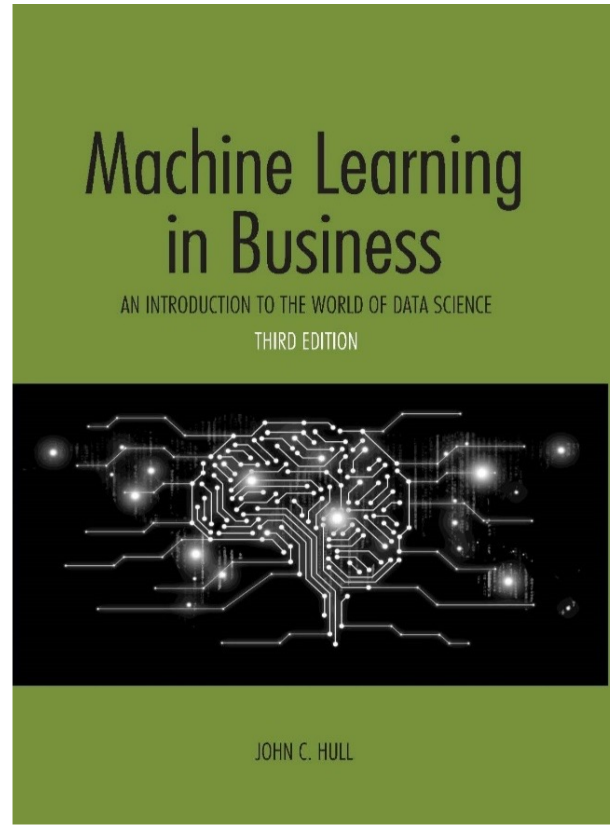
Introduction to Machine Learning

John Hull
Joseph L. Rotman School of Management
University of Toronto

Background

- This presentation is based on the 3rd edition of my book “Machine Learning in Business: An Introduction to the World of Data Science”
- For more information on the book, see

  www-2.rotman.utoronto.ca/~hull
- The 4th edition, with 3 co-authors, will be out shortly



The book cover courtesy of John Hull. Used with permission.

Plan for Today

- Nature of ML
- Neural networks
- Reinforcement learning

What is Machine Learning?

- Machine learning is a branch of AI
- The idea underlying machine learning is that we give a computer program access to lots of data and let it learn about relationships between variables and make predictions
- Some of the techniques of machine learning date back to the 1950s but improvements in computer speeds and data storage costs have now made machine learning a practical tool

Machine Learning vs. Automation

- Computers have been used to automate many business decisions (payroll, sending out invoices, summarizing sales by region, etc)
- This is digitization: the third industrial revolution
- Machine learning is central to the fourth industrial revolution where computers are used to create intelligence

Example: Loan Applications

- If loan officers applied certain known rules we could automate their activities (digitization)
- If we did not know the rules used, we could use ML to determine them
- But we could go one step further and use ML to improve upon the rules for accepting or rejecting loans

Traditional Statistics vs Machine Learning

Statistics

Develop Hypothesis → Collect Data → Test Hypothesis

Machine Learning

Collect Data → Try different models → Find Patterns or Develop a Predictive Tool

Types of Machine Learning Algorithms

- Unsupervised learning (clustering)
- Supervised learning (predict numerical value or classification)
- Reinforcement learning (multi-stage decision making)

Machine learning has its own terminology (different from statistics): features, targets, labels, activation functions….

Unsupervised Learning

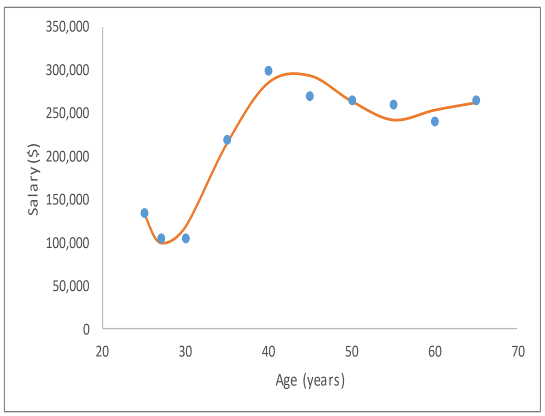
- In a typical application we divide data into clusters so that observations in the same cluster have similar feature values
- Example: it can help a company understand its customers better

