STA2104 Assignment 3

Matthew Scicluna

998367342

**Summary**

We compared a linear model to two Gaussian process models (one with linear covariance and one without) using data from the 1990 US census. Our goal was to analyze the differences in these models. We also analyzed a paper on predicting Asperger’s syndrome using some of the machine learning techniques learned in the STA2104 course.

**Models**

Linear Regression – We first considered the ordinary least squares regression model

Where is the best linear unbiased estimator of . We found that this model trained very quickly and had very good accuracy (MSE of test data was 0.2876) and run time (0.16s).

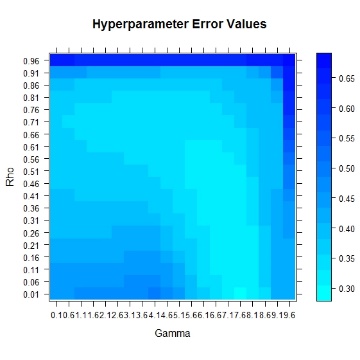
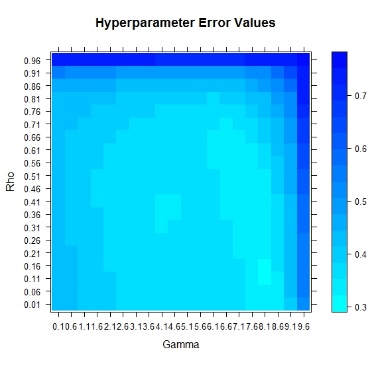
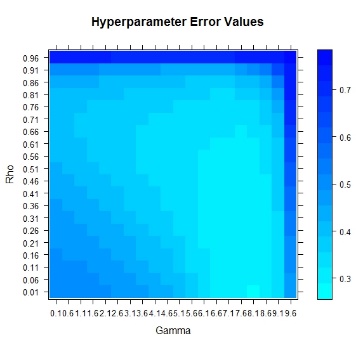
Gaussian Process Model with Linear Covariance – Next we considered a Gaussian Process model with a simple linear covariance. The Kernel function was given by:

We found that this model performed similarly to the ordinary least squares approximation, which was expected since it was equivalent to it. The higher error in this model seems to be the result of the Covariance matrix being ill conditioned. This can been seen from its high condition number, which was of the order of 109. We also noticed that without the beta term the matrix was singular. Numerical instability seems to have led to the inverse of the covariance matrix to be erroneously non-symmetric and thus leading to the increase in mean squared error observed.

Gaussian Process Model – Next we considered a more flexible Gaussian Process model with the following kernel function.

)

The hyper parameters g and ρ were computed using a simple grid search with 5 fold cross validation, and the predictions were averaged over the 5 resultant models with equal weighting. It can be seen by figure one that each validation set induced its own error surface over the space of hyper-parameters, which 5 different minima, and so each of the 5 models were different.

C:\Users\Matthew Scicluna\AppData\Local\Microsoft\Windows\INetCache\Content.Word\E1.tiff

*Figure 1: The error surface over the space of hyper-parameters. Areas in light blue have the lowest error. Note only 4 of the 5 Error surfaces are shown here.*

Rescaled Covariates

Using the same Kernel as before, we rescaled covariates 1 and 7 to make them of the same magnitude as the other covariates, since the scale parameter ρ is expected to work better with covariates that are of the same magnitude (since it is a scale parameter), using the same model as before.

The optimal g and ρ for each of the five cross validated models with and without the rescaled covariates is provided here:

|  |  |  |
| --- | --- | --- |
| With Rescaling | | |
| g | ρ | Error |
| 9.6 | 0.91 | 0.3258 |
| 4.6 | 0.96 | 0.2810 |
| 8.6 | 0.96 | 0.3595 |
| 2.1 | 0.96 | 0.3386 |
| 9.1 | 0.21 | 0.2691 |

