - (1-p) log_2(1-p)$$



Parameter Estimation

- *Goal*: Using observations $\mathbf{y} = \{y_1, \ldots, y_n\}$, learn the best model $\hat{g} := g(\mathbf{y}|\hat{\theta})$ that minimizes the cross-entropy.
- In this case

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, \mathcal{H}(f, g).$$

- We *could* use the log likelihood function $\ell(\theta|\mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} lng(y_i|\theta)$ as an estimator for the negative cross-entropy[3]

$$-cH(f, g) \approx \ell(\theta|\mathbf{y}) + \text{penalty-term}$$

[3] c is a constant for the conversion from base 2 to base e.

Maximum Likelihood Estimation

- Finding the maximum likelihood estimator $\hat{\theta}$ 'should' minimize the cross entropy
- If $\mathbf{y}$ is the realization of Bernoulli trials, e.g. $1, 0, 1, 1, 1, 1, 0, 0$ then

$$
\begin{aligned}
\ell(\hat{\theta}|\mathbf{y}) &= ln\left(p_{\hat{\theta}}^{y_1}(1-p_{\hat{\theta}})^{y_1} \times \ldots \times p_{\hat{\theta}}^{y_n}(1-p_{\hat{\theta}})^{y_n}\right) \\
&= \frac{1}{n}\sum_{i=1}^{n} y_i lnp_{\hat{\theta}} + (1-y_i)ln(1-p_{\hat{\theta}})
\end{aligned}
$$

- In practice, this just leads to model over-fitting. AIC was introduced in econometrics to fix this.

Machine Learning: A tale of two coins

- Let X and Y denote the number of heads thrown from each of two correlated coins.
- Suppose that the conditional density function is $g(y|x, \theta)$ with $g(1|x) = p_\theta(x)$ and $g(0|x) = 1 - p_\theta(x)$.
- The cross-entropy between the true density $f(y|x)$ and $g(y|x, \theta)$ is

$$\mathcal{H}\left(f(\cdot|x), g(\cdot|x,\theta)\right) = -p(x)\log_2 p_\theta(x) - (1-p(x)) log_2(1-p_\theta(x))$$

- Given data $\mathcal{D} = \{x_1, y_i\}_{i=1}^n$, the likelihood function $\ell(\theta|\mathcal{D}) = \frac{1}{n}\sum_{i=1}^n lng(y_i|x_i, \theta)$ is maximized for a choice of $\theta = \hat{\theta}$

$$\ell(\hat{\theta}|\mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n} y_i lnp_{\hat{\theta}}(x_i) + (1 - y_i) ln(1 - p_{\hat{\theta}}(x_i))$$

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*[4] $X = (X_1, \dots, X_P)$ of P variables.
- Machine learning can be simply viewed as the study and construction of an input-output map of the form

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

- The output variable, Y, can be continuous, discrete or mixed.
- For example, in a classification problem, $F : X \to Y$ where $Y \in \{1, \dots, K\}$ 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

$$Y = AX + b.$$

[4]With abuse of notation, X is hereon used to denote an observation matrix of a random vector.

Mathematical View of Deep Learning

- The theoretical roots of DL are in the Kolmogorov-Arnold representation theorem [Kolmogorov, 1963][5] of multivariate functions.
- Pascanu et al. (2013)[6] and Montfar and Morton (2015)[7] provide results on the advantage of representing some functions compactly with deep layers.
- Poggio (2016) [8] extends theoretical results on when deep learning can be exponentially better than shallow learning.

[5]Andrei Nikolaevich Kolmogorov. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. American Mathematical Society Translation, 28(2):55-59, 1963

[6]Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to Construct Deep Recurrent Neural Networks. arXiv:1312.6026

[7]When Does a Mixture of Products Contain a Product of Mixtures? SIAM Journal on Discrete Mathematics, 29(1):321-347, 2015.