Supervised Learning

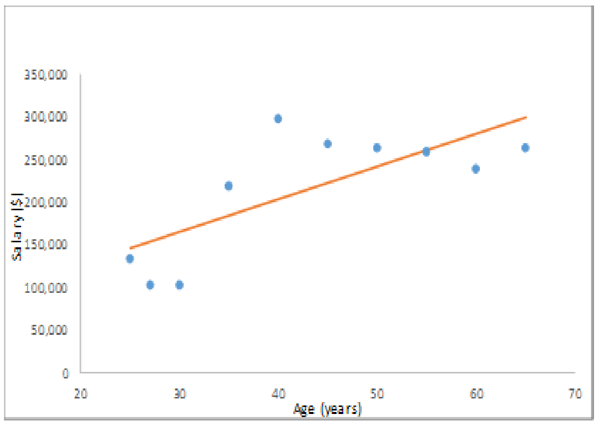
- Divide data into three sets
  - Training set
  - Validation set
  - Test set
- Develop different models using the training set and compare them using the validation set
- Rule of thumb: increase model complexity until model starts to perform worse on the validation set
- The test set is used to provide a final out-of-sample indication of how well the chosen model works

Overfitting/Underfitting;

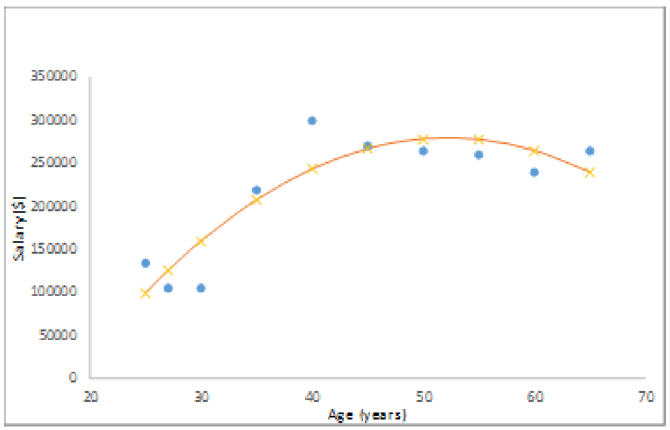
Example: predicting salaries for people in a certain profession in a certain area as a function of age



