Term Assignment

Vincent de Vos (0741795)
v.j.h.d.vos@student.tue.nl

Thom Hurks (0828691)
t.p.hurks@student.tue.nl

April 6, 2015

Solutions

Question 1: “The data set contain some missing values. Explain and motivate how you will deal with this issue.”

We inspected the unique values present in each column of the input dataset to verify if there are missing values and if the present values are sane. We found that each column can contain values that indicate missing data, as shown in the following table:

Value	**Description**
NaN	For numerical columns, no number is present
<undefined>	For categorical columns, no entry is present
“unknown”	Actual value in the data indicating that the value is unknown
-1	In the case of “nr of days since..”, -1 must indicate a missing value

The variables in the dataset have the following number of missing values:

Variable	**NaN**	**“unknown”**	**<undefined>**	**-1**	**Total**
age	87	0	0	0	87
job	0	288	61	0	349
marital	0	0	72	0	72
education	36	0	0	0	36
credit_card	95	0	0	0	95
balance	89	0	0		89
mortgage	80	0	0	0	80
loan	132	0	0	0	132
contact	0	13020	0	0	13020
day	149	0	0	0	149
month	0	0	31	0	31
duration	68	0	0	0	68

campaign	56	0	0	0	56
pdays	76	0	0	36887	36963
previous	62	0	0	0	62
poutcome	0	36933	34	0	36967
target	0	0	0	0	0

Since our dataset is really unbalanced, we will not be able to use a lot of data because we need to undersample the dataset. Therefore, if we remove entries with missing values then we lose too much data. Besides that, the fact that there is a missing value is in itself also information that can be used by an algorithm for predictions. Also, we don't even need to handle the NaN values, since MatLab already handles those in a special way and will ignore NaN targets for purposes of calculating performance and derivatives of performance.

Another option would be to replace all missing values by a single standardized "unknown" value, but the benefits of that are unclear; it might improve generalization since the data becomes more "general" in that aspect, but since the largest amount of missing values are in single cells in the input table and only regard single values within a certain column, the gain would likely be very low. For example, the 36933 "unknown" values in poutcome massively overshadow the 34 <undefined> values in the same row. Together with the 13020 "unknown" values in contact and the 36887 "-1" values in pdays they massively overshadow all other missing values in the entire table. The generalization gains would also be small because of this.

Because of these considerations, we have chosen to leave the missing values as-is; the fact that there is a missing value and which type of missing value is in itself also information that an algorithm can use, and with the data being imbalanced we need as much data as possible.

Question 2: "The data set is imbalanced (only roughly 11% of customers in this dataset subscribed to this product). Explain and motivate how you will deal with this issue."

Notice that we only have two classes: subscribed (11%) and unsubscribed (89%). A good approach would be to balance our dataset with undersampling. We will sample our balanced data set in such a way that we fully include all of our subscribed observations and let this make up for 30% of the dataset.

The remaining 70% will be containing unsubscribed data. This means we won't be using the full 89% of unsubscribed data but only a subset, this subset will be chosen randomly.

The 70/30 ratio was chosen in this case, because it is a tradeoff between two extremes; if we keep the 11/89 ratio then our data is too much skewed towards the unsubscribed outcome, and since we want to predict the subscribers that data shouldn't be too scarce because that can hurt the training algorithm. If we would balance the dataset to a 50/50 ratio, then we would have to throw lots of data away which can cause us to miss certain patterns. So the 70/30 ratio is a good tradeoff between the two scenarios.

Once we have our balanced data set we can split up this data into a training set, on which we will train our models, and a test set in order to determine the performance. When creating training and validation sets we will always use stratification to ensure that all those sets are representative of the balanced dataset. We also need a final test set, for which we will not use stratification, to make sure it has the same subscribed/unsubscribed ratio as the real data, because otherwise the final test performance would not be representative in reality. We will use a final test set of size 10% because we do not want to take too much data way from the training and validation sets, since the data is so imbalanced.

Our solution in short is to use uniform random sampling to select a 10% final testing set and then balance the other 90% of the data to a 70/30 ratio, where all the "subscribed" data that is left becomes the 30% of the training and validation set.

Question 3: "Some variables are categorical. Explain and motivate how you will deal with this issue."

A variable that has categorical values can be converted to vectors of variables where each variable has the value 0 or 1. This approach should only be used with (rule of thumb) < 20 different possible categorical values. For example, the variable Colour {"Red", "Green", "Blue"} can be converted to the variables Red_0_1, Green_0_1 and Blue_0_1, where Red_0_1 has the value 1 and Green_0_1 and Blue_0_1 are both 0 if Colour="Red".

