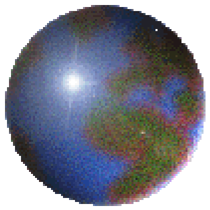
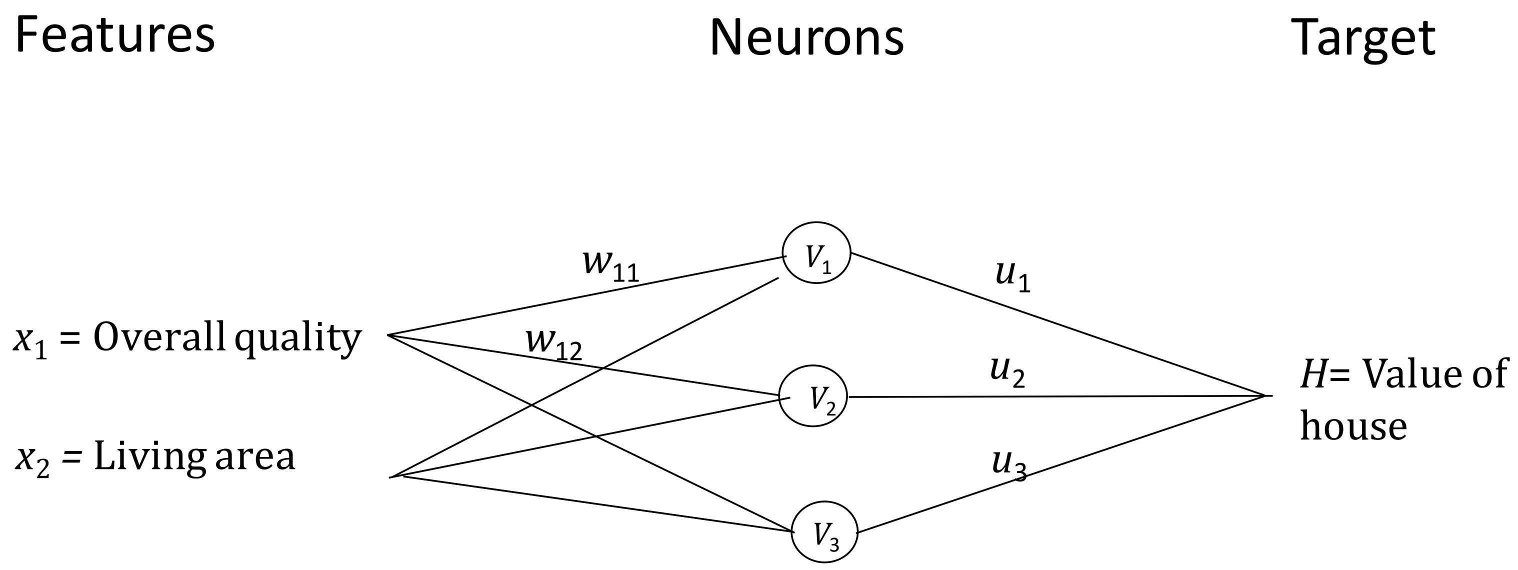


Machine Learning in Business
John C. Hull

Chapter 6

Supervised Learning: Neural Networks

A Simple ANN (Artificial Neural Network), Figure 6.1



Activation Functions

- H is not related directly to the features
- There is a hidden layer with neurons
- H is related to the V_k
- The V_k are related to the features
- Activation functions define the relationships

Activation Functions continued

- Popular activation functions are:

  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: $f(y) = \max(y, 0)$
- These could be used to relate the V_k to the features
- To calculate the value of a continuous variable from the hidden layer we usually use a linear function: $f(y) = y$
- The argument, y, of the function is a bias (constant) plus linear combination of the preceding values