Overfitting



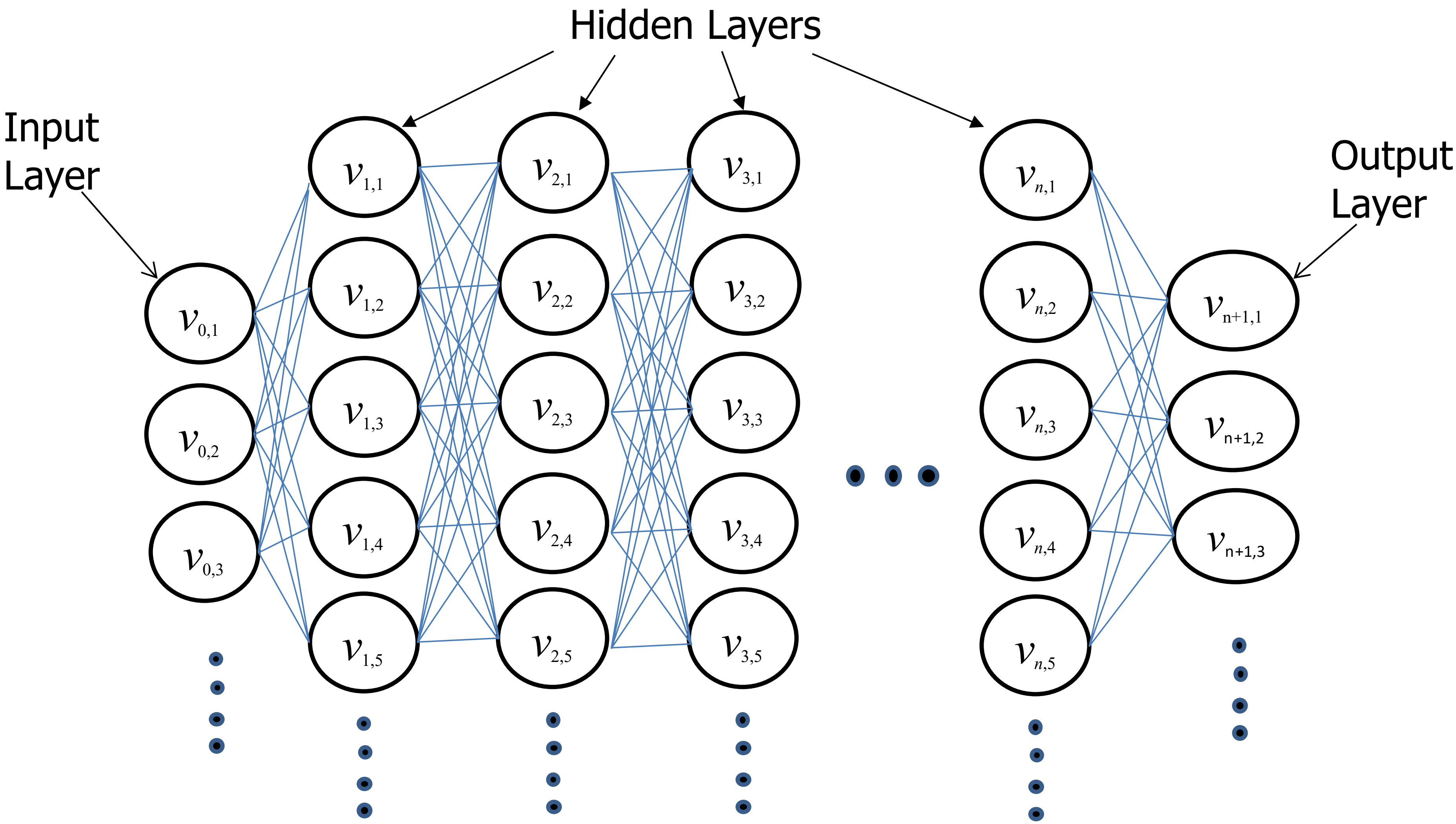
Underfitting



Best model?

Neural Networks

An Artificial Neural Network (ANN)



Activation Function

- An activation function relates values at one neuron to a linear combination of values in the previous layer.
- Popular activation functions are:

Identity: $f(y) = y$

Sigmoid: $f(y) = \frac{1}{1+e^{-y}}$ (gives values between 0 and 1)

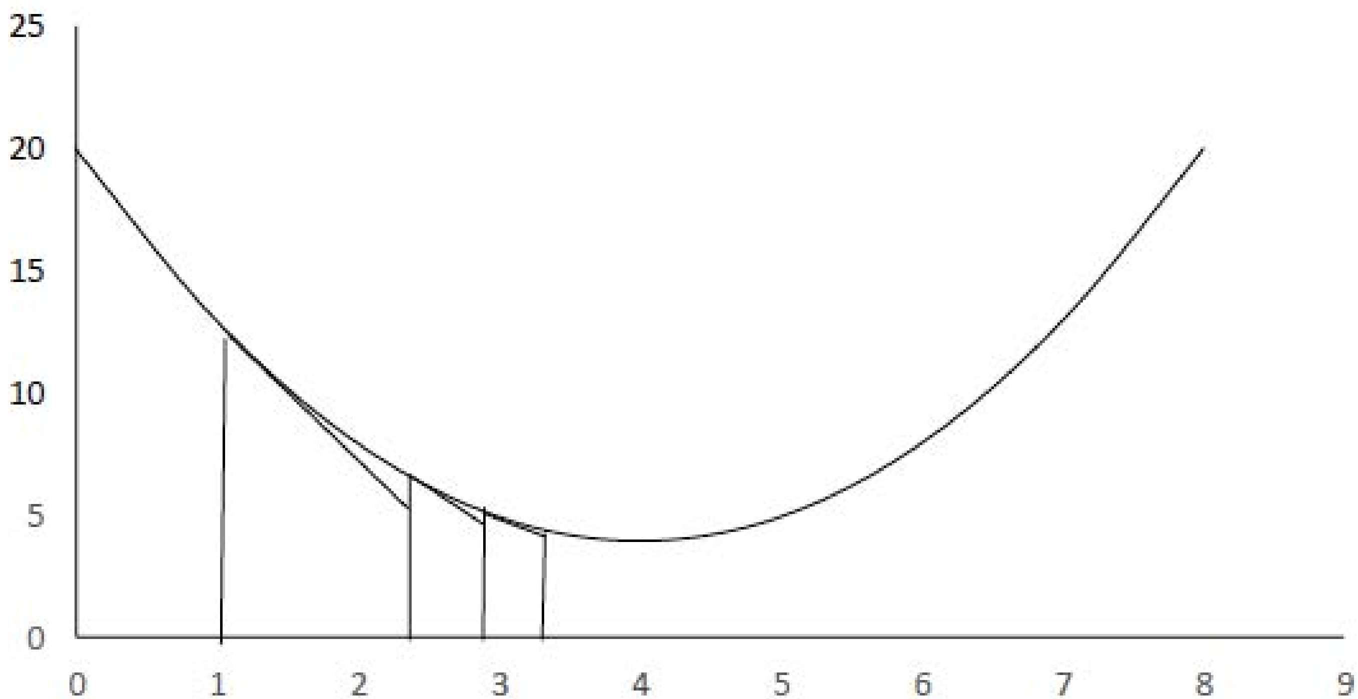
Hyperbolic tangent: $f(y) = \frac{e^{2y}-1}{e^{2y}+1}$ (gives values between -1 and 1)