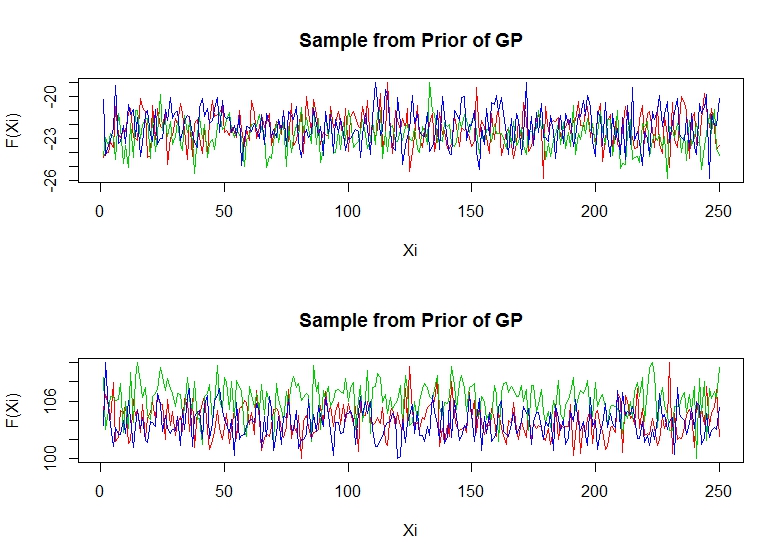
|  |  |
| --- | --- |
| Without Rescaling | |
| g | ρ |
| 9.6 | 0.11 |
| 9.6 | 0.16 |
| 8.1 | 0.16 |
| 9.6 | 0.21 |
| 9.6 | 0.16 |

*Figure 2: Comparison of minimum-error hyper-parameters selected from 5 fold cross validation*

The value of ρ increased relative to the unscaled model, indicating that the rescaled covariates became more relevant in predicting the response variable relative to when they were not scaled.

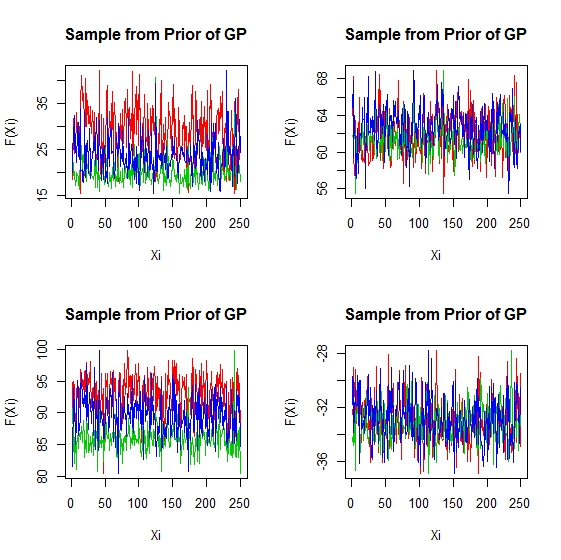
To visualize the change in the model parameters after rescaling, we plotted three samples from the prior distribution over the Gaussian Process (i.e. N250(0,Σ), where [Σ]ij = K(xi, xj) ) for a model without rescaling and for one with rescaling. This can be seen in figure 3. It is worth noting that the GP sampled from in the top graph of figure 3 has g = 9.6 and ρ = 0.11, while the GP sampled from in the bottom graph of figure 3 has g = 2.1 and ρ = 0.96. We can see that the increased value of g causes the GP to have decreased in scale, compared to the bottom of the graph, as evidenced by the change in the upper and lower values on the axis.

Furthermore, when looking at the variance in the samples from each models prior, it can be seen that the rescaled covariates cause an increase in ρ which further caused its induced prior to have higher variance when compared priors with smaller ρ. This makes sense because the larger value of ρallowed for more flexible models and thus higher variability in possible models.

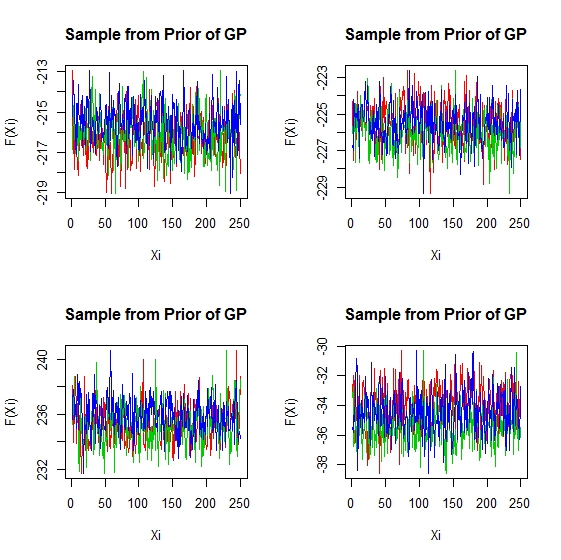


*Figure 3: Some samples from each prior distribution over GPs with (bottom) and without (top) rescaling*

For the purposes of completeness we also include plots of 3 samples drawn from the prior distributions of GPs using scaled and unscaled covariates. These can be seen in Figure 4 and 5. Notice that the same general patterns can be seen when comparing the samples of priors in figure 4 to 5 as when comparing the samples of the priors in figure 3.