[8]Deep Learning: Mathematics and Neuroscience. A Sponsored Supplement to Science, Brain-Inspired intelligent robotics: The intersection of robotics and neuroscience: 9-12, 2016

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

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

- $f^{l}_{W^{l},b^{l}}(X) := f^{l}(W^{l}X + b^{l})$ *is a semi-affine function, where* f^{l} *is univariate and continuous.*
- $W = (W^{1}, \ldots, W^{L})$ *and* $b = (b^{1}, \ldots, b^{L})$ *are weight matrices and offsets respectively.*
- *Many statistical techniques are 'shallow learners', e.g. PCA* $Y = f(X) = WX + b$*, columns of* W *form an orthogonal basis.*

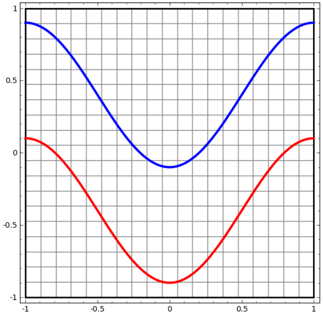
Deep Predictors

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

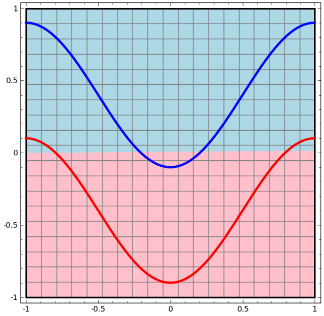
$$\begin{aligned}
\hat{Y}(X) &= f^L(Z^{L-1}),\\
Z^0 &= X,\\
Z^1 &= f^1\left(W^1 Z^0 + b^1\right),\\
Z^2 &= f^2\left(W^2 Z^1 + b^2\right),\\
&\dots\\
Z^{L-1} &= f^{L-1}\left(W^{L-1} Z^{L-2} + b^{L-1}\right).
\end{aligned}$$

- 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

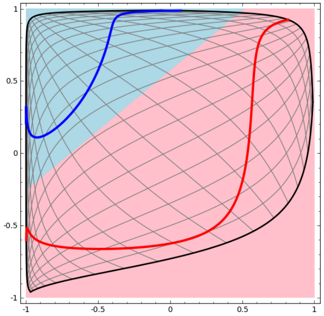


Solution using a linear method

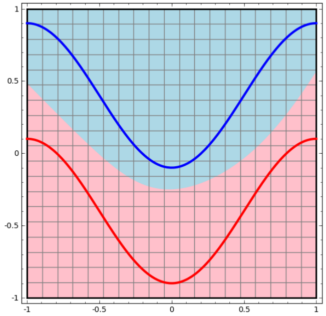
Figure: Image source: Chris Olah, Google Brain.

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.

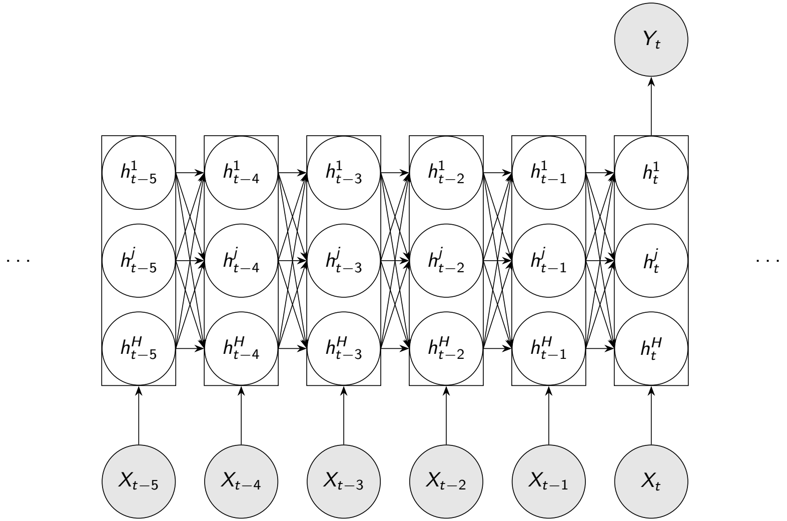
Recurrent Neural Network Predictors

- Input-output pairs $\mathcal{D} = \{X_t, Y_t\}_{t=1}^N$ are auto-correlated observations of X and Y at times $t = 1, \dots, N$
- Construct a nonlinear times series predictor, $\hat{Y}(\mathcal{X})$, of an output, Y, using a high dimensional input matrix of T length sub-sequences $\mathcal{X}$:

$$y = F(\mathcal{X}) \text{ where } \mathcal{X}_t = seq_T(X_t) = (X_{t-T+1}, \dots, X_t)$$

- X_{t-j} is a j^{th} lagged observation of X_t, $X_{t-j} = L^j[X_j]$, for $j = 0, \dots, T-1$.