Relu: $f(y) = \max(y, 0)$

Gradient Descent and Neural Nets

- Choosing the parameters to minimize a function in a neural network is like stepping down a valley
- Choose initial parameter values and a learning rate
- Calculate gradients to determine the direction in which parameter values can be improved
- Take a step to update parameter values (reduce each one by its gradient times the learning rate)
- Calculate gradients again
- Take another step to update parameter values
- etc

Very Simple Example: Calculating the value of x that minimizes y when $y=x^2-8x+20$



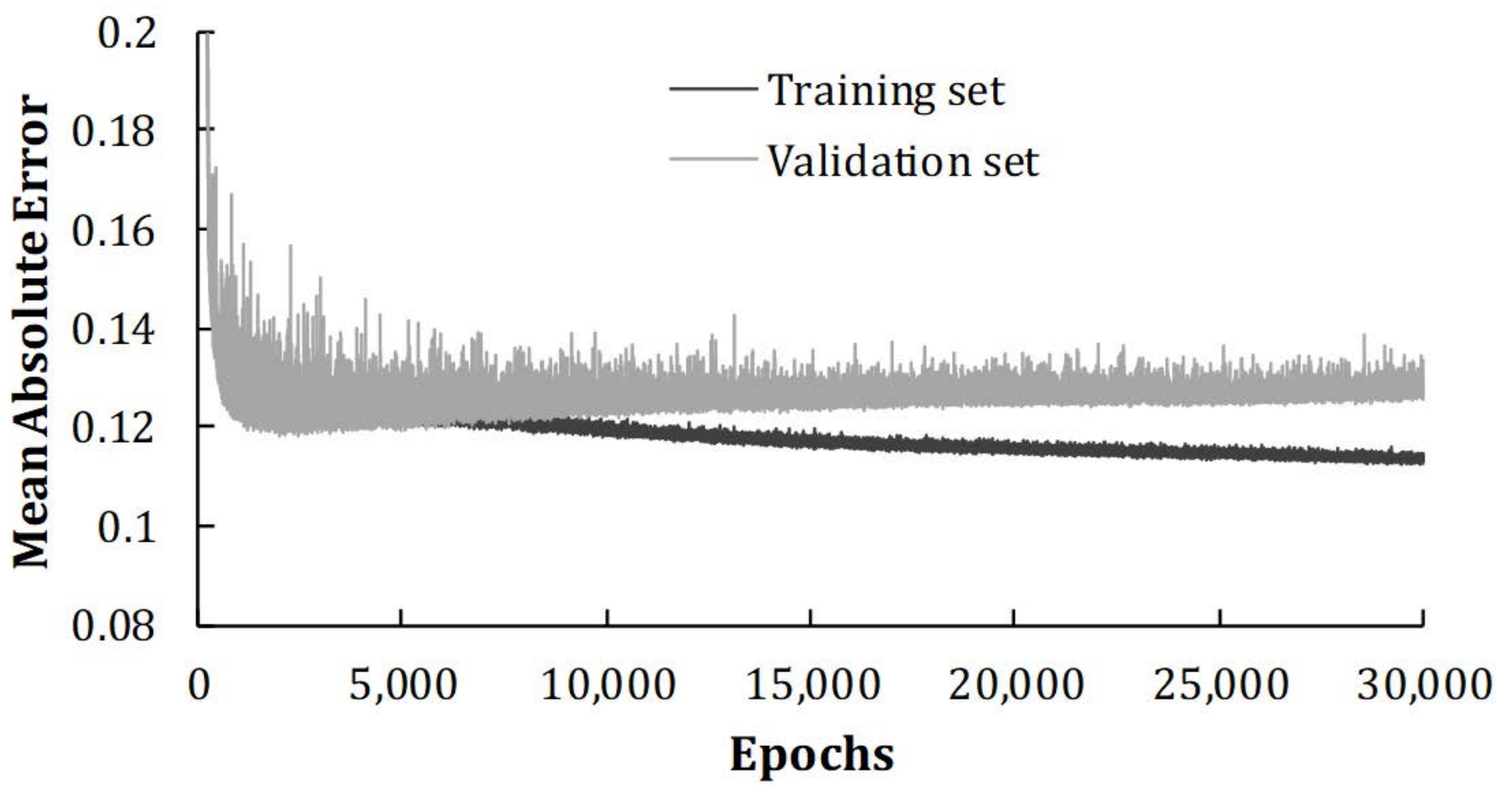
Other points

- We continue training the model (taking steps toward the bottom of the valley) until results for the validation set diverge from those for the training set
- Choosing learning rate is important
- Method works best when variables have been scaled
- Backpropagation is used to calculate partial derivatives

