

DSC 433/533 – Class 12 – Neural network examples for Tayko case

This class exercise is based on the **Tayko Software Reseller** Case (see separate document on the handouts page of the course website) and the Excel datasets **Tayko_allpart.xls** and **Tayko_part.xls** (available on the data page of the course website).

- The **Tayko_allpart.xls** has already been partitioned into Training (800), Validation (700), and Test (500) samples. Fit a neural network model to classify “purch.” In particular:
 - Select XLMiner > Classification > Neural Network (MultiLayer Feedforward).
 - Step 1: Move all the variables from “usa” to “res” to the “Input variables” box and “purch” to the “Output variable” box.
 - Step 2: Leave all options at their default values (i.e., select “Normalize input data,” set “# hidden layers” to 1, set “# nodes” to 25, set “# epochs” to 30, set “step size for gradient descent” to 0.1, set “weight change momentum” to 0.6, set “error tolerance” to 0.01, set “weight decay” to 0, select “squared error” for the cost function, set “standard” for the hidden layer sigmoid, and set “standard” for the output layer sigmoid).
 - Step 3:
 - Select “Summary report” for “Score training data.”
 - Select “Detailed report,” “Summary report,” and “Lift charts” for “Score validation data.”
 - De-select “Summary report” for “Score test data” (we’ll be using the test data later in the exercise).

We can use the results of the neural network analysis to estimate the probability that a customer will purchase based on the node connection weights in the network. The “error reports” on the Output worksheet summarize the results for the training and validation samples based on a 0.5 probability cut-off:

Training Error Report				Validation Error Report			
Class	# Cases	# Errors	% Error	Class	# Cases	# Errors	% Error
1	379	61	16.09	1	377	93	24.67
0	421	38	9.03	0	323	54	16.72
Overall	800	99	12.38	Overall	700	147	21.00

Another summary of model performance is the lift in the first decile, which is 1.80 in the validation sample.

- (i) Our next task is to see if models with different network architecture can outperform this first model. To simplify matters we will use just these two measures of model performance: “% validation error using a 0.5 cut-off” (21.00) and “validation lift in the first decile” (1.80). (A more sophisticated analysis might consider different costs for the two types of error: predicting purchase for non-purchasers and predicting non-purchase for purchasers.)
 - The model with 50 epochs (everything else the same) has 21.71% validation error and 1.80 validation lift in the first decile.
 - The model with 50 epochs and 2 hidden layers (everything else the same) has 20.29% validation error and 1.80 validation lift in the first decile.

The final model (50 epochs and 2 hidden layers) has the smallest validation error, and all three models have the same lift. The final model even beats the best logistic regression model from class 8, which had validation error 21.14% and lift 1.80.

- For now, for the purposes of illustration, we will digress from the Case and consider lift charts and profit calculations using the third model results and some alternate assumptions about costs and revenues. First, consider the “classification confusion matrix” for the validation data on the “CT_Output3” sheet:

Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	312	65
0	77	246

Expected net profits		
	Predicted Class	
Actual Class	1	0
1	1248	0
0	-154	0

1094

You can manually add a table for expected net profits based on it costing \$2 to send a catalog to a customer and Tayko receiving \$6.00 on average for each purchaser. For example, 312 customers sent a catalog purchased something and resulted in a profit of $312 \times (6 - 2) = \$1,248$. Conversely, 77 customers sent a catalog did not purchase anything and resulted in a loss of $77 \times 2 = \$154$. The other $65 + 246$ customers have a modeled probability of purchase less than 0.5 and so would not be sent a catalog. Net profit is therefore $1,248 - 154 = \$1,094$.

You can change the cut-off probability from 0.5 to some other number to see whether setting a different cut-off can lead to better profitability – in XLMiner it is the number in the blue cell just above the “classification confusion matrix” for the validation data (cell F71?). Complete the following table:

Cut-off probability	1	0.8	0.5	0.1535	0.05	0
Catalogs sent	0	221	389	463	500	700
Neural network model 3 profit	0	\$764	\$1,094	\$1,168	\$1,166	\$862

- The **Tayko_part.xls** has already been partitioned into Training (379), Validation (377), and Test (244) samples. Fit a neural network model to predict “spend.” In particular:
 - Select XLMiner > Prediction > Neural Network (MultiLayer Feedforward).
 - Step 1: Move all the variables from “usa” to “res” to the “Input variables” box and “spend” to the “Output variable” box.
 - Step 2: Select “Normalize input data” and leave all other options at their default values (i.e., set “# hidden layers” to 1, set “# nodes” to 25, set “# epochs” to 30, set “step size for gradient descent” to 0.1, set “weight change momentum” to 0.6, set “error tolerance” to 0.01, set “weight decay” to 0).
 - Step 3:
 - Select “Summary report” for “Score training data.”
 - Select “Detailed report,” “Summary report,” and “Lift charts” for “Score validation data.”
 - De-select “Summary report” for “Score test data” (we’ll be using the test data later in the exercise).
 Results are RMSE of 154.5 in the training sample, RMSE of 170.9 in the validation sample, and first decile lift of 2.61.
- Similar results for 50 epochs are RMSE of 150.2 in the training sample, RMSE of 169.3 in the validation sample, and first decile lift of 2.46.

Similar results for 50 epochs and 2 hidden layers are RMSE of 146.5 in the training sample, RMSE of 176.9 in the validation sample, and first decile lift of 2.42.

The first neural network model with 30 epochs seems to be the best of the three, but none are as good as the best multiple linear regression model from homework 3 which had RMSE of 163.0 in the validation sample, and first decile lift of 2.73.

- In summary, the best classification model for this case appears to be the neural network model with 50 epochs and 2 hidden layers, while the best prediction model appears to be the multiple linear regression model with 5 predictors (usa, s_p, freq, last, res). To assess the models’ fit on the test data:
 - Re-open the **Tayko_allpart.xls** spreadsheet.
 - Fit the neural network classification model and in step 3 select “detailed report” for “score test data.”
 - Inset a column on the left of worksheet “testdata” and copy/paste the “Prob. for 1 (success)” column from the NNC_TestScore1 worksheet.

- Re-open the **Tayko_part.xls** spreadsheet.
- Fit the multiple linear regression model and in step 2 select “Score new data” “In worksheet.”
 - select “Tayko_allpart.xls” for the Workbook and “testdata” for the Worksheet;
 - click on “Match variables with same names” and click OK.
- Inset another column on the left of worksheet “testdata” in **Tayko_allpart.xls** and copy/paste the “Predicted Value” column from the MLR_NewScore1 worksheet in **Tayko_part.xls**.
- Inset another column on the left of worksheet “testdata” in the **Tayko_allpart.xls** spreadsheet and calculate “expected spending” by multiplying the “Predicted Value” and “Prob. for 1 (success)” columns.
- Sort all the records in descending order of “expected spending.”
- Move the actual “spend” column to the left of the worksheet.
- Insert four more columns on the left:
 - calculate cumulative spending using the models as “=sum(E2:E\$2)” into column D;
 - insert cumulative integers from 1 to 500 into column C;
 - calculate cumulative random spending as “=D\$501*C2/C\$501” into column B;
 - calculate lift as column D divided by column B.
- Lift for the 18th of the 500 test records is 5.449565826, so total expected profit for sending catalogs to 180,000 names from the list of 5 million is $((5.449565826*(46951/500)*0.1065)-2)*180000 = ((9211/18)*0.1065)-2)*180000 = \9.45m .
- This compares with random mailing profit of $((46951/500)*0.1065)-2)*180000 = ((1690.23924/18)*0.1065)-2)*180000 = \1.44m .