Recurrent Neural Networks


Y_t
h^1_{t-5} h^1_{t-4} h^1_{t-3} h^1_{t-2} h^1_{t-1} h^1_t
h^j_{t-5} h^j_{t-4} h^j_{t-3} h^j_{t-2} h^j_{t-1} h^j_t
h^H_{t-5} h^H_{t-4} h^H_{t-3} h^H_{t-2} h^H_{t-1} h^H_t
$\cdots$ $\cdots$
X_{t-5} X_{t-4} X_{t-3} X_{t-2} X_{t-1} X_t

Recurrent Neural Networks

- For each time step $t = 1, \ldots, T$, a function F_h generates a hidden state h_t:

$$h_t = F_h(X_t, h_{t-1}) := \sigma(W_h X_t + U_h h_{t-1} + b_h),\ W_h \in \mathbb{R}^{H \times P}, U_h \in \mathbb{R}^{H \times H}$$

- When the output is continuous, the model output from the final hidden state is given by:

$$Y = F_y(h_T) = W_y h_T + b_y,\ W_y \in \mathbb{R}^{M \times H}$$

- When the output is categorical, the output is given by

$$Y = F_y(h_T) = \text{softmax}(F_y(h_T))$$

- *Goal*: find the weight matrices $W = (W_h, U_h, W_y)$ and biases $b = (b_h, b_y)$.

Training

- With a training set $D = \{Y_i, X_i\}_{i=1}^{n}$, solve the constrained optimization

$$(\hat{W}, \hat{b}) = \underset{W,b}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(Y_i, \hat{Y}^{W,b}(X_i))$$

- When Y is categorical, $\mathcal{L}(Y, \hat{Y})$ gives an approximation to the cross-entropy

$$\mathcal{L}(Y_i, \hat{Y}^{W,b}(X_i)) = -Y_i log \hat{Y}^{W,b}(X_i) + \phi(W, b)$$

- For regression, the L_2-norm for a traditional least squares problem is chosen as an error measure

$$\mathcal{L}(Y_i, \hat{Y}^{W,b}(X_i)) = \| Y_i - \hat{Y}^{W,b}(X_i) \|_2^2 + \phi(W, b)$$

- $\nabla \mathcal{L}$ is given in closed form by a chain rule and, through back-propagation, each layer's weights $\hat{W}^l$ are fitted with stochastic gradient descent.

Time Series Cross Validation

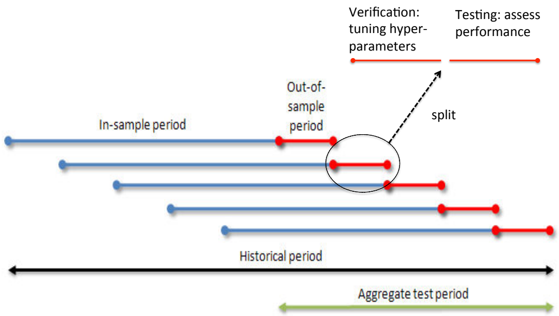
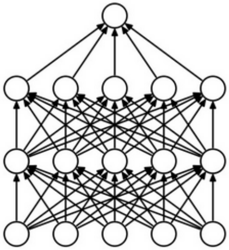
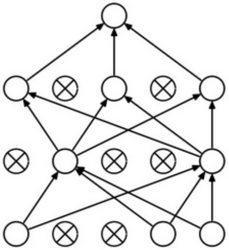


Figure: Times series cross validation, also referred to as walk forward optimization, is used to avoid introducing look-ahead bias into the model.

Drop-Out

- Dropout[9] 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.