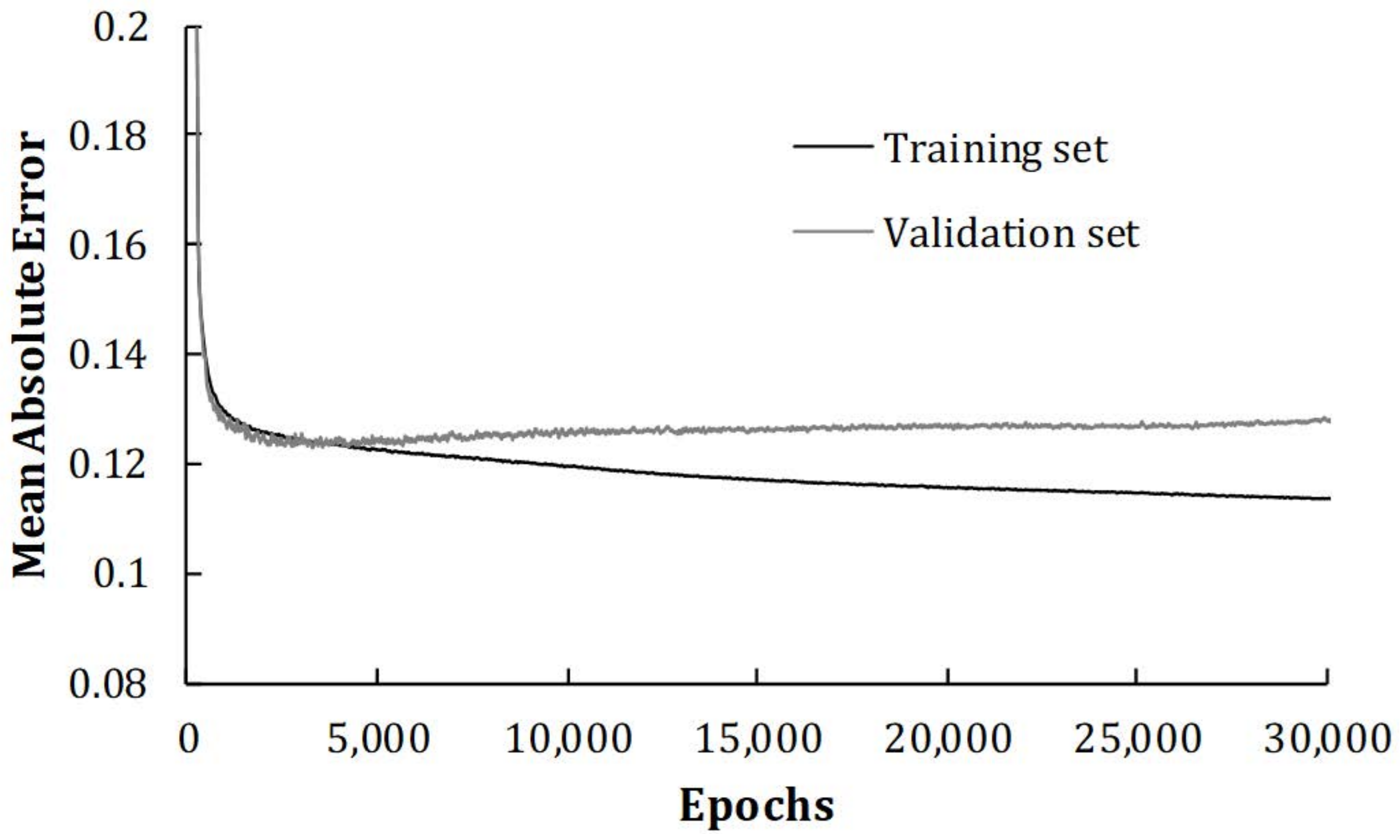
Black-Scholes-Merton Application

- We generated 10,000 call option prices using the Black-Scholes-Merton model and then added a normally distributed error (mean=0, SD=0.15) to the price.
- The parameters were sampled randomly (S between 40 and, 60, K between 0.5S and 1.5S, r between 0 and 5%, σ between 10% and 40%, T between 0.25 and 2 years)
- The model had three hidden layers and 20 neurons per layer
- Activation function was sigmoid (except final layer)
- 6,000 observations in training set, 2,000 in validation set, 2,000 in test set

Training set and validation set mse as the epochs of training is increased



Smoothed mse (Moving Average over 50 epochs) Stop after 2575 epochs



Results

- With only 10,000 observations the neural network imitated the Black-Scholes-Merton model very well
- It overcame the random noise we added to the BSM prices.