One possible set of equations for the house price example

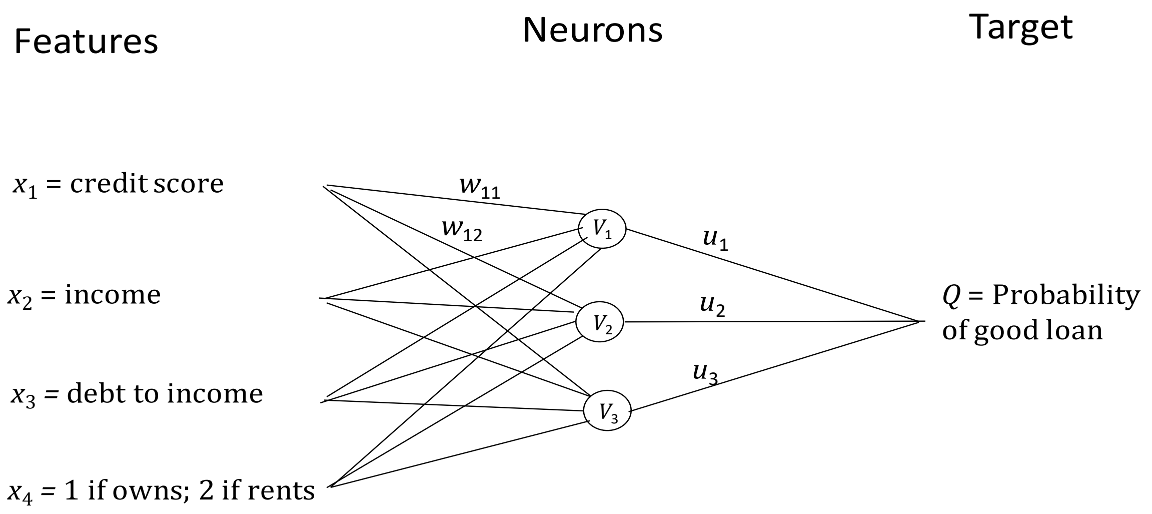
$$v_1 = \frac{1}{1+\exp(-a_1 - w_{11}x_1 - w_{21}x_2)}$$

$$v_2 = \frac{1}{1+\exp(-a_2 - w_{12}x_1 - w_{22}x_2)}$$

$$v_3 = \frac{1}{1+\exp(-a_3 - w_{13}x_1 - w_{23}x_2)}$$

$$H = c + u_1 v_1 + u_2 v_2 + u_3 v_3$$

Another Example (Figure 6.2)



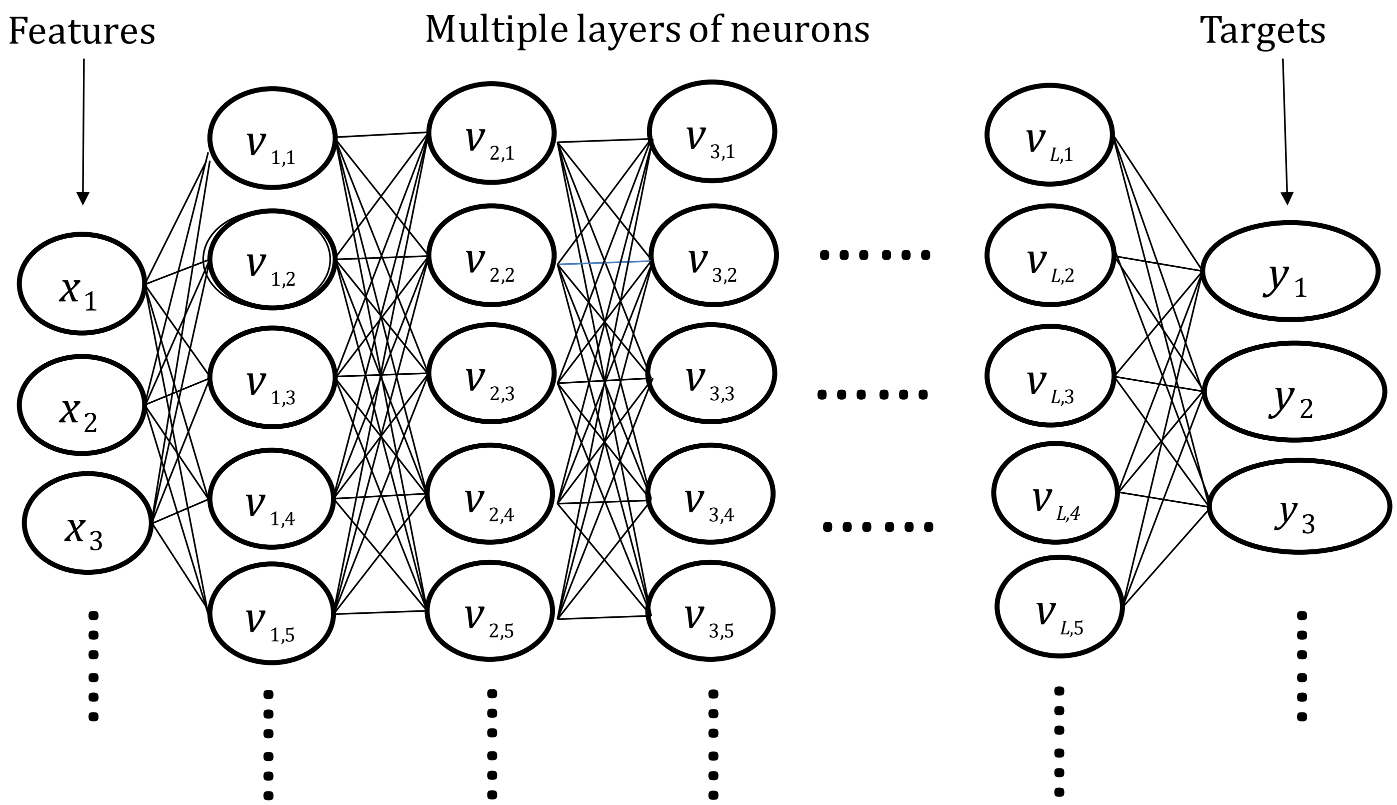
This works in the same way as the previous example except that the sigmoid function would usually be used to relate Q to the V "s