*Figure 4: Samples drawn from the prior distribution over GPs using scaled covariates.*



*Figure 5: Samples drawn from prior distribution over GPs with unscaled covariates.*

**Performance Evaluation**

|  |  |  |  |
| --- | --- | --- | --- |
|  | OLS Regression | Linear Gaussian Process | Nonlinear Gaussian Process |
| MSE of Test Set | 0.2876 | 0.4435 | 0.2895 |
| MSE of Test Set after Rescaling | NA | NA | 0.241 |
| Time to Compute (Seconds) | 0.16 | 7.83 | 45.52 |

While the nonlinear GP had the best MSE (once the covariates were rescaled), it also took the longest in terms of computation time. It took almost 6 times as long as the Linear GP, since it had to average over 5 separate model predictions. Notice that this does not include the time it took to do the grid search to find the optimal parameters. It is interesting to note that the average model of the Nonlinear GP after rescaling did far better than any of the individual models it was averaged from. These ones performed comparably to the OLS regression model and the unscaled Nonlinear GP model, suggesting that the averaged model was able to fit some of the individual cases that the other models had trouble handling.

**Appendix**

Below is the code used to complete this assignment. It is written in the R programming language.

rm(list=ls())

#options(digits=5)

##############################

####Read the Training Data####

##############################

URL <- url("http://www.cs.toronto.edu/~rsalakhu/STA414\_2015/train1x")

raw <- matrix(scan(URL),,8,byrow=TRUE)

TrainX <- data.frame(House = raw[,1],

Asian = raw[,2],

Age = raw[,3],

Persons = raw[,4],

Minority = raw[,5],

Sale = raw[,6],

Rooms = raw[,7],

Vacant6Mo = raw[,8])

URL <- url("http://www.cs.toronto.edu/~rsalakhu/STA414\_2015/train1y")

raw <- matrix(scan(URL),,1,byrow=TRUE)

TrainY <- data.frame(Price = raw[,1])

##########################

####Read the Test Data####

##########################

URL <- url("http://www.cs.toronto.edu/~rsalakhu/STA414\_2015/testx")

raw <- matrix(scan(URL),,8,byrow=TRUE)

TestX <- data.frame(House = raw[,1],

Asian = raw[,2],

Age = raw[,3],

Persons = raw[,4],

Minority = raw[,5],

Sale = raw[,6],

Rooms = raw[,7],

Vacant6Mo = raw[,8])

URL <- url("http://www.cs.toronto.edu/~rsalakhu/STA414\_2015/testy")

raw <- matrix(scan(URL),,1,byrow=TRUE)

TestY <- data.frame(Price = raw[,1])

##############################

####Fitting a Linear Model####

##############################

ptm <- proc.time()

RegModel <- function(XR, YR) #Function computes Regression model and finds error for 2 dataframes

{

X <- as.matrix(XR)

Intercept <- numeric(nrow(XR))+1

X <- cbind(Intercept, X)

Y <- as.matrix(YR)

LSWeights <- solve(t(X)%\*%X)%\*%t(X)%\*%Y #Computing OLS

Predictions <- X%\*%LSWeights

SError <- round(mean((Predictions-Y)^2),4) #Computing MSE

print(paste("The Mean squared error of the model is", SError))

return(list(MSE=SError, Weights=LSWeights)) #Output is a list

}

LinModel <- RegModel(TrainX,TrainY) #make the linear model

#Check MSE on test cases

X <- as.matrix(TestX)

Intercept <- numeric(nrow(X))+1

X <- cbind(Intercept, X)

Predictions <- X%\*%LinModel$Weights

Y <- as.matrix(TestY)

SError <- round(mean((Predictions-Y)^2),4) #Computing MSE

print(paste("The Mean squared error of the model on the testing data is", SError))

proc.time() - ptm #Ellapsed time for Regression

#So MSE of training data is 0.2802 and MSE of test data is 0.2876.