Using a similar idea to value exotic derivatives

- Some “exotic” derivatives have traditionally been valued using Monte Carlo simulation, which is slow
- Neural networks can be used as follows:
  - Do an initial analysis to generate a large amount of data relating prices to input variables
  - Construct a neural network to replicate prices
  - Obtain fast pricing by working forward through network
- Useful for scenario analysis

Implied volatilities

- The volatility surface shows implied volatilities as a function of
  - Moneyness or how likely the option is to be exercised
  - Time to maturity, T
- There is a non-zero correlation between implied volatility changes and asset price changes
- This means that calculating delta using the current implied volatility may be suboptimal

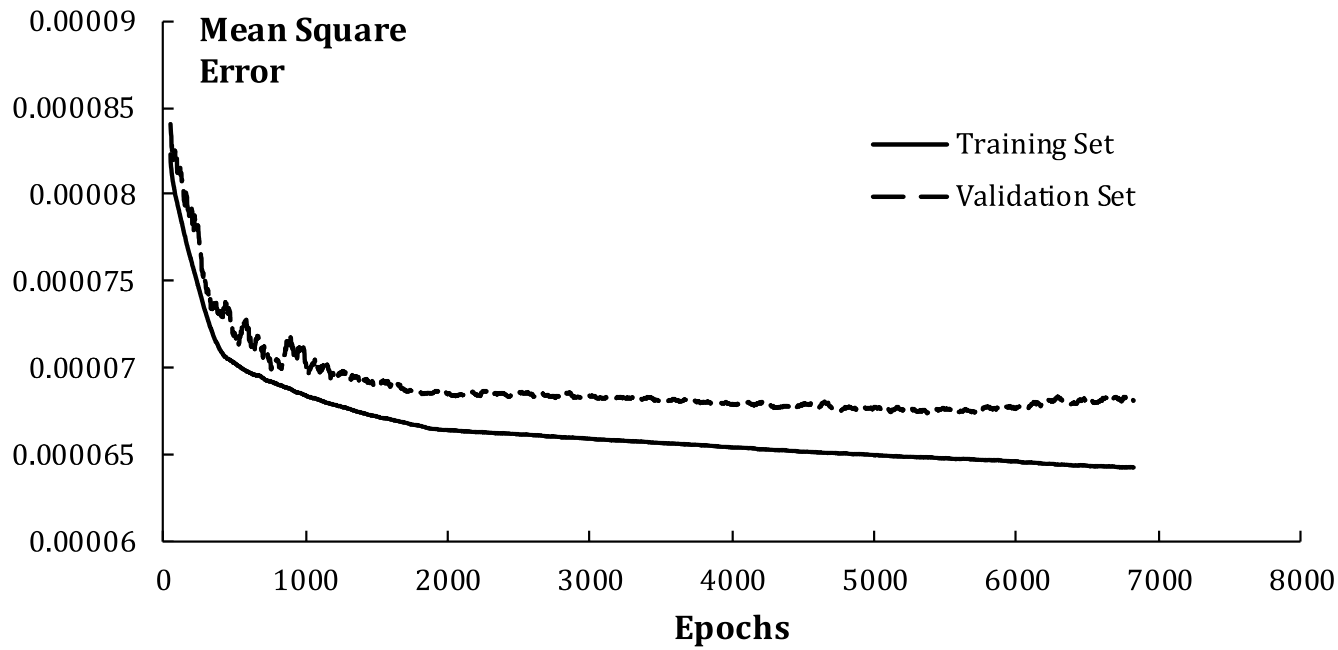
Understanding Volatility Surface Movements

- To understand volatility surface movements we used data on S&P 500 call options to construct a neural network
- Input layer:
  - Daily asset price return
  - Moneyness (measured by delta)
  - Time to maturity
- Output layer:
  - Change in implied volatility

Details

- 3 hidden layers
- 20 neurons per layer
- Observations from 2014-2019
- Randomly sampled 100 options per day
- 125,700 options in total
- 60% for training set
- 20% for validation set
- 20% for test set
- Experimented with different activation functions (sigmoid best)

Mean Squared Error (Training stopped after 5,826 epochs)



