1 Main objectives and scope of the assignment
investigate properties of multi-layer perceptron network.

2 Methods
For part II, we used the Deep Learning Toolbox in MATLAB.

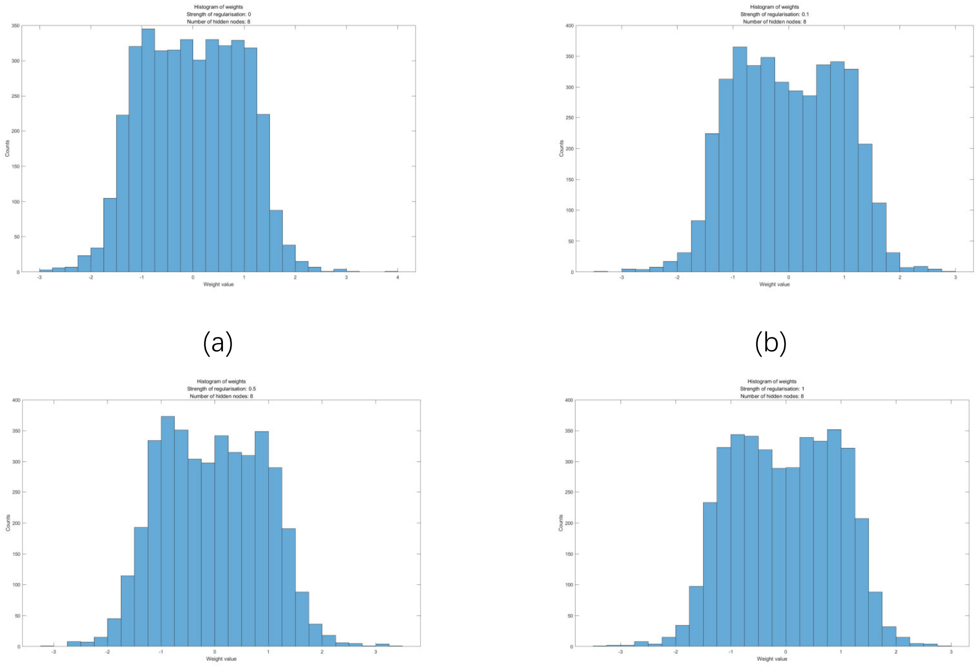
4 Results and discussion - Part II

The choice of network architecture was fitnet (specialized versions of the feedforward network), with the default early stopping criterion, max fail = 6. The training function used was trainscg (scaled conjugate gradient backpropagation). The regularization method used was a built-in method (L2). For each network configuration, the evaluation was repeated 100 times.

4.1 Two-layer perceptron for time series prediction - model selection, regularization and validation

	h=2	h=4	h=6	h=8
r=0	0.0062	0.0048	0.0030	0.0027
r=0.1	0.0067	0.0042	0.0039	0.0031
r=0.5	0.0075	0.0037	0.0031	0.0030
r=1	0.0077	0.0037	0.0037	0.0032

Table 1: Validation MSE for different regularization strengths and numbers of hidden nodes (r: regularization value, h: the number of hidden nodes)



(a)

(b)

(c)

(d)