With our number of different categorical values, this method is possible. No column has more than 20 different values, as can be seen in the following table:

Variable name	**Number of categorical values (excluding "unknown")**
contact	2
job	11
marital	3
poutcome	3
target	2

Question 4: “Explain and motivate what you will do with the date of the last contact (2 variables: date and month).”

Date of last contact is problematic because it is stored in two variables; day and month. Since these two variables together form a date, they are obviously strongly tied together and should not be fed into a classifier in a separated way. Unfortunately, we cannot convert the two to a date variable, since classifiers cannot just handle dates. Besides that, we do not have a year variable with which to create a full valid date. A solution would be to remove the month variable, and increase the day variable by the amount of days in the months that we removed. As a result we have one variable that tells us the number of days ago since last contact.

We also discussed the possibility of discretizing this variable by dividing it by 7 and rounding it up, which gives us a number then tells us roughly how many weeks have passed since last contact. This will reduce the number of possible values of this feature (from 365 initially, to 52) and probably yields better classification results because it is far more discriminative. However, the cardinality of the different values wasn't a big number, and therefore we choose to don't use this discretization in order to increase the performance of our model.

Question 5: “Which variables you think are relevant for the classification task? Motivate your answer.”

All variables are relevant for the classification task. If a variable is not, then the classification algorithm together with proper validation and testing will likely make sure that irrelevant variables get a very low or even 0 weight in the final classifier. Since we are not marketing or banking experts, it is not up to us to remove variables from the classification task beforehand based on our intuition; an intuition that could very well be wrong and hurt the classifier quality. Ideally we would consult a domain expert to advise us on the matter. We did inquire with the teachers to discuss the problem as well and heard that removing some variables can be beneficial, however as we are both computer science students choosing one or more variables to remove would be just a guess. A possible future solution to this problem would be to generate a permutation of variable subsets to use to train models and check which one has good performance with the best generalization (least amount of variables used) and pick that set of variables.

Question 6: “Is the data normalization required for this data set? Motivate your answer.”

Yes, data normalization is necessary, especially for neural networks. Not all variables have the same magnitude and not all classification algorithms can handle that. All variables should be normalized such that the magnitude of the variables is roughly the same. For fuzzy systems normalization is not necessary per se. We can use two versions of the dataset: one normalized and another not normalized.

It's good to note that for Neural Networks you can basically use any data as long as you are able to normalize it. However for Fuzzy systems this won't be useful if the data you're normalizing is categorized to begin with, because in that case you won't have an overlapping boundaries, meaning your prediction falls completely in one single membership function.

So, for neural networks we will normalize the dataset by making sure all numerical variables (columns) are within the range [0-1]. The categorical variables will be normalized according to the answer of question 3, by converting them to vectors where each dimension of the vector corresponds to one of the categorical values of the variable; we then place a single 1 somewhere in the vector te denote the category. The dataset does not have other variable types besides categorical or numerical, so those cases handle all data. However, we will use a specialised approach for the target variable; that one is only “yes” or “no” and has no missing values, so we can convert it directly to 0 or 1 (binary).

Question 7: “Prepare the data set, so that it is in the proper form for the neural network and fuzzy inference modeling. [Hint: Consider preparing 2 data sets, one for FIS and one for NN]”

We have done the data preparation in our MatLab script. The data preparation, including normalization and balancing, have been done as described in our previous answers. The MatLab script contains extensive documentation in the form of comments, so the script is very readable.

Question 8: “Explain and motivate how you will use the available data to create the model.”

For the neural network we took a simple but effective approach that allowed us to completely use all the columns within our data; all variables are normalized to numerical values in the range [0-1]. Categorical values are converted to numerical vectors where one of the dimensions is set to 1; the dimension corresponding to that category. If the category value is undefined for some entry, then the vector is zero-vector. NaN values are left as-is, as MatLab already handles those in a special way and will ignore NaN targets for purposes of calculating performance and derivatives of performance. This means that our entire dataset is normalized, numerical, and can contain some NaN values. This data is fed into the neural network after balancing; we use 10% of the original dataset (unbalanced) as the final test set and the remaining training/validation set is split up in a 95/5 ratio for training and validating. This ratio was found experimentally by checking which ratio resulted in the lowest MSE on the final test set. All this data is vertically concatenated and we use indices to inform the neural network in MatLab of what rows to use for what purpose. All variables are used, since we could use them all and did not want to remove variables as explained in our answer to question 1.

For the fuzzy inference system we have chosen to leave out the categorical data columns and used only a specific set of columns that are more suitable for modeling into a fuzzy inference system. Initially we wanted to use the normalized data set also used for our neural network, however since we used vectorization for that model (transforming column-values into vectors spanning multiple columns), we were unable to use these columns for the fuzzy inference system; this is because there is no way to define a rule that spans multiple columns instead of just one column. The result of this is that we are not able to use certain information from our original data, however fuzzy inference systems enable us to correctly model data that is subject to small nuances, which in our neural network would more likely result in model overfitting.