Results

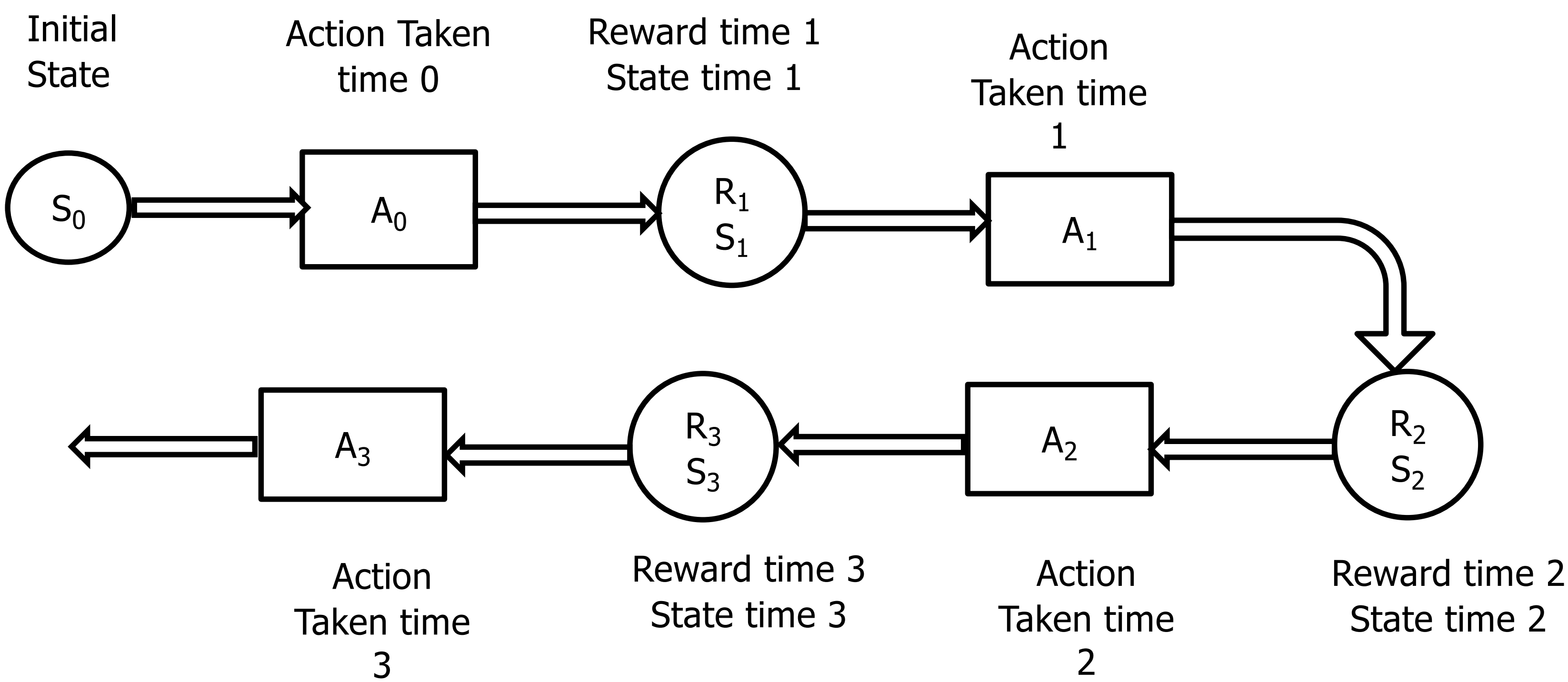
- Test set gave a modest 11% improvement over a simple analytic model proposed by Hull and White in 2017
- However, when the VIX index on Day *t* was used as a feature to predict changes between Day *t* and Day *t*+1 there was a further improvement of over 60%
- The behavior of the volatility surface is different in high and low volatility environments

Reinforcement Learning

Reinforcement Learning

- Reinforcement learning is concerned with finding a strategy for taking a series of decisions rather than just one
- The environment is usually changing unpredictably
- It has been used to develop software that can beat the best human players of chess and Go

The General Model: Actions, States, and Rewards



Exploration vs. Exploitation

- At any given time there is a best decision based on the data you have processed so far.
- In trials to develop your best strategy
  - there is a probability ε that you will randomly chose a decision (referred to as exploration)
  - There is a probability $1-\varepsilon$ that you will choose the best decision identified so far (referred to as exploitation)
- Typically the probability of exploration, ε, is initially one and reduces with the number of trials
- Probability of exploration on trial n equals probability of exploration on trial n-1 times a “decay factor”. The decay factor (e.g. 0.995) is a hyperparameter chosen by trial and error