###############################################################

####Fitting a Gaussian Process Model With Linear Covariance####

###############################################################

ptm <- proc.time()

#Set Seed

set.seed(2015)

#Initialize Data

X <- as.matrix(TrainX)

Y <- as.matrix(TrainY)

XT <- as.matrix(TestX)

YT <- as.matrix(TestY)

#Initialize Hyperparameters

Alpha = 10000

Beta = 1

#Compute Kernel

Kernel<-function(x1, x2, Alpha)

{Alpha\*t(x1)%\*%x2}

#Compute Gram Matrix column

GramColumn <- function(j)

{apply(X,1,Kernel,X[j,],10000)}

#Compute C inverse

C <- sapply(as.matrix(seq(1,250)),GramColumn)

C <- C + Beta\*diag(nrow(X))

CInv <- solve(C)

#Predict from XN

GPPred <- function(XN)

{K <- apply(X,1,Kernel,XN,10000)

c <- Kernel(XN,XN,10000) + Beta

Mu <- t(K)%\*%CInv%\*%Y

St.dev <- c - t(K)%\*%CInv%\*%K

#Can sample from normal using

#rnorm(1,Mu,St.dev)

return(Mu)}

Predtt <- apply(as.matrix(XT), 1, GPPred)

SError <- round(mean((Predtt-YT)^2),4) #Computing MSE

print(paste("The Mean squared error of the linear GP model on the testing data is", SError))

#Test Error is 0.4435

proc.time() - ptm #time elapsed

#Visualize the GP

library("MASS")

PriorSample<-mvrnorm(3, rep(0,250), CInv)

plot(PriorSample[1,],type="l",col=2, xlab="Xi", ylab="F(Xi)", main="Sample from Prior of Gaussian Process")

par(new=T)

plot(PriorSample[2,],type="l",axes=F,col=3, xlab="", ylab="")

par(new=T)

plot(PriorSample[3,],type="l",axes=F,col=4, xlab="", ylab="")

par(new=F)

##################################################################

####Fitting a Gaussian Process Model Without Linear Covariance####

##################################################################

#Add hyperparameters

Theta <- 10000

GPPredictor<- function(Gamma,Rho,tX, tY, tXT, tYT)

{#Change Kernel

Kernel2<-function(x1, x2, Gamma, Rho)

{Theta + (Gamma^2)\*exp(-(Rho^2)\*sum((x1-x2)^2))}

#Compute Gram Matrix column

GramColumn2 <- function(j)

{apply(tX,1,Kernel2,tX[j,], Gamma, Rho)}

#Compute C inverse

C <- sapply(as.matrix(seq(1,nrow(tX))),GramColumn2)

C <- C + Beta\*diag(nrow(tX))

CInv <- solve(C)

#Predict from XN

GPPred2 <- function(tXN)

{K <- apply( tX, 1, Kernel2, tXN, Gamma, Rho)

c <- Kernel2(tXN,tXN, Gamma, Rho) + Beta

Mu <- t(K)%\*%CInv%\*%tY

St.dev <- c - t(K)%\*%CInv%\*%K

#Can sample from normal using

#rnorm(1,Mu,St.dev)

return(Mu)}

Predss <- apply(as.matrix(tXT), 1, GPPred2)

return(Predss)}

GPFit2 <- function(Gamma,Rho,tX, tY, tXT, tYT)

{Predictions2 <- GPPredictor(Gamma,Rho,tX, tY, tXT, tYT)

SError <- round(mean((Predictions2-tYT)^2),4) #Computing MSE

#print(paste("The Mean squared error of the GP model on the testing data is", SError))

return(list(Gamma=Gamma,Rho=Rho,Error=SError))

}

#Minimize parameters using grid search.