Figure 1： histogram of weights for different regularization strengths

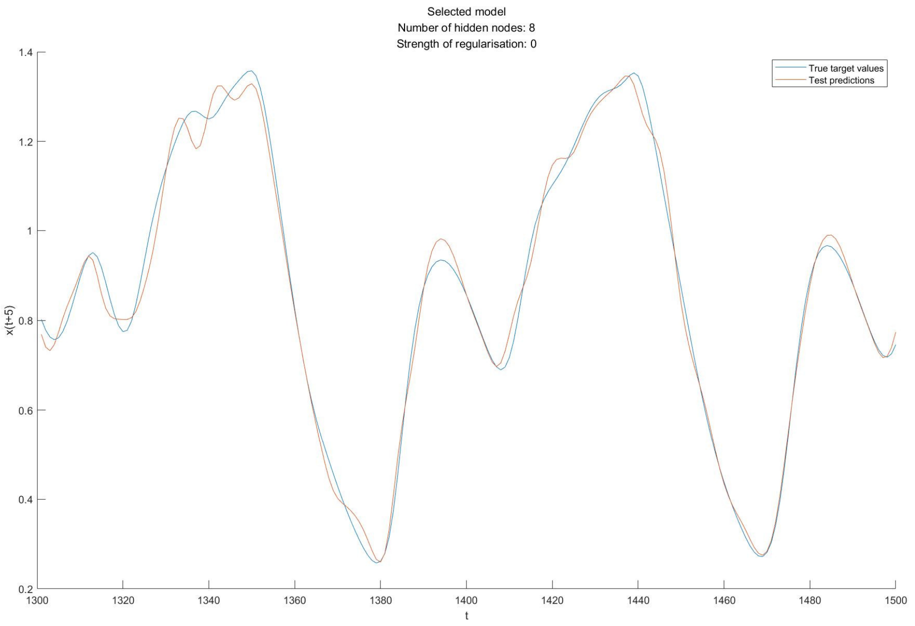


Figure 2: Plot of test predictions along with the true target values for the best configuration

A two-layer perceptron trained by no-noise data may not suffer from overfitting. Therefore, the more hidden nodes the better the validation performance. We use regularization to reduce overfitting but for this two-layer perceptron network, the greater the regularization the greater the hold-out validation error. Theoretically, stronger regularization should lead to smaller weights. Since the two-layer perceptron network may not suffer from overfitting, the regularization effect is not significant. The best network configuration was 8 hidden nodes with 0 regularization value.

4.2 Comparison of two- and three-layer perceptron for noisy time series prediction

	h=2	h=4	h=6	h=8
σ = 0.03	0.0059	0.0050	0.0060	0.0058
σ = 0.09	0.0086	0.0071	0.0068	0.0081
σ = 0.18	0.0120	0.0105	0.0095	0.0094

Table 2: Validation MSE for different amount of noise and numbers of hidden nodes (r=0)

The addition of noise is used to reduce overfitting. The larger the noise, the more significance the effect. Increasing the number of nodes in the second hidden layer reduced the validation MSE up to a certain point, after which the model began to suffer from overfitting. The validation prediction performance became worse.

	r=0	r=0.1	r=0.5	r=1
σ = 0.03	0.0056	0.0051	0.0068	0.0044
σ = 0.09	0.0060	0.0053	0.0056	0.0059

σ = 0.18	0.0087	0.0093	0.0105	0.0104

(a)

	r=0	r=0.1	r=0.5	r=1
σ = 0.03	0.0058	0.0054	0.0064	0.0048
σ = 0.09	0.0067	0.0057	0.0065	0.0067
σ = 0.18	0.0091	0.0098	0.0107	0.0108

(b)

Table 3: MSE for different amount of noise and different regularization strengths [8, 8] (a) training (b) validation

Both additive noise and regularization are used to reduce overfitting. For the most complex network ([8, 8]), the largest noise σ = 0.18 was enough to reduce overfitting. Therefore, the greater the regularization value the worse the validation performance. As the amount of noise increases, we should choose a smaller regularization value. For smaller noise, the model still suffers from overfitting. Increasing the regularization value reduced the validation MSE up to a certain point, after which the combined effect is enough. The validation prediction performance became worse. It is difficult to draw a conclusion for the training dataset.

	h=2	h=4	h=6	h=8
r=0	0.0088	0.0069	0.0076	0.0077
r=0.1	0.0110	0.0075	0.0069	0.0070
r=0.5	0.0106	0.0088	0.0073	0.0072
r=1	0.0088	0.0080	0.0070	0.0063

Table 4: Validation MSE for different regularization strengths and numbers of hidden nodes

no-noise data (2)	noisy data (2)	noisy data (3)
0.0030	0.0067	0.0069

Table 5: Test MSE of different models

The best three-layer model was configuration [8, 8] with 1 regularization value. Compared to the best two-layer model (8 hidden nodes with 0 regularization value), their generalization errors on the evaluation test set were similar. When comparing the model trained by no-noise data to the models trained with noisy data, we can see that noise made the generalization performance worse.

configuration	computation time (s)
[8]	1.0781
[8, 2]	1.1274
[8, 4]	1.0450
[8, 6]	1.2088
[8, 8]	1.1406

Table 6: Computation time of different models (r=0, σ = 0.09)

Increasing the number of layers and the number of hidden nodes did not lead to more computation time. The main cause might be the earlier stopping.