Question 9: “Explain and motivate how you will evaluate the performance of the models you will create.”

The performance is evaluated using our final test data as input for our trained models and then comparing the output from our model with the expected output. Since we've separated our final test set from our training data we know that these are completely independent. This allow us to ensure that the results from the models are not biased and therefore give us a good and realistic indication of the actual results we could expect from real data.

The neural network performance is evaluated by checking the mean squared error. A total of 10 neural networks are generated. Before each network is trained, we create a new balanced dataset from the normalized data. Since randomness is used during the data balancing and undersampling, this ensures the neural networks are each trained on different selections of the dataset. The neural network with the lowest MSE is then selected as the final model. The neural network with the lowest MSE has a MSE of 0.2346

The evaluation for the FIS model is done by measuring the mean-squared value. This is the sum of the squared difference between the value of the actual label in our data and the actual prediction made, basicly a zero-one loss function. This gives us an average on the error for our model. A larger error value indicates that the performance of our model is not performing that good, which allowed us to fine-tune the model by comparing the performance of one model to another.

Together, the best model can be found by picking the model with the lowest MSE between the neural network and the fuzzy inference system.

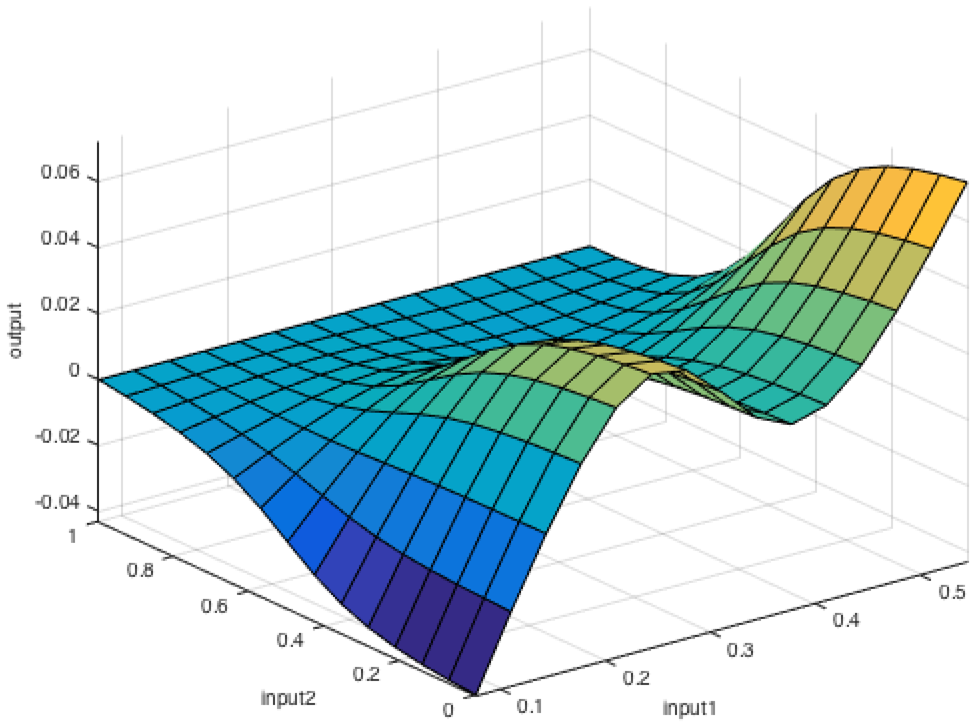
Question 10: “Build the fuzzy inference system model. Which variables are you using? What parameters are you using? Explain, why you decided to choose those values. What is the quality of your model? [Hint: to keep the model smaller you use only one of the categorical variables that has 3 or more values (indicate which variable did you choose)]”

In order to get the most out of our Fuzzy Inference Model we've decided to to leave out the categorical data columns and focus only on a specific set of columns that are more suitable for modeling into a fuzzy inference system.

After some data analysis: inspecting the unique values of columns and the ranges of the column values, we picked variables that modeled various aspect within the given data-set

- age: demographic information of the actual account holder
- education: an indicator on the level of education. This is more useful than the job name which cannot be measured/compared.
- balance: gives some insight on spending behaviour: saver/spender
- mortgage: tells us something about the home situation (bought a house or renting)
- loan: the personal loan of an account holder
- number of days since last contact: this models any actual interaction patterns between the bank and customer and also implies values from the "contact" column.
- campaign: this is the categorical variable we decided to keep in our system. This is possibly the only piece of information that allows us to steer on data specifically for the account holder and on results from a previous campaign.