$$Q = \frac{1}{1 + \exp(-c - u_1 v_1 - u_2 v_2 - u_3 v_3)}$$

Universal Approximation Theorem

- Any continuous function can be approximated to arbitrary accuracy with one hidden layer (See K. Hornik: *Neural Networks*, 1991, 4:251-257)
- But this may require a very large number of neurons.
- Using several hidden layers can be computationally more efficient

A Generalization (Figure 6.3)



Number of Parameters

- Neural networks can have a very large number of parameters
- If there are F features, H hidden layers, M neurons in each hidden layer and T targets the number of parameters is

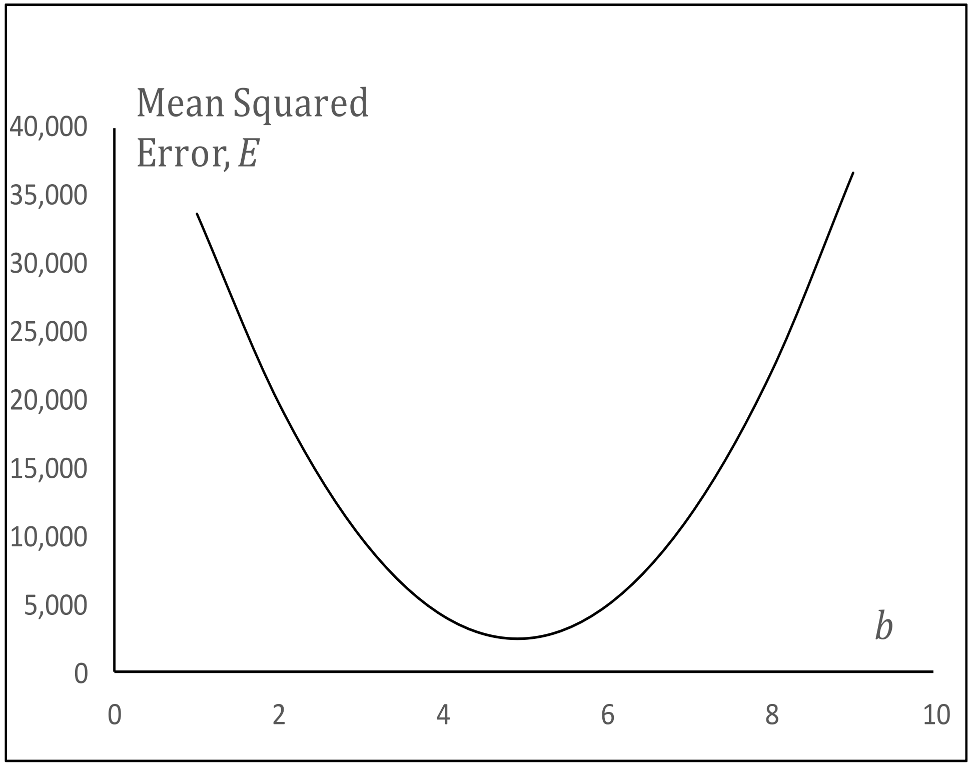
$$(F+1)M+M(M+1)(H-1)+(M+1)T$$

- The first (house price) neural network has 13 parameters and the second (credit) neural network has 19 parameters
- But in practice a small neural network might have 4 features, 3 hidden layers, 80 neurons per layer and one target for a total of 13,441 parameters.
- Larger neural networks have hundreds of thousands of parameters