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

Drop-Out

The dropout architecture with stochastic search is

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

Limit Order Book Updates

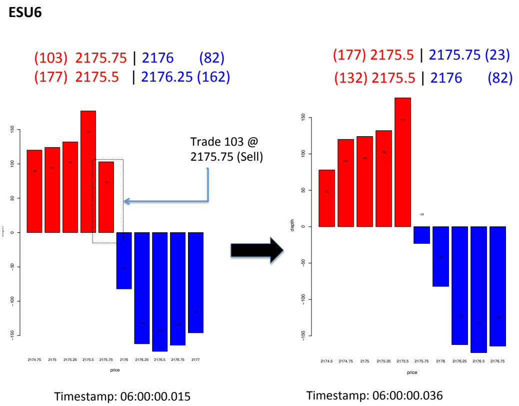


Figure: An exemplary sequence of limit order book updates in the ES futures market (ESU6) is shown before and after the arrival of a sell market order. The sell order is observed to match the supply of liquidity on offer at the best bid price and the entire book moves down by a tick. The sequence has been restricted to the top five levels of the order book out of ten levels provided in the CME exchange feed.

Historical Data

- At any point in time, the amount of liquidity in the market can be characterized by the cross-section of book depths.
- We build a mid-price forecasting model based on the cross-section of book depths.

Timestamp	$p^b_{1,t}$	$p^b_{2,t}$	...	$d^b_{1,t}$	$d^b_{2,t}$	...	$p^a_{1,t}$	$p^a_{2,t}$	...	$d^a_{1,t}$	$d^a_{2,t}$	...	Response
06:00:00.015	2175.75	2175.5	...	103	177	...	2176	2176.25	...	82	162	...	-1
06:00:00.036	2175.5	2175.25	...	177	132	...	2175.75	2176	...	23	82	...	0

Table: The spatio-temporal representation of the limit order book before and after the arrival of the sell market order. The response represents the direction of the mid-price movement over the subsequent interval. $p^b_{i,t}$ and $d^b_{i,t}$ denote the level i quoted bid price and depth of the limit order book at time t. $p^a_{i,t}$ and $d^a_{i,t}$ denote the corresponding level i quoted ask price and depth.

Price Impact Model

- The response is

$$Y = \Delta p_{t+h}^{t} \tag{1}$$

- Δp_{t+h}^{t} is the forecast of discrete mid-price changes from time t to $t+h$, given measurement of the predictors up to time t.
- The predictors are embedded

$$x = x^t = \mathrm{vec}\begin{pmatrix} x_{1,t-k} & \cdots & x_{1,t} \\ \vdots & & \vdots \\ x_{n,t-k} & \cdots & x_{n,t} \end{pmatrix} \tag{2}$$

- n is the number of quoted price levels, k is the number of lagged observations, and $x_{i,t} \in [0,1]$ is the relative depth, representing liquidity imbalance, at quote level i:

$$x_{i,t} = \frac{d_{i,t}^{a}}{d_{i,t}^{a} + d_{i,t}^{b}}. \tag{3}$$

Spatial-Temporal Representation

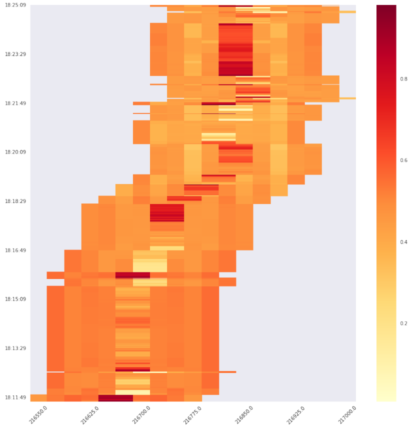


Figure: A space-time diagram showing the limit order book. The contemporaneous depths imbalances at each price level, $x_{i,t}$, are represented by the color scale: red denotes a high value of the depth imbalance and yellow the converse. The limit order book are observed to polarize prior to a price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