These variables give us a solid profile of an account holder. It's good to note that by leaving out categorical data we certainly mis (possible valuable) information from our original data. However our fuzzy inference model will enable us to correctly model data that is subject to small nuances. A visualization of the fuzzy inference system can be seen in figure 1.



1) Visualization of the fuzzy inference system.

Question 11: “Build a neural network model. Which variables are you using? What parameters are you using? Explain, why you decided to choose those values. What is the quality of your model?”

As has also been explained in the previous questions, we are using all variables for the neural network. The reasons are that we can use all variables here easily, because all categorical variables are converted to numerical vectors containing a single 1 value and all data is normalized. The other reason is that we are unsure of which variables to remove, although in the future we could test this by leaving out various variables using a script and checking how the results and the MSE change. One change of note is that the day and month variables were merged to a single variable. This variable is the number of days since the beginning of the year. This variable is then discretized and normalized. Some experiments were done with the type of discretization, and it was discovered that discretizing to two-week periods resulted in the lowest MSE. Both leaving the precision of the variable to days or reducing it to entire months resulted in much higher MSE than the two-week variant.

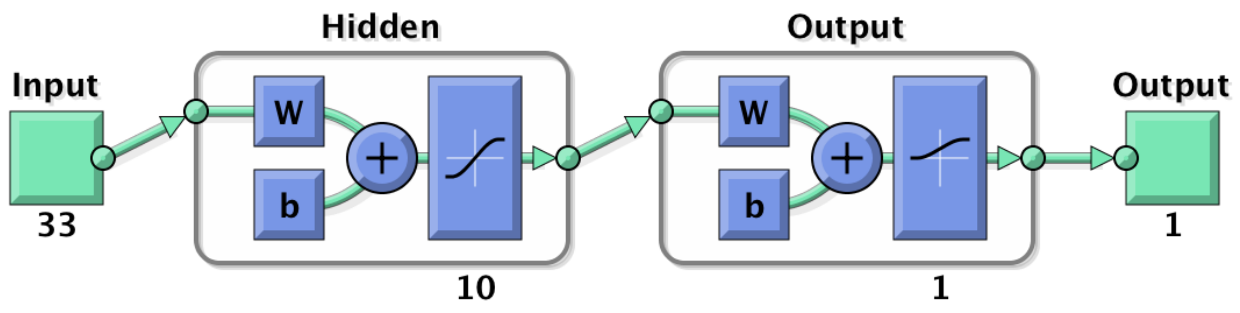
The neural network parameters used are a patternnet with one hidden layer containing 10 neurons. This was chosen because pattern nets are the best nets for classification tasks according to MatLab. Also, the neuron count of 10 it is the default and recommended value in MatLab. Also, experimentally changing the value or adding layers increases the MSE value.

As training function, “trainlm” (Levenberg-Marquardt) was used. The reason is because MatLab claims it is the best overall training function they have, although the downside is that it is computationally more expensive.

Other parameters are the balancing of the data, which has been done using 10% of the original data (unbalanced) as the final test set. Of the remaining part, 95% is used as the training set and 5% as the validation set. This ratio was picked because it experimentally led to the best (lowest) minimum MSE and the lowest average MSE between all trained networks.

The end result of training 10 different neural networks, using random sampling of the dataset before each training session, is that the best neural network has a MSE of 0.2346. All the other neural networks from that batch have a very close MSE, with the second-worst neural network having a MSE of 0.2509. This means that the chosen parameters are good in general for the neural network.

A different method of training the neural networks is using trainscg (scaled conjugate gradient backpropagation) training function, which uses the cross-entropy performance measure by default. This special performance measure means that it is harder to compare to other models, but regardless it is interesting to note that our best (lowest) cross-entropy value was 0.0920. We are no experts in performance-measures of neural networks, but some research using the internet showed that some experts claim that cross-entropy is a better performance measure for neural network classifiers than MSE. Unfortunately we cannot compare the value at this moment. An overview of the neural network can be seen in image 2. In total there are 33 input variables, 10 neurons in the hidden layers and one output neuron.



2) Overview of the trained neural network.

Question 12: “Provide a comparison of the fuzzy set and neural network models. Which model would you recommend to the B-BANK? Motivate your answer.”

Our fuzzy inference system is very crude, so we are not able to make a good comparison right now, unless we improve our fuzzy inference system generation code and write more code for getting error metrics. Right now, we would definitely recommend our neural network to B-BANK. The mean-squared error of 0.2346 is a low value, since we saw neural networks with much higher MSE during experiments. The cross-entropy error value was also quite low. However, we can only really confirm our neural network is good when we also create a fuzzy inference model, so it is quite unfortunate that we did not manage that yet.