Gradient Descent and Neural Nets

- To minimize an objective function such as mse, a gradient descent algorithm calculates the direction of steepest descent, takes a step, calculate a new direction of steepest descent, takes another step, and so on
- The partial derivatives with respect to the parameters are calculated by a procedure known as backpropagation. We work back through the network using the chain rule. (See Runnelhart, Hinton, and Williams, *Nature*, 1986, 323, 533-536)
- The size of the step is the learning rate. If the step is too small the algorithm will be very slow. If it is too large there are liable to be oscillations

Simple Example: Calculating a single regression coefficient, b (Figure 6.4)



When Learning Rate is 0.0002 (Table 6.2)

Iteration	**Value of *b***	**Gradient**	**Change in *b***
0	1.0000	−15,986.20	+3.1972
1	4.1972	−2,906.93	+0.5814
2	4.7786	−528.60	+0.1057
3	4.8843	−96.12	+0.0192
4	4.9036	−17.48	+0.0035
5	4.9071	−3.18	+0.0006
6	4.9077	−0.58	+0.0001
7	4.9078	−0.11	+0.0000
8	4.9078	−0.02	+0.0000
9	4.9078	0.00	+0.0000

When Learning Rate is 0.00001 (Table 6.3)

Iteration	Value of *b*	Gradient	Change in *b*
0	1.0000	−15,986.20	+0.1599
1	1.1599	−15,332.24	+0.1533
2	1.3132	−14,705.03	+0.1471
3	1.4602	−14,104.47	+0.1410
4	1.6013	−13,526.53	+0.1353
5	1.7365	−12,973.18	+0.1297
6	1.8663	−12,442.48	+0.1244
7	1.9907	−11,933.48	+0.1193
8	2.1100	−11,445.31	+0.1145
9	2.2245	−10,977.37	+0.1098

When Learning Rate is 0.0005 (Table 6.4)

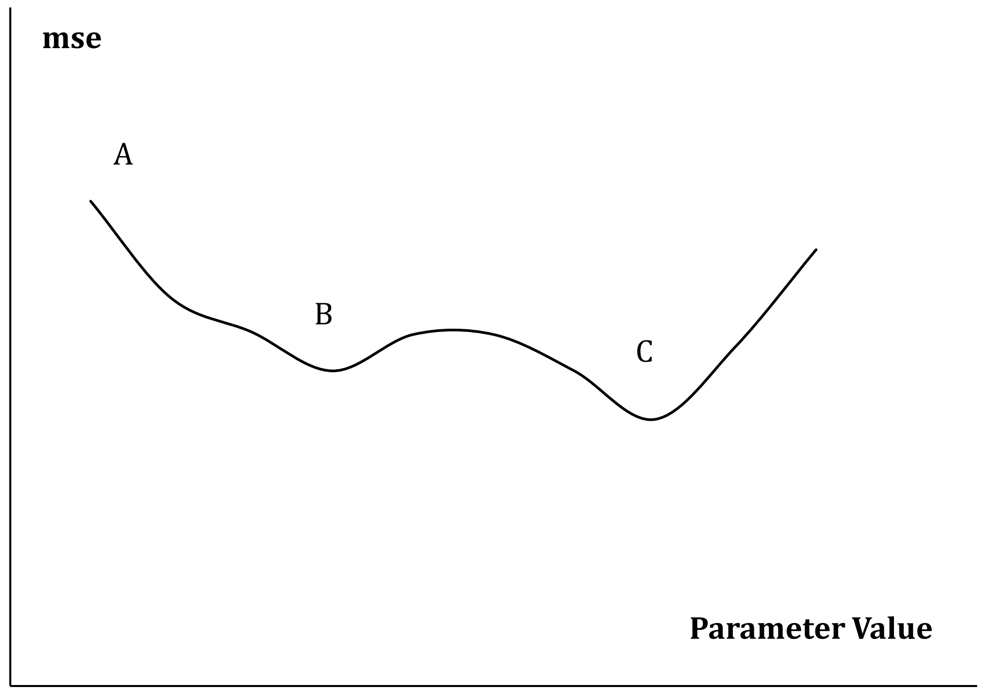
Iteration	Value of *b*	Gradient	Change in *b*
0	1.0000	−15,986.20	+7.9931
1	8.9931	16,711.97	−8.3560
2	0.6371	−17,470.70	+8.7353
3	9.3725	18,263.87	−9.1319
4	0.2405	−19,093.05	+9.5465
5	9.7871	19,959.87	−9.9799
6	−0.1929	−20,866.05	+10.4330
7	10.2401	21,813.37	−10.9067
8	0.6665	−22,803.69	+11.4018
9	10.7353	23,838.98	−11.9195