The Price Impact of Order Flow

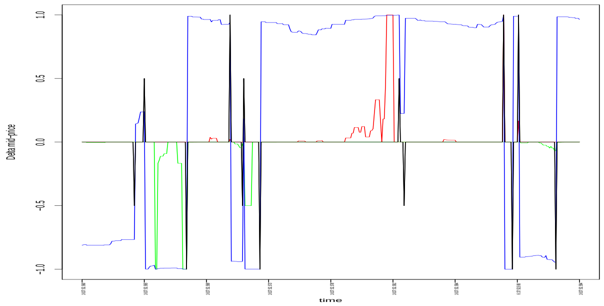


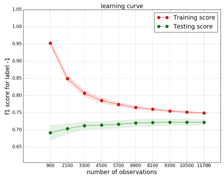
Figure: The black line represents the observed change in mid-price over a 34 milli-second period from 16:37:52.560 to 16:37:52.594. The liquidity imbalance (blue), scaled here to the $[-1, 1]$ interval, although useful in predicting the direction of the next occurring price change, is generally a poor choice for predicting when the price change will occur. The order flow is a better predictor of next-event price movement, although is difficult to interpret when either of the buy (red) and sell order flows (green) are small.

Model Configuration[10]

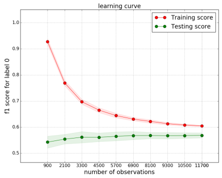
$$\begin{aligned}
\text{Activation function: } & f \in \{\text{ReLU}(x),\ \text{softmax}(x)\} \\
\text{Number of hidden layers: } & L \in \{3, \ldots, 7\} \\
\text{Number of nodes in each layer: } & N_l \in \{50, \ldots, 200\} \\
L_1 \text{ regularization: } & \lambda_1 \in \{10^{-3}, 10^{-2}, 10^{-1}\}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

[10] 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.

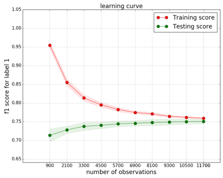
The Bias-Variance Tradeoff



(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.

The Bias-Variance Tradeoff

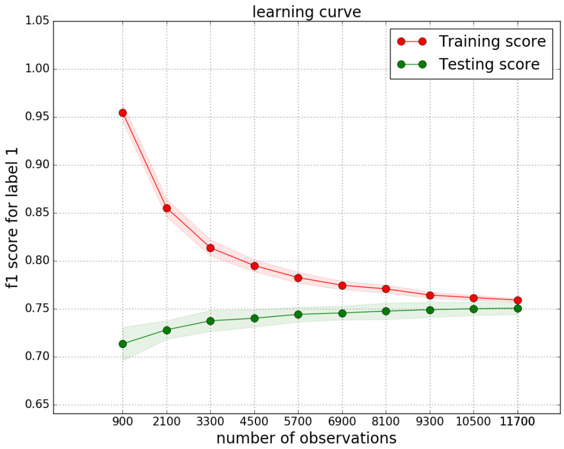
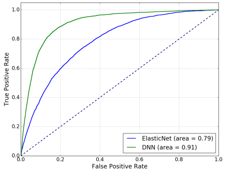
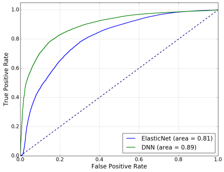


Table: The learning curve of the deep learner used to assess the bias-variance tradeoff for an upward price prediction. The variance is observed to reduce with an increased training set size and shows that the deep learning is not-overfitting. The bias on the test set is also observed to reduce with increased training set size.

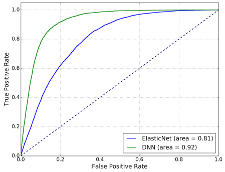
Receiver Operator Characteristics



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



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



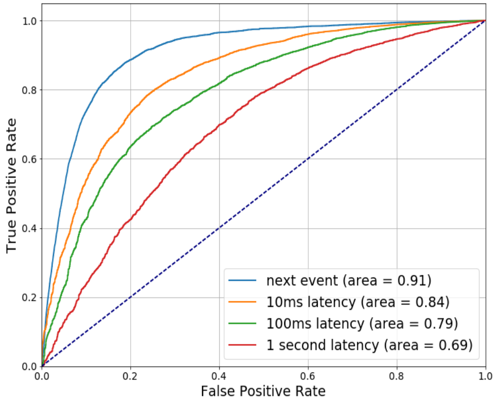
(b) ROC curves of $\hat{Y} = -1$.

