Assignment 1 - Language Development in ASD - part 3

Riccardo Fusaroli

August 10, 2018

## Welcome to the third exciting part of the Language Development in ASD exercise

In this exercise we will delve more in depth with different practices of model comparison and model selection, by first evaluating your models from last time, then learning how to cross-validate models and finally how to systematically compare models.

N.B. There are several datasets for this exercise, so pay attention to which one you are using!

1. The (training) dataset from last time (the awesome one you produced :-) ).
2. The (test) datasets on which you can test the models from last time:

* Demographic and clinical data: <https://www.dropbox.com/s/ra99bdvm6fzay3g/demo_test.csv?dl=1>
* Utterance Length data: <https://www.dropbox.com/s/uxtqqzl18nwxowq/LU_test.csv?dl=1>
* Word data: <https://www.dropbox.com/s/1ces4hv8kh0stov/token_test.csv?dl=1>

### Exercise 1) Testing model performance

How did your models from last time perform? In this exercise you have to compare the results on the training data () and on the test data. Report both of them. Compare them. Discuss why they are different.

Two two models from last time are the linear and the quadratic model with an interaction effect between diagnosis and visit. We most simple model is the linear model wihtout an interaction.

qua\_model = lmer(CHI\_MLU ~ Diagnosis \* VISIT + I(VISIT^2) + (1+VISIT+ I(VISIT^2)|SUBJ)

lin\_model = lmer(CHI\_MLU ~ Diagnosis \* VISIT + (1+VISIT |SUBJ))

lin\_model\_simple = lmer(CHI\_MLU ~ Diagnosis + VISIT + (1+VISIT |SUBJ))

* recreate the models you chose last time (just write the model code again and apply it to your training data (from the first assignment))
* calculate performance of the model on the training data: root mean square error is a good measure. (Tip: google the function rmse())
* create the test dataset (apply the code from assignment 1 part 1 to clean up the 3 test datasets)
* test the performance of the models on the test data (Tips: google the functions "predict()")
* optional: predictions are never certain, can you identify the uncertainty of the predictions? (e.g. google predictinterval())

formatting tip: If you write code in this document and plan to hand it in, remember to put include=FALSE in the code chunks before handing in.

We use the root mean square error to calculate the performance of the model.

The performance of the quadratic model on the training data is 0.29. The performance on the test data is 1.01. The difference indicates that our model is overfitted, as it performs farely well on the training set, but fails to predict the test data. This shows that the model is not generalisable.

### Exercise 2) Model Selection via Cross-validation (N.B: ChildMLU!)

One way to reduce bad surprises when testing a model on new data is to train the model via cross-validation.

In this exercise you have to use cross-validation to calculate the predictive error of your models and use this predictive error to select the best possible model.

* Use cross-validation to compare your model from last week with the basic model (Child MLU as a function of Time and Diagnosis, and don't forget the random effects!)
* (Tips): google the function "createFolds"; loop through each fold, train both models on the other folds and test them on the fold)

Which model is better at predicting new data: the one you selected last week or the one chosen via cross-validation this week?

* Test both of them on the test data.
* Report the results and comment on them.
* Now try to find the best possible predictive model of ChildMLU, that is, the one that produces the best cross-validated results.
* Bonus Question 1: What is the effect of changing the number of folds? Can you plot RMSE as a function of number of folds?
* Bonus Question 2: compare the cross-validated predictive error against the actual predictive error on the test data

#cross validation --> so we do not have to go "throw" out data in order to test on it   
#make a cake and split it in 5 parts --> 5 folds   
#use cross validation to find the best model without overfitting  
#put the kids in a random fold  
  
  
  
  
folds <- createFolds(unique(train\_data$SUBJ), k = 5, list = TRUE) #create folds. Unique() puts each unique kid in a fold, so all data from one subj goes into one fold  
  
folds

## $Fold1  
## [1] 1 8 15 22 24 30 32 39 45 46 55 57  
##   
## $Fold2  
## [1] 3 7 14 21 23 25 31 36 38 47 51 56  
##   
## $Fold3  
## [1] 6 9 11 19 26 29 33 40 44 48 50 59  
##   
## $Fold4  
## [1] 4 5 13 16 17 20 34 37 42 52 54 60  
##   
## $Fold5  
## [1] 2 10 12 18 27 28 35 41 43 49 53 58