Other Details

- Use mini batches of data to calculate gradients. (One epoch is one complete use of training set.)
- Use a momentum strategy
- Adaptively calculate the learning rate (Adam)
- Learning rate decay
- Dropouts

Local Minima

Important to try and avoid local minima



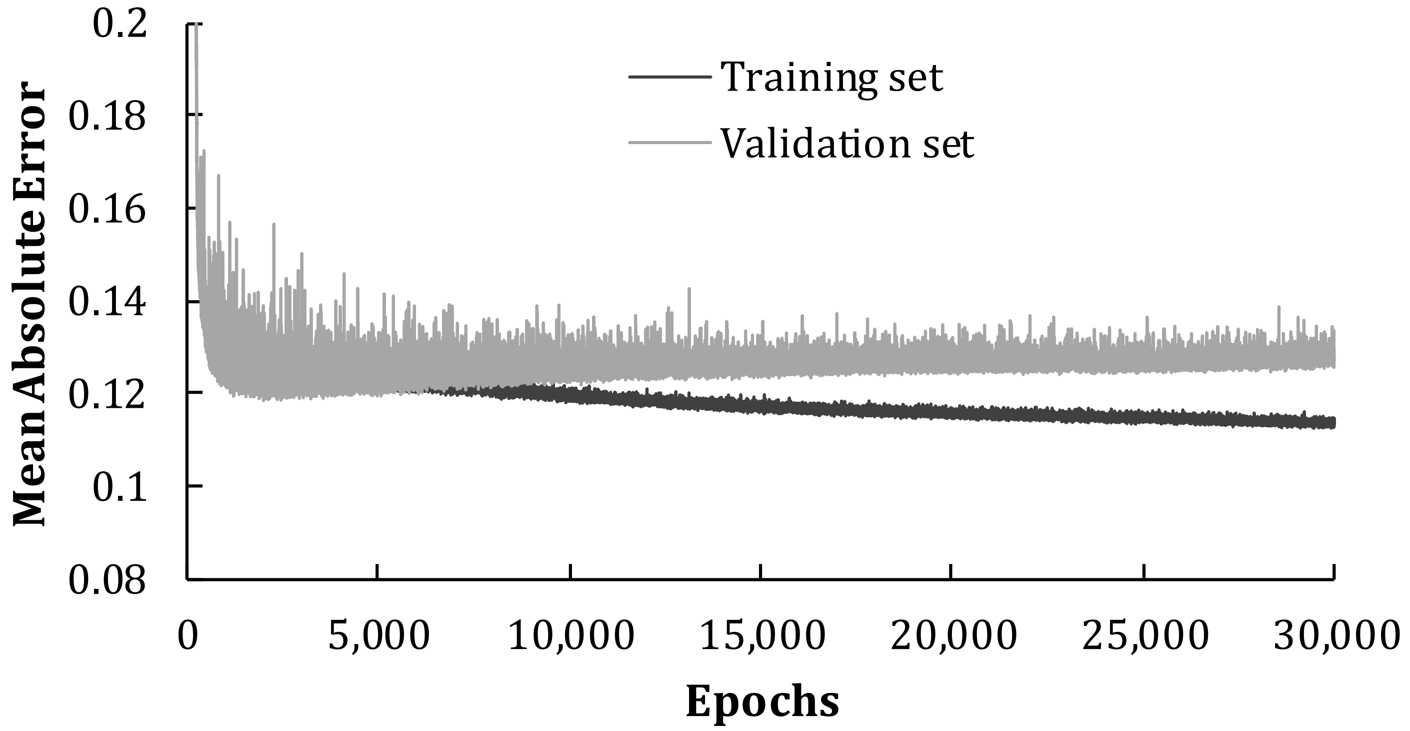
Stopping Rule

- Because applications have many parameters it is important to use a stopping rule to avoid over-fitting
- We calculate results for the validation set at the same time as the training set.
- When the results for the validation set start to get worse we stop