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



Prediction Example

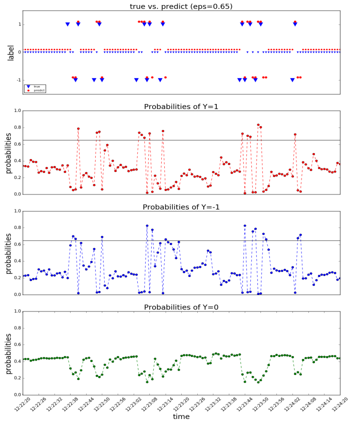


Figure: The (top) comparison of the observed ESU6 mid-price movements with the deep learner forecasted price movements over one milli-second intervals between 12:22:20 and 12:24:20 CST. The bottom three panels show the corresponding probabilities of predicting each class. A probability threshold of 0.65 for the up-tick or down-tick classification is chosen here for illustrative purposes.

Predictive Performance Comparisons

Features	Method	$\hat{Y}=-1$			$\hat{Y}=0$			$\hat{Y}=1$		
		precision	recall	f1	precision	recall	f1	precision	recall	f1
Liquidity Imbalance	Logistic	0.010	0.603	0.019	0.995	0.620	0.764	0.013	0.588	0.025
	Kalman Filter	0.051	0.540	0.093	0.998	0.682	0.810	0.055	0.557	0.100
	RNN	0.037	0.636	0.070	0.996	0.673	0.803	0.040	0.613	0.075
Order Flow	Logistic	0.042	0.711	0.079	0.991	0.590	0.740	0.047	0.688	0.088
	Kalman Filter	0.068	0.594	0.122	0.996	0.615	0.751	0.071	0.661	0.128
	RNN	0.064	0.739	0.118	0.995	0.701	0.823	0.066	0.728	0.121
Spatio-temporal	Elastic Net	0.063	0.754	0.116	0.986	0.483	0.649	0.058	0.815	0.108
	RNN	0.084	0.788	0.153	0.999	0.729	0.843	0.075	0.818	0.137
	FFWD NN	0.066	0.758	0.121	0.999	0.657	0.795	0.065	0.796	0.120
	White Noise	0.004	0.333	0.007	0.993	0.333	0.499	0.003	0.333	0.007

ESU6 Limit Order Book Activity

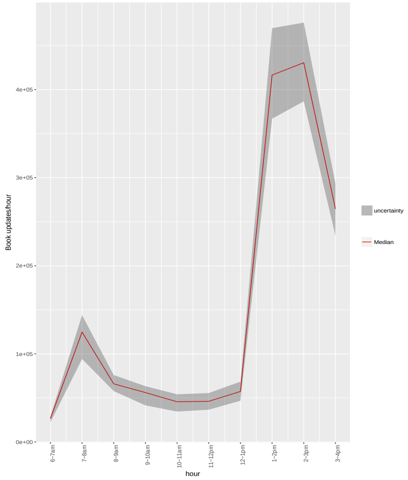
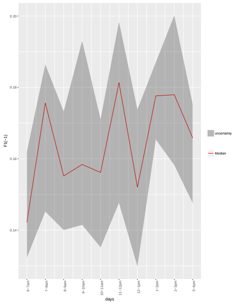
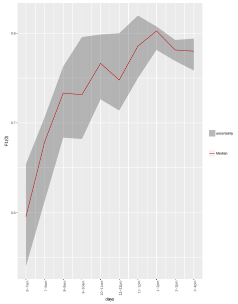


Figure: *The hourly limit order book rates of ESU6 are shown by time of day. A surge of quote adjustment and trading activity is consistently observed between the hours of 7-8am CST and 3-4pm CST.*

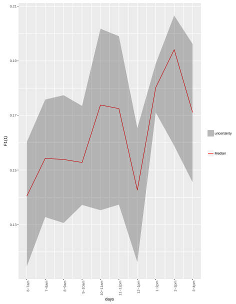
Intra-day Predictive Performance



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



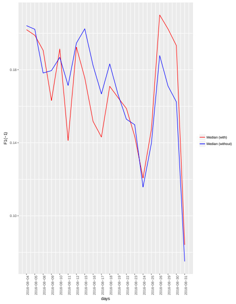
(b) F1 score of $\hat{Y} = 0$



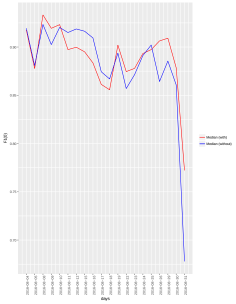
(c) F1 score of $\hat{Y} = 1$.