#now we need to create a loop, important to run through the empty lists before running   
  
  
rmse\_train\_qua = NULL  
  
rmse\_test\_qua = NULL  
  
rmse\_test\_lin = NULL  
rmse\_train\_lin = NULL  
   
  
temp\_test = NULL  
  
temp\_train = NULL  
  
  
  
for (f in folds) {  
 #divide into test and train  
 temp\_test = filter(train\_data, SUBJ %in% f)  
 temp\_train = filter(train\_data, ! SUBJ %in% f)  
   
 #run models on train  
 lin\_model <- lmer(CHI\_MLU ~ Diagnosis \* VISIT + (1+VISIT|SUBJ), data= temp\_train, REML=FALSE)  
 qua\_model <- lmer(CHI\_MLU ~ Diagnosis \* VISIT + I(VISIT^2) + (1+VISIT+ I(VISIT^2)|SUBJ), data=temp\_train, REML=FALSE)  
   
 #predict from the two models on both train and test data, train data does not really make sense --> just pedagogic   
 lin\_predict\_test <- predict(lin\_model, temp\_test, allow.new.levels = TRUE)  
 qua\_predict\_test <- predict(qua\_model, temp\_test, allow.new.levels = TRUE)  
 lin\_predict\_train <- predict(lin\_model, temp\_train, allow.new.levels = TRUE)  
 qua\_predict\_train <- predict(qua\_model, temp\_train, allow.new.levels = TRUE)  
   
   
 #get average error and put into list  
 rmse\_test\_lin <- c(rmse\_test\_lin, rmse(temp\_test$CHI\_MLU, lin\_predict\_test))  
 rmse\_train\_lin <- c(rmse\_train\_lin, rmse(temp\_train$CHI\_MLU, lin\_predict\_train))  
 rmse\_test\_qua <- c( rmse\_test\_qua, rmse(temp\_test$CHI\_MLU, qua\_predict\_test))  
 rmse\_train\_qua <- c(rmse\_train\_qua, rmse(temp\_train$CHI\_MLU, qua\_predict\_train))  
   
   
}  
  
#rmse, errors for the linear model on the test data set  
 rmse\_test\_lin

## [1] 0.9489862 0.7996476 0.6623629 0.5878465 0.6407821

rmse\_train\_lin

## [1] 0.3578674 0.3260652 0.3443568 0.3549591 0.3548682

rmse\_test\_qua

## [1] 0.9187468 0.7888001 0.6546901 0.6047215 0.6531303

rmse\_train\_qua

## [1] 0.2916734 0.2482526 0.2949918 0.2976566 0.2980160

mean(rmse\_test\_lin) #get the mean error across the folds for the linear model

## [1] 0.727925

mean(rmse\_test\_qua) #get the mean error across the folds for the quadratic model

## [1] 0.7240178

mean(rmse\_train\_lin)

## [1] 0.3476233

mean(rmse\_train\_qua)

## [1] 0.2861181

From cross-validation with 5-folds on the quadratic and the linear model, we found that the quadratic model performed best (average rmse = 0.72), when predicting CHI-MLU. However the linear model had an average performance at 0.78 and thereby the difference between the to is rather small. The two models are therefore almost equally good.

One thing to notice is that when we use the models to predict from the training data (inside the fold), both models perform much better, linear rsme = 0.35, quadratic rsme = 0.28. This again is an indicator of the models overfitting.

In order to test both models on the test data, we first train the models on the training data.Thereafter predict the test data, from the trained models. The quadratic model was able to predict the test data with an average error (rmse) of 1.009. The linear model predict the test data with an average error (rmse) of 1.007.

Both models performs poorly in predicting the test data. After the cross-validation we found that the two models were almost equally good and it therefore makes sense, that their predictive abilities are similar. As mentioned earlier, the models perform well in predicting the training data. Thus, this poor performance on the test data is an indocator of the models being overfitted. One way to overcome this obstacle would be to sample more data to train the models on.