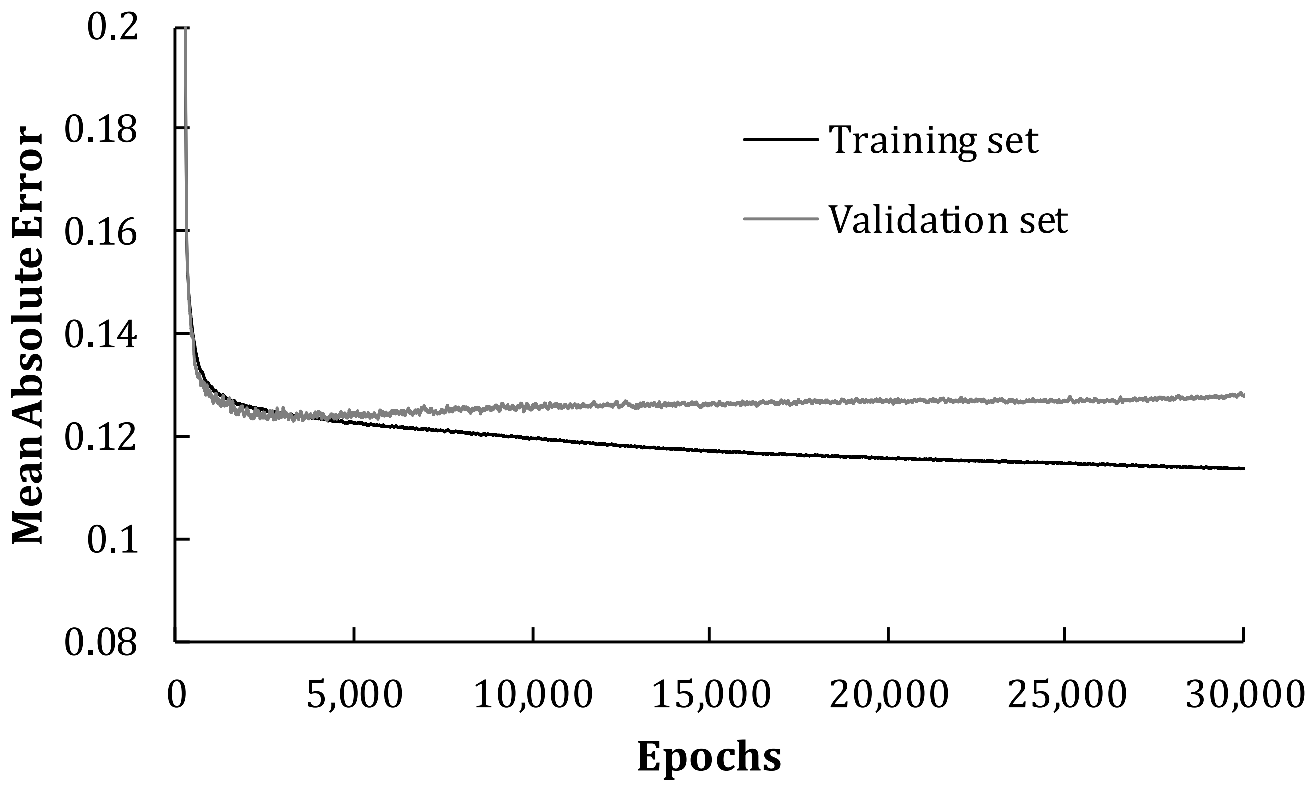
Black-Scholes-Merton Application

- We generated 10,000 call option prices using the Black-Scholes-Merton model and then added a normally distributed error of 0.15 to the price.
- The parameters were sampled randomly from uniform distributions
- The model had three hidden layers and 20 neurons per layer

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



Smoothed mse (Moving Average over 50 epochs)