A Simple Example: Nim

- Two players, *n* matches in pile initially. Each player picks up 1, 2, or 3 matches. What is the best strategy to avoid picking up the last match?
- Assume only 8 matches initially
  - Reward = +1 from winning and -1 from losing
  - Opponent behaves randomly
- State, S, =number of matches; Action, A, =number picked up
- $Q(S, A)$ is value of taking action A in state S. Initially the Q's are zero:

Action A	State, S: Number of matches							
No. Picked Up	1	2	3	4	5	6	7	8
1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2		0.000	0.000	0.000	0.000	0.000	0.000	0.000
3			0.000	0.000	0.000	0.000	0.000	0.000

Nim continued

Suppose the matches picked up on first simulation (with opponent's choice in brackets) is

1,[3],1,[3] (win)

We can update Q(8,1) and Q(4,1) to 0.05

The updating formula is

$$Q(S,A)^{new} = Q(S,A)^{old} + \alpha\left[G - Q(S,A)^{old}\right]$$

where G is the gain on the trial and α (e.g. 0.05) is a hyperparameter chosen by trial and error.

Updating

- On each trial we observe certain states and take actions in those states.
- The gain used in updating $Q(S, A)$ can be either:
  - The final reward (referred to as the "Monte Carlo method")
  - The value at the next state assuming best decision is taken (referred to as "temporal difference learning" or "*Q*-learning")

Example of convergence. ε starts at 1 and has a decay factor of 0.9995. Monte Carlo method

After 1,000 trials

Number Picked up	Number of matches 1	2	3	4	5	6	7	8
1	-1.000	0.999	0.104	0.479	-0.180	0.482	0.000	0.562
2		-0.993	0.996	0.306	0.303	-0.076	0.000	0.193
3			-0.976	1.000	0.110	0.259	0.000	0.522

After 5,000 trials

Number Picked up	Number of matches 1	2	3	4	5	6	7	8
1	-1.000	1.000	0.068	0.084	-0.072	0.831	0.000	0.491
2		-1.000	1.000	0.074	0.172	-0.105	0.000	0.544
3			-1.000	1.000	-0.020	0.065	0.000	0.960

After 10,000 trials

Number Picked up	Number of matches 1	2	3	4	5	6	7	8
1	-1.000	1.000	0.194	0.190	-0.018	0.913	0.000	0.674
2		-1.000	1.000	-0.160	0.426	-0.150	0.000	0.793
3			-1.000	1.000	0.031	0.065	0.000	1.000

Example of convergence. ε starts at 1 and has a decay factor of 0.9995. Temporal Difference Learning

After 1,000 trials

Number Picked up	Number of matches 1	2	3	4	5	6	7	8
1	-1.000	1.000	0.559	0.660	0.626	0.676	0.000	0.708
2		-0.996	0.998	0.411	0.551	0.296	0.000	0.852
3			-0.982	1.000	0.488	0.535	0.000	0.999

After 5,000 trials

Number Picked up	Number of matches 1	2	3	4	5	6	7	8
1	-1.000	1.000	0.521	0.526	0.644	0.997	0.000	0.851
2		-1.000	1.000	0.518	0.489	0.401	0.000	0.832
3			-1.000	1.000	0.509	0.549	0.000	1.000

After 10,000 trials

Number Picked up	Number of matches 1	2	3	4	5	6	7	8
1	-1.000	1.000	0.618	0.586	0.654	0.999	0.000	0.813
2		-1.000	1.000	0.400	0.465	0.381	0.000	0.910
3			-1.000	1.000	0.509	0.549	0.000	1.000

When there are many states or actions (or both)

- The cells of the state/action table do not get filled in very quickly
- It becomes necessary to estimate the complete $Q(S, A)$ function from observed values.
- As this function is in general non-linear a natural approach is to use artificial neural networks (ANNs).
- We use an ANN to minimize the sum of squared errors between the estimates and the target
- This is known as deep *Q*-learning or deep reinforcement learning

Derivatives Hedging Applications

- Traditional approach is to use Greek letters (delta, gamma, vega, etc)
- Can reinforcement learning (i.e., looking several periods ahead) produce improvements?

Our Conclusions *(see papers on my website)*

Advantages of using RL for hedging

- For vanilla options it saves transaction costs
- For some exotics such as barrier options it produces superior results even when there are no transaction costs
- The user can choose the objective function (e.g. VaR95, CVaR95)
- It is robust and gives good results during stressed periods

Disadvantage: it is much more computationally demanding than traditional approaches