GridSearch <- function( ttX, ttY, ttXT, ttYT)

{

Errors <- matrix(,,3)

colnames(Errors)<-c("Gamma","Rho","Error")

Errors <- Errors[-1,]

for (Gamma in seq(0.1,10,0.5))

{for (Rho in seq(0.01,1,0.05))

{Fit <- GPFit2(Gamma,Rho, ttX, ttY, ttXT, ttYT)

Errors <- rbind(c(Fit$Gamma,Fit$Rho,Fit$Error),Errors)

}}

#Get Optimal Error

MinParams <- Errors[Errors[,3]==min(Errors[,3]),]

return(list(Gamma=MinParams[1],Rho=MinParams[2], Table=Errors))}

#Split Data

T1<- 1:50; V1<-setdiff(1:250,T1)

T2<- 51:100; V2<-setdiff(1:250,T2)

T3<- 101:150; V3<-setdiff(1:250,T3)

T4<- 151:200; V4<-setdiff(1:250,T4)

T5<- 201:250; V5<-setdiff(1:250,T5)

TrainingSets <- list(X[T1,],X[T2,],X[T3,],X[T4,],X[T5,])

TrainingTargets <- list(Y[T1,],Y[T2,],Y[T3,],Y[T4,],Y[T5,])

ValidSets <- list(X[V1,],X[V2,],X[V3,],X[V4,],X[V5,])

ValidTargets <- list(Y[V1,],Y[V2,],Y[V3,],Y[V4,],Y[V5,])

#Compute Optimal params for each training set

ParamList<-list()

for (k in 1:5)

{ParamList[[k]]<-GridSearch(TrainingSets[[k]],TrainingTargets[[k]],ValidSets[[k]],ValidTargets[[k]])}

#Train Model using average of 5 predictors on Test data

ptm <- proc.time()

Predictions3<-matrix(,2500,5)

for (k in 1:5)

{

Gamma <- ParamList[[k]]$Gamma

Rho <- ParamList[[k]]$Rho

Predictions3[,k] <- GPPredictor(Gamma, Rho, X, Y, XT, YT)

}

ModelPredictions <- rowMeans(Predictions3)

proc.time() - ptm #time elapsed

Kernel2<-function(x1, x2, Gamma, Rho)

{Theta + (Gamma^2)\*exp(-(Rho^2)\*sum((x1-x2)^2))}

#Compute Gram Matrix column

GramColumn2 <- function(j)

{apply(tX,1,Kernel2,tX[j,], Gamma, Rho)}

ParamTable<-matrix(,5,2)

for (k in 1:5)

{ParamTable[k,] <- cbind(ParamList[[k]]$Gamma,ParamList[[k]]$Rho)

Gamma<-ParamList[[k]]$Gamma}

require("lattice")

Z1 <- matrix(ParamList[[1]]$Table[,3],20,20)

colnames(Z1)<-seq(0.01,1,0.05)

rownames(Z1)<-seq(0.1,10,0.5)

Z2 <- matrix(ParamList[[2]]$Table[,3],20,20)

colnames(Z2)<-seq(0.01,1,0.05)

rownames(Z2)<-seq(0.1,10,0.5)

Z3 <- matrix(ParamList[[3]]$Table[,3],20,20)

colnames(Z3)<-seq(0.01,1,0.05)

rownames(Z3)<-seq(0.1,10,0.5)

Z4 <- matrix(ParamList[[4]]$Table[,3],20,20)

colnames(Z4)<-seq(0.01,1,0.05)

rownames(Z4)<-seq(0.1,10,0.5)

levelplot(Z1, xlab="Gamma", ylab="Rho", main="Hyperparameter Error Values", col.regions = rainbow(100, start=0.5, end=0.66))

levelplot(Z2, xlab="Gamma", ylab="Rho", main="Hyperparameter Error Values", col.regions = rainbow(100, start=0.5, end=0.66))

levelplot(Z3, xlab="Gamma", ylab="Rho", main="Hyperparameter Error Values", col.regions = rainbow(100, start=0.5, end=0.66))

levelplot(Z4, xlab="Gamma", ylab="Rho", main="Hyperparameter Error Values", col.regions = rainbow(100, start=0.5, end=0.66))

#Compute MSE

SError <- round(mean((ModelPredictions-YT)^2),4) #Computing MSE

print(paste("The Mean squared error of the GP model on the testing data is", SError))