Results

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

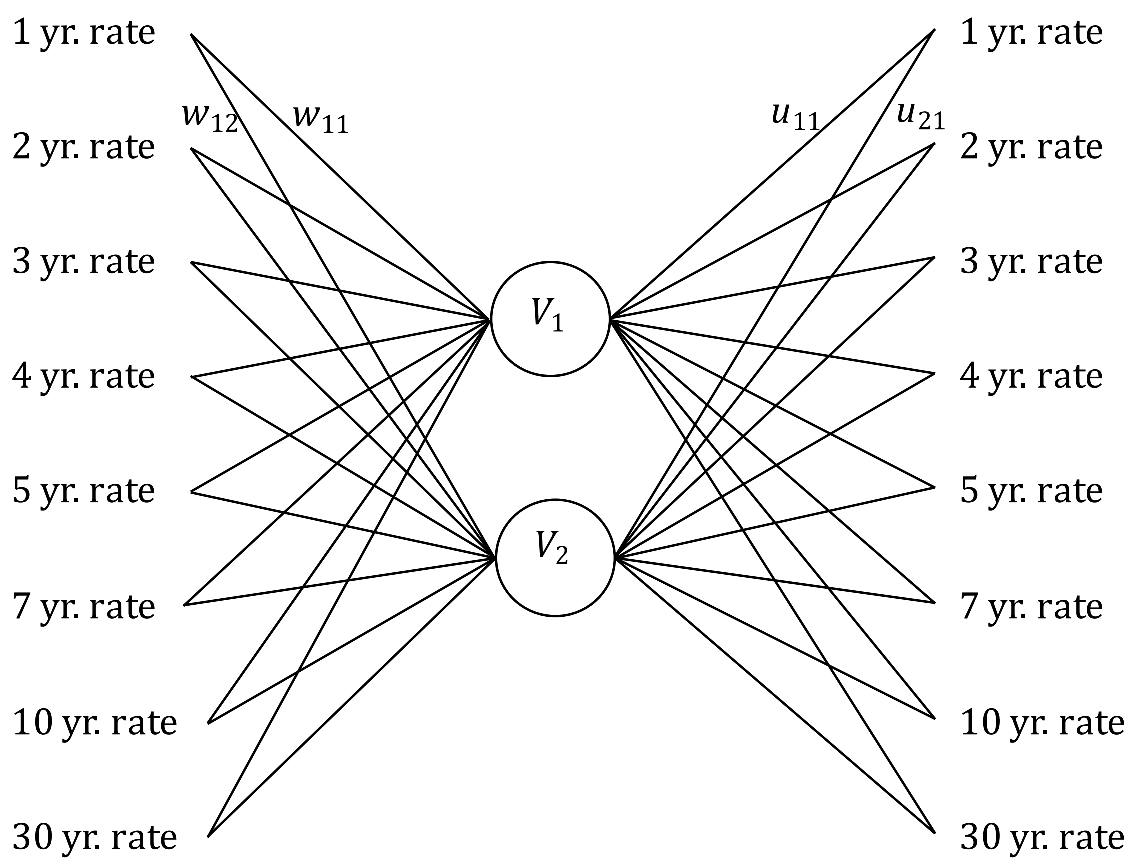
Application to Derivatives

- Some instruments are valued using Monte Carlo simulation
- But banks often have to use Monte Carlo simulation for scenario analysis and to calculate the VAs
- There is then a “Monte Carlo within Monte Carlo” problem.
- To solve the problem you can do an initial analysis to generate a large amount of data relating prices to input variables
- Use a neural network to get a good fast approximation for a pricing model

Autoencoders

- In an autoencoder the input and output are the same
- The objective is similar to PCA: to find a small number of features that capture nearly all the properties of the data