Figure: *The intra-day F1 scores are shown for (left) downward, (middle) neutral, or (right) upward next price movement prediction.*

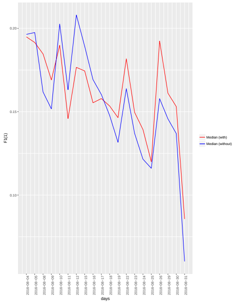
Daily Retraining



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



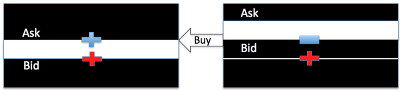
(b) F1 score of $\hat{Y} = 0$

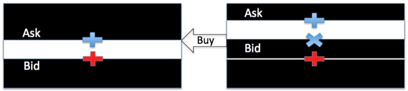


(c) F1 score of $\hat{Y} = 1$.

Figure: *The F1 scores over the calendar month, with (red) and without (blue) daily retraining of the RNN, are shown for (left) downward, (middle) neutral, or (right) upward next price movement prediction.*

Adverse Selection



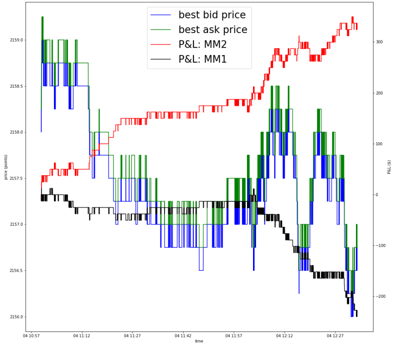


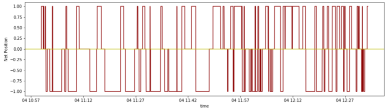
Adverse price selection

Avoiding adverse price selection.

Figure: Limit orders placed by the market marker are denoted with the '+' symbol- red denotes a buy limit order and blue denotes a sell limit order. The minus symbol denotes that the order has been filled.

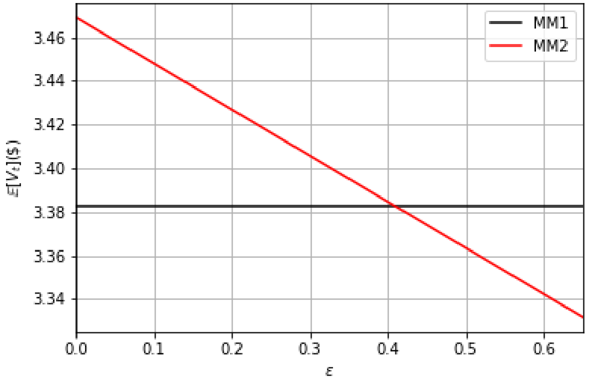
Using Predictions for Market Making[11]





[11] M.F. Dixon, High Frequency Market Making with Machine Learning, arXiv:1707.05642

Using Predictions for Market Making[12]



$$\mathbb{E}[V_t] = tr(C(t_0)T'(t))$$

[12] M.F. Dixon, High Frequency Market Making with Machine Learning, arXiv:1707.05642

Example: Portfolio returns

- Consider a portfolio of positions in n stocks
- The portfolio is assumed to be equally weighted so that the returns are

$$r_P(t) = \sum_{i=1}^{n} w_i r_i(t),\ \ w_i = \frac{1}{n}$$

- *Goal:* Learn the relationship between all the previous stock returns and directional change in the portfolio returns
- Key idea is to maximize the capacity to predict the next day portfolio returns given cross-sectional market data

Example: Portfolio returns

- The observed data consists of historical returns $\{X_t\}_{t=1}^{T}$ where $X_t = r_1(t), \ldots, r_n(t)$
- The data is labeled with a categorical variable

$$Y_t = \begin{cases} 1, & r_P(t+1) > \epsilon \\ 0, & |r_P(t+1)| \leq \epsilon \\ -1, & r_P(t+1) < \epsilon \end{cases}$$

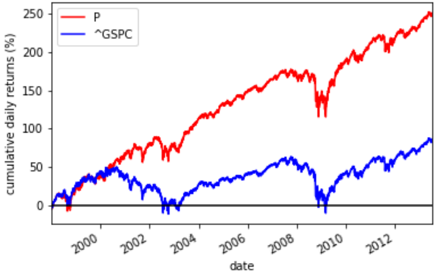
- Goal is to learn the map $\hat{Y}_t = F_{W,b}(X_t)$
- ϵ is a threshold chosen from the training data to balance the observations.