#############################################################

####Rescaling the covariates and repeating the experiment####

#############################################################

X[,1] <- X[,1]/10 ; X[,7] <- X[,7]/10

XT[,1] <- XT[,1]/10 ; XT[,7] <- XT[,7]/10

#Compute Optimal params for each training set

TrainingSets <- list(X[T1,],X[T2,],X[T3,],X[T4,],X[T5,])

TrainingTargets <- list(Y[T1,],Y[T2,],Y[T3,],Y[T4,],Y[T5,])

ValidSets <- list(X[V1,],X[V2,],X[V3,],X[V4,],X[V5,])

ValidTargets <- list(Y[V1,],Y[V2,],Y[V3,],Y[V4,],Y[V5,])

#Compute Optimal params for each training set

ParamList<-list()

for (k in 1:5)

{ParamList[[k]]<-GridSearch(TrainingSets[[k]],TrainingTargets[[k]],ValidSets[[k]],ValidTargets[[k]])}

#Train Model using average of 5 predictors on Test data

ptm <- proc.time()

Predictions3<-matrix(,2500,5)

for (k in 1:5)

{

Gamma <- ParamList[[k]]$Gamma

Rho <- ParamList[[k]]$Rho

Predictions3[,k] <- GPPredictor(Gamma, Rho, X, Y, XT, YT)

}

ModelPredictions <- rowMeans(Predictions3)

proc.time() - ptm #time elapsed

ParamTable2<-matrix(,5,2)

for (k in 1:5)

{ParamTable2[k,] <- cbind(ParamList[[k]]$Gamma,ParamList[[k]]$Rho)}

#Visualize the GP

VisFun<-function(Gamma,Rho)

{Kernel2<-function(x1, x2, Gamma, Rho)

{Theta + (Gamma^2)\*exp(-(Rho^2)\*sum((x1-x2)^2))}

#Compute Gram Matrix column

GramColumn2 <- function(j)

{apply(X,1,Kernel2,X[j,], Gamma, Rho)}

#Compute C inverse

C <- sapply(as.matrix(seq(1,nrow(X))),GramColumn2)

C <- C + Beta\*diag(nrow(X))

CInv <- solve(C)

Machine Learning Paper Review

**Automatic Conversational Scene Analysis in Children with Asperger Syndrome/High-Functioning Autism and Typically Developing Peers**

Published: January 29, 2014 to PLOS ONE

In this paper researchers Alessandro Tavano, Anna Pesarin, Vittorio Murino, and Marco Cristani from the Italian Ministry of Health developed and analyzed a model which can classify Asperger’s syndrome from patterns in speech.

The model relied on the state of mind-blindness, the inability to attribute mental state to one self and others, known to be a classic symptom of Aspeger’s syndrome. It is thought that this state can be inferred from patterns of stoppages in conversations, which is when mentalization is thought to take place. Patterns in communication for each participant was modelled as a six state Markov Chain. Conversations were recorded between a moderator and 9 children diagnosed with Asperger’s syndrome and 9 Neurotypical age matched children.