Using an autoencoder for interest rates



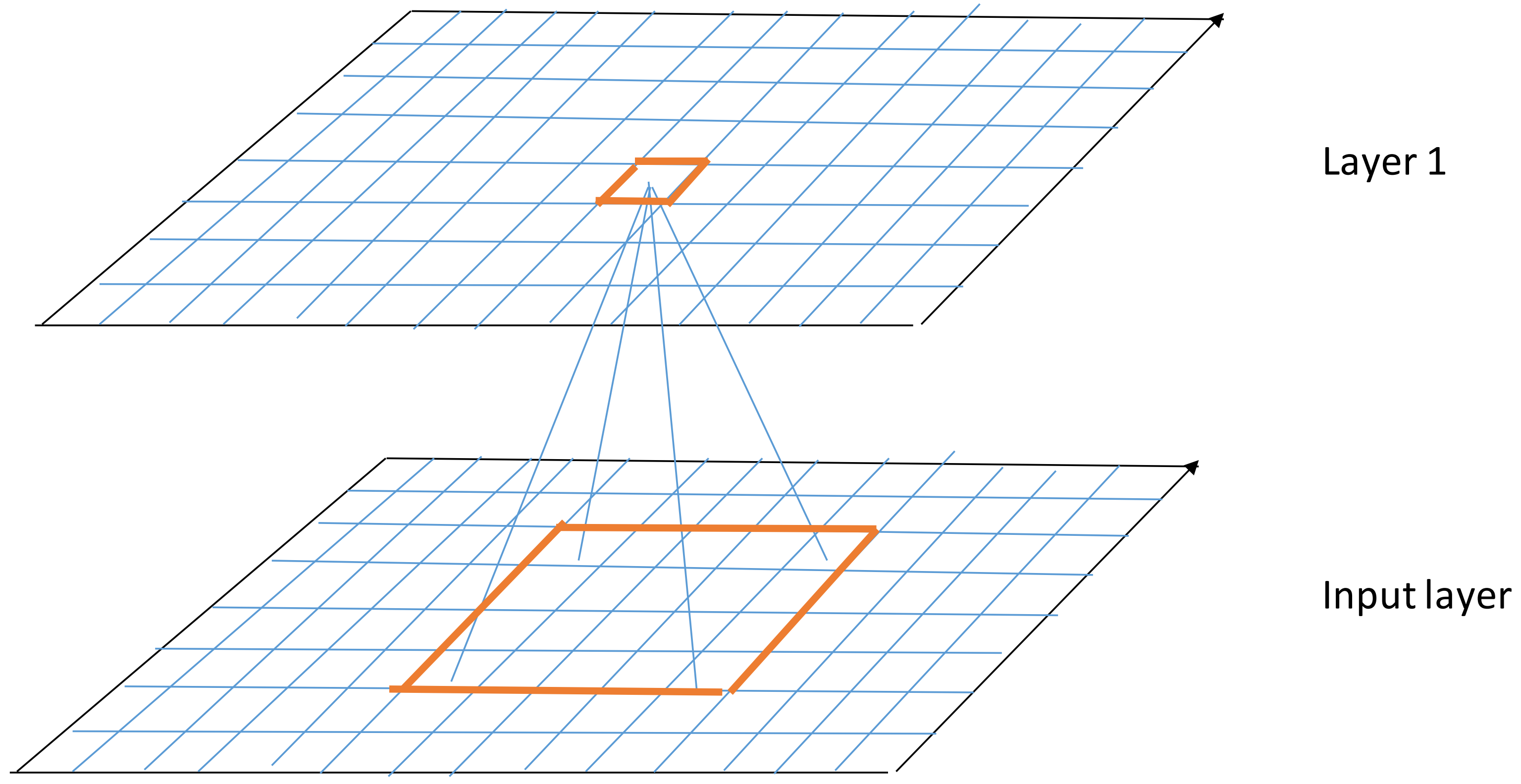
Results

- An autoencoder with linear activation functions produces an equivalent result to PCA
- The PCA factors are uncorrelated and easier to use
- But autoencoders can use non-linear activation functions
- They have been found to be useful in language translation and image recognition

Convolutional Neural Network (CNN)

- In a vanilla ANN the layers are fully connected which can give rise to a very large number of parameters
- In a convolutional neural network, the number of neurons in one layer that affect the next layer is reduced.
- Used for image recognition where each pixel can be a feature
- Consistent with the way neurons in the eye work

A "receptive field" in input layer determines value at a neuron in "feature map" of first hidden layer



CNN: More Details

- There are several layers as in a regular neural network
- Each layer is actually three-dimensional consisting of several two-dimensional feature maps
- For each feature map, biases and weights are the same across all receptive fields
- There may be "padding" where extra observations are added at the edges to avoid successive layers becoming smaller

CNN: Key Advantages

- Consider a image that is 100×100 or 10,000 pixels
- A regular neural network would lead to 10,000×(10,000+1) or about 100 million parameters to define the first layer.
- If a CNN’s receptive field is 10×10 and a layer has 6 feature maps only 6×101=606 are required
- More importantly, once the CNN has learned to recognize a pattern in one location it can recognize it in another location.

Recurrent Neural Network (RNN)

- In a vanilla ANN, the *v*'s are functions of the inputs and the sequence of the inputs does not matter
- In a recurrent neural network there is a time dimension to the data. The current *v*'s are made functions of the corresponding previous *v*'s (or possibly previous outputs) as well as current inputs
- A long short-term memory (LSTM) network is a type of RNN where part of the algorithm learns what should be remembered and what should be forgotten from previous data

RNN: Applications

- Sequential data where the relationship between targets and features may be changing through time
- Particularly relevant for finance where we are often dealing with time series data
- Also use in natural language processing (e.g. autocomplete tool)