* Now try to find the best possible predictive model of ChildMLU, that is, the one that produces the best cross-validated results.
* Bonus Question 1: What is the effect of changing the number of folds? Can you plot RMSE as a function of number of folds?
* Bonus Question 2: compare the cross-validated predictive error against the actual predictive error on the test data

rmse\_test\_token <- NULL  
rmse\_test\_token\_verbal <- NULL  
rmse\_test\_ados <- NULL  
rmse\_test\_ados\_verbal <- NULL  
  
temp\_test = NULL  
  
temp\_train = NULL  
  
  
  
for (f in folds) {  
 #divide into test and train  
 temp\_test = filter(train\_data, SUBJ %in% f)  
 temp\_train = filter(train\_data, ! SUBJ %in% f)  
   
 #run models on train  
 token\_model <- lmer(CHI\_MLU ~ Diagnosis \* VISIT + tokens\_CHI + (1+VISIT|SUBJ), data= temp\_train, REML=FALSE)  
 token\_verbal\_model <- lmer(CHI\_MLU ~ Diagnosis \* VISIT + tokens\_CHI + verbalIQ + (1+VISIT|SUBJ), data= temp\_train, REML=FALSE)  
 ados\_model <- lmer(CHI\_MLU ~ Diagnosis \* VISIT + ADOS + (1+VISIT|SUBJ), data= temp\_train, REML=FALSE)  
 ados\_verbal\_model <- lmer(CHI\_MLU ~ Diagnosis \* VISIT + ADOS + verbalIQ + (1+VISIT|SUBJ), data= temp\_train, REML=FALSE)  
  
 #predict from the two models on test data  
 predict\_token\_model <- predict(token\_model, temp\_test, allow.new.levels = TRUE)  
 predict\_ados\_model <- predict(ados\_model, temp\_test, allow.new.levels = TRUE)  
 predict\_ados\_verbal\_model <- predict(ados\_verbal\_model, temp\_test, allow.new.levels = TRUE)  
 predict\_token\_verbal\_model <- predict(token\_verbal\_model, temp\_test, allow.new.levels = TRUE)  
  
   
 #get average error and put into list  
 rmse\_test\_token <- c(rmse\_test\_token, rmse(temp\_test$CHI\_MLU, predict\_token\_model))  
 rmse\_test\_token\_verbal <- c( rmse\_test\_token\_verbal, rmse(temp\_test$CHI\_MLU, predict\_token\_verbal\_model))  
 rmse\_test\_ados <- c(rmse\_test\_ados, rmse(temp\_test$CHI\_MLU, predict\_ados\_model))  
 rmse\_test\_ados\_verbal <- c( rmse\_test\_ados\_verbal, rmse(temp\_test$CHI\_MLU, predict\_ados\_verbal\_model))  
  
  
   
}  
  
  
mean(rmse\_test\_token)

## [1] 0.5007117

mean(rmse\_test\_token\_verbal)

## [1] 0.4797874

mean(rmse\_test\_ados)

## [1] 0.6699631

mean(rmse\_test\_ados\_verbal)

## [1] 0.5838796

We have cross-validated the following models:

token\_model <- lmer(CHI\_MLU ~ Diagnosis \* VISIT + tokens\_CHI + (1+VISIT|SUBJ), data= temp\_train, REML=FALSE) token\_verbal\_model <- lmer(CHI\_MLU ~ Diagnosis \* VISIT + tokens\_CHI + verbalIQ + (1+VISIT|SUBJ), data= temp\_train, REML=FALSE) ados\_model <- lmer(CHI\_MLU ~ Diagnosis \* VISIT + ADOS + (1+VISIT|SUBJ), data= temp\_train, REML=FALSE) ados\_verbal\_model <- lmer(CHI\_MLU ~ Diagnosis \* VISIT + ADOS + verbalIQ + (1+VISIT|SUBJ), data= temp\_train, REML=FALSE)

The model that produces the best cross-validated results are the one with verbal IQ and tokens-CHI (number of unique words) as fixed effects together with diagnosis and visit (rmse = 0.48). The reason for this is the high correlation between tokens\_CHI and child MLU as they measure almost the same thing. There is a leakage in the model. The model which takes the severety of autism (ADOS) and verbal IQ at the first visit into account together with diagnosis and visit as fixed effects is the second best model. It makes more sense to have a model which uses values obtained at the first visit, as we often want to predict what will happen at the following five visits.