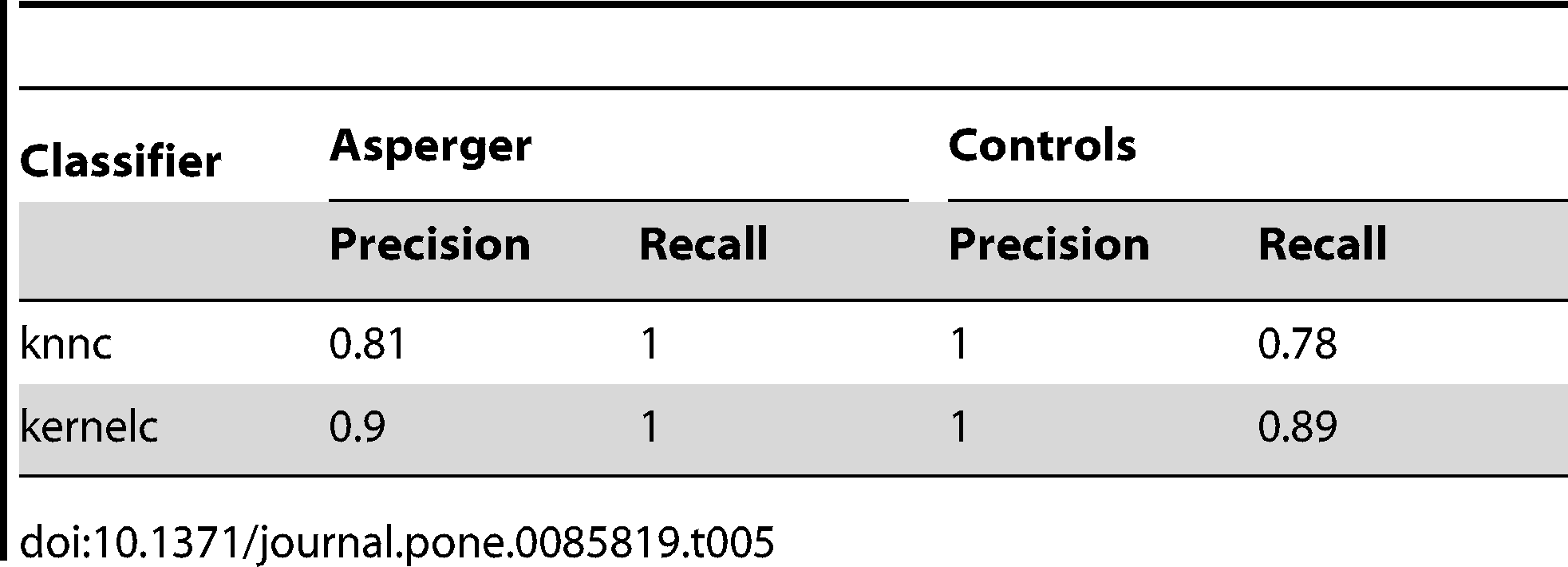
Each conversation was modelled using an Observed Influence Model (OIM) which allowed changes in the state of each Markov Chain to affect the other. The OIM is a simplification of a first order Hidden Markov Model (HMM) defined in the following way:

http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e046.PNG

http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e035.PNG

http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e054.PNGWhere: http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e047.PNG = ,

For http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e036.PNG, http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e037.PNG and http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e038.PNG is the probability of going from state http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e039.PNG of chain http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e040.PNG to state http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e041.PNG of the chain http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e042.PNG at recorded time point and http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e043.PNG represents the influence that chain http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e044.PNG exerts on http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e045.PNG. In our case: {1,…C} are the participants of the conversation (C=2 in this set-up), http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e051.PNG ∈CxC is defined as [http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e051.PNG]c,d = http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e043.PNG and http://journals.plos.org/plosone/article/asset?id=info:doi/10.1371/journal.pone.0085819.e041.PNG{Speech/Silence and whether conversation was Long/Med/ Short at timepoint t}. Note these labels were assigned by a Mixture of Gaussians model using the EM algorithm trained on manually labelled data. Each conversation was modelled by a separate ∈160, as there was 160 parameters to learn. Feature selection was employed on the, and the data was reduced to 2 parameters and then was fitted with KNN and SVM classifiers to predict whether or not the subject had Asperger’s, with the following results:



The main contributions of this paper are twofold. Firstly, the methods used in this paper have the potential be used as a cheap, non-invasive and fairly accurate diagnostic tool for Asperger’s syndrome. Secondly, using the OIM described above allows each’s parameters to have a simple interpretation, which gives meaning to the results of the feature selection procedure. The results of the OIM and Feature Selection procedure in fact supported a theory that conversational talent can be predicted by coordination of speech and attention rather than just linguistic abilities (like vocabulary length).

A flaw in this paper is that this technique is not practically useful as a predictive tool, since Asperger’s syndrome is generally simple to detect. Despite this, the approaches described in this paper seem to be readily generalizable to model other social disorders that may be more subtle (e.g. sociopathy). Another flaw is that the classifier uses a Hidden Markov Model, which is computationally intensive and relies on the first order Markov property, which might not be appropriate when modelling conversations. This can be fixed by changing the choice of model, although this comes at the expense of the interpretability of the parameters. Finally, the data set wasn’t very large and only consisted of two person conversations. Since the research was non-intrusive it seems that past recorded conversations can be collected and analyzed to increase the sample size. The model can also be easily generalized to be able to handle conversations with arbitrarily many participants. Finally, other more complex discriminative approaches can be used in the final classifier, since the KNN and SVM models used in this paper are quite simple.