Example: Portfolio strategy

- A simple strategy $\mathcal{S}(Y_t)$ chooses whether to hold a long, short or neutral position in all stocks over the next period.

$$\mathbf{w}_{t+1} = \mathcal{S}(Y_t) = \begin{cases} \frac{1}{n}, Y_t = 1 \\ 0, Y_t = 0 \\ -\frac{1}{n}, Y_t = -1 \end{cases}$$

Example: Portfolio returns



	r_P	r_{GSPC}
μ	0.064	1.330
σ	0.016	1.317

The selected portfolio consists of a subset of the S&P 500 and outperforms it over a 15 year period.

Data preparation

- Use all symbols of the S&P 500 on 2013-7-3.
- Extract historical daily adjusted close prices from Yahoo finance from 1998-1-2 to 2013-7-3
- Remove symbols which have missing prices rather than truncate the time series.
- Use a training horizon of 3840 days, a test horizon of 30 days, and retrain every 30 days over 300 test observations.
- Avoid look ahead bias by only normalizing input with moment estimation from the training data.

Modeling

- Logistic regression, with regularization (a.k.a. Tikhonov or ridge regression) is used as the baseline classifier
- Neural network contains 3 hidden layers with a 'tapered' feed-forward architecture $(200, 100, 50$ neurons in each layer).
- There are several other 'hyper-parameters' which can be tuned to manage the bias-variance trade-off.
- This example does not consider how to capture auto-correlations in the data. The model is cross-sectional only.

Example: performance of logistic regression

	in-sample				out-of-sample			
label	precision	recall	f1-score	support	precision	recall	f1-score	support
-1	0.91	0.88	0.89	1222	0.26	0.32	0.29	82
0	0.90	0.90	0.90	1347	0.38	0.35	0.36	121
1	0.89	0.92	0.91	1271	0.37	0.34	0.35	97
avg / total	0.90	0.90	0.90	3840	0.34	0.34	0.34	300

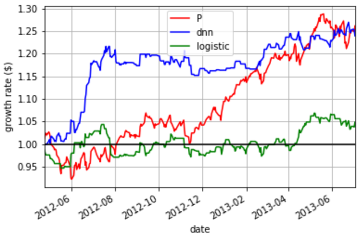
Table: The bias-variance tradeoff is characterized by the difference between the in-sample and out-of-sample performance.

Example: performance of deep neural networks

	in-sample				out-of-sample			
label	precision	recall	f1-score	support	precision	recall	f1-score	support
-1	0.94	0.95	0.94	1222	0.35	0.35	0.35	82
0	0.97	0.93	0.95	1347	0.44	0.52	0.48	121
1	0.93	0.94	0.93	1271	0.41	0.32	0.36	97
avg / total	0.95	0.94	0.94	3840	0.41	0.41	0.41	300

Table: The bias-variance tradeoff is characterized by the difference between the in-sample and out-of-sample performance.

Strategy Performance Measurement[13]



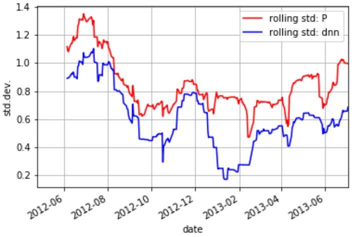


Figure: (left) The DNN based strategy is observed, over the short term, to outperform the portfolio P. (right) The DNN based strategy is also observed to exhibit less risk than the portfolio P.

[13] The performance measurement does not incorporate transaction costs.

Summary

- Deep learning is a powerful algorithmic modeling approach [Breiman,2001] for high dimensional nonlinear data reduction which does *not* assume a data generation process.
- Deep learners are best suited to high dimensional feature spaces
- Our research codes show how TensorFlow, an open source deep learning library, can be used to build price prediction models for market making strategies
- Current research is addressing how to improve interpretability of the model