We can try to train the model which includes ados as fixed effects on the training data and then see how well it predicts the test data.

ados\_verbal\_model <- lmer(CHI\_MLU ~ Diagnosis \* VISIT + ADOS + verbalIQ + (1+VISIT|SUBJ), data= train\_data, REML=FALSE)  
  
predictions\_ados\_verbal <- predict(ados\_verbal\_model, test\_data, allow.new.levels = TRUE)  
  
  
  
rmse\_ados\_verbal <-rmse(test\_data$CHI\_MLU, predictions\_ados\_verbal)  
  
rmse\_ados\_verbal

## [1] 0.832853

ados\_model <- lmer(CHI\_MLU ~ Diagnosis \* VISIT + ADOS + (1+VISIT|SUBJ), data= train\_data, REML=FALSE) #better without verbal iq on test data   
  
predictions\_ados <- predict(ados\_model, test\_data, allow.new.levels = TRUE)  
  
  
  
rmse\_ados <-rmse(test\_data$CHI\_MLU, predictions\_ados)  
  
rmse\_ados

## [1] 0.720644

From training the models and then predicting the test data, we see that the model which only includes ados together with diagnosis and visit (and not verbal IQ) predict the test data better than the model which also uses verbal IQ. This clearly shows that the model is overfitting, when more predicters is added to the model.

### Exercise 3) Assessing the single child

Let's get to business. This new kiddo - Bernie - has entered your clinic. This child has to be assessed according to his group's average and his expected development.

Bernie is one of the six kids in the test dataset, so make sure to extract that child alone for the following analysis.

You want to evaluate:

* how does the child fare in ChildMLU compared to the average TD child at each visit? Define the distance in terms of absolute difference between this Child and the average TD. (Tip: recreate the equation of the model: Y=Intercept+BetaX1+BetaX2, etc; input the average of the TD group for each parameter in the model as X1, X2, etc.).
* how does the child fare compared to the model predictions at Visit 6? Is the child below or above expectations? (tip: use the predict() function on Bernie's data only and compare the prediction with the actual performance of the child)

We want to look at the difference between Bernie and the avearge TD child at the 6 visits. We will use the quadratic model for this.

First we run the quadratic model and look at the summary. From there we can extract the estimates for the fixed effects to recreate the model.

Thereby we have the following equation:

CHI\_MLU = 0.98 + Diagnosis*-0.36 + VISIT*0.41 + -0.04*VISIT^2 + VISIT*Diagnosis\*0.25

Diagnosis is either 0 or 1. ASD = 0 and TD = 1. As the estimate describe the change when we move from ASD to TD (A is first in the alphabet).

We then set TD = 1, and put in from 1 to 6 in visits. From that we get the average MLU of TD's for each visit. These values are (1.24, 1.78, 2.24, 2.62, 2.92, 3.14).

After this we subtract Bernies values from the avearge TD values and find the absolute difference. These are: 0.7444560 0.7644444 1.1131915 0.5630986 0.2532523 0.3084127.

The quadratic model would predict Bernie to have an MLU of 1.88 at the sixth visit. However the measured MLU for Bernie was 3.45, thereby we can see that above average from what would be expected of children with autism. The difference between the predicted and the actual value of mean length of utterance is 1.57. If the look at ADOS for Bernie, which we can see that his autism is not very severe. This could explain why he overperforms in MLU compared to what the model would predict, as it does not take into account that autism varies on a spectum.

### OPTIONAL: Exercise 4) Model Selection via Information Criteria

Another way to reduce the bad surprises when testing a model on new data is to pay close attention to the relative information criteria between the models you are comparing. Let's learn how to do that!

Re-create a selection of possible models explaining ChildMLU (the ones you tested for exercise 2, but now trained on the full dataset and not cross-validated).

Then try to find the best possible predictive model of ChildMLU, that is, the one that produces the lowest information criterion.

* Bonus question for the optional exercise: are information criteria correlated with cross-validated RMSE? That is, if you take AIC for Model 1, Model 2 and Model 3, do they co-vary with their cross-validated RMSE?

### OPTIONAL: Exercise 5): Using Lasso for model selection

Welcome to the last secret exercise. If you have already solved the previous exercises, and still there's not enough for you, you can expand your expertise by learning about penalizations. Check out this tutorial: <http://machinelearningmastery.com/penalized-regression-in-r/> and make sure to google what penalization is, with a focus on L1 and L2-norms. Then try them on your data!