> There are two kinds of people in the world: people who think there are two kinds of people in the world and people who don't

(borrowed from Menand (2018)). Because things are always simpler when we face only binary choice, aren't they? But consider here the case were multiple options are possible, and let us see if we cannot get back to simpler binary choices. Consider a collection of observations $(y_i,\boldsymbol{x}_i)$ where y_i is some categorical variable, $y_i\in\mathcal{A}$ where $\mathcal{A}=\lbrace A_1,\cdots,A_\kappa \rbrace$, with κ possible categories. Let $\mathcal{I}_k=\lbrace i:y_i\in A_k \rbrace$.

In a classical multinomial logistic regression, suppose that A_1 is the reference, then $\mathbb{P}[Y=A_j|\boldsymbol{X}=\boldsymbol{x}]=\frac{\exp[\boldsymbol{x}^\top\boldsymbol{\beta}_j]}{1+\exp[\boldsymbol{x}^\top\boldsymbol{\beta}_2]+\cdots+\exp[\boldsymbol{x}^\top\boldsymbol{\beta}_k]}$With a lot a categories, and a small number of observations, inference can be complicated, and non-robust.

- **the Siamese dataset**

The name *Siamese* I use, here, comes from Siamese Networks. Or sort of… As we say in French, it is an « *histoire de l'homme qui a vu l'homme qui a vu l'ours* » (story of the man who saw the man who saw the bear). A few years ago, a student tried to explain to me the idea of Siamese Networks and this is what I understood. I might be completely wrong, but the idea I got from it did make sense, in my mind at least. That is the story of that blog post…

The idea of the siamese algorithm will be to consider all pairs of observations, $(y_i,\boldsymbol{x}_i)$ and $(y_j,\boldsymbol{x}_j)$:

1. $\tilde y_{i,j}=\boldsymbol{1}(y_i=y_j)$ indicating if individuals i and j are in the same category
2. $\tilde{\boldsymbol{x}}_{i,j}$ is a collection of $p-1$ variables,

- $\tilde {x}_{k:i,j}={x}_{k:i}-{x}_{k:j}$ if x_k is **continuous**, or $\tilde {x}_{k:i,j}=|{x}_{k:i}-{x}_{k:j}|$ (we can use another metric, e.g. $\tilde {x}_{k:i,j}=|{x}_{k:i}-{x}_{k:j}|^2$), and this is why I decided to use some GAM model in the logistic regression on the Siamese dataset)
- $\tilde {x}_{k:i,j}=({x}_{k:i},{x}_{k:j})\in\mathcal{X}_k\times\mathcal{X}_k$ if x_k is a **categorical** variables (taking values in the set $\mathcal{X}_k$), or $\tilde {x}_{k:i,j}=\boldsymbol{1}({x}_{k:i}\neq{x}_{k:j})\in\{0,1\}$

The original dataset was a $n\times p$ matrix, and (if there are no categorical variable), it becomes a $n(n-1)/2\times p$ matrix. The key point is that if the original variable y_i was multinomial, $y_{i,j}$ is now binomial. For instance, if our initial dataset was the following, with two covariates, one continuous and one categorical

i	y_i	$x_{1:i}$	$x_{2:i}$
1	A	0	H
2	A	0	F
3	B	1	H
4	B	1	H
5	C	0	F
6	C	1	F

its siamese counterpart is the following

(i,j)	$\tilde y_{i,j}$	$\tilde x_{1:i}$	$\tilde x_{2:i}$
(1,2)	1	0	1
(1,3)	0	-1	0
(1,4)	0	-1	0
(1,5)	0	0	1
(1,6)	0	-1	1
(2,3)	0	-1	1
(2,4)	0	-1	1
(2,5)	0	0	0
(2,6)	0	-1	0
(3,4)	1	0	0
(3,5)	0	1	1
(3,6)	0	0	1
(4,5)	0	1	1
(4,6)	0	0	1
(5,6)	1	-1	0

- **Classification step**

On the dataset \((\tilde y_{i,j},\tilde {x}_{k:i,j})_{i,j}\), fit a logistic regression, \(\mathbb{P}[\tilde Y|\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{x}}]=\frac{\exp[\tilde{\boldsymbol{x}}^\top\boldsymbol{\beta}]}{1+\exp[\tilde{\boldsymbol{x}}^\top\boldsymbol{\beta}]}\)(or any classification model – CART, random forest, etc). But that is the easy part (unless \(n\) is large, because the siamese dataset has (roughly) \(n^2/2\) rows). The difficult task is the prediction

- **Prediction step**

Consider a new input variable \(\boldsymbol{x}_{\cdot}\), and define its siamese version, \(\tilde{\boldsymbol{x}}_{\cdot}=(\tilde{\boldsymbol{x}}_{\cdot,j})_j\), i.e. a database with \(n\) rows. Then compute
\(p_{\cdot,j}=\mathbb{P}[\tilde Y|\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{x}}_{\cdot, j}]=\frac{\exp[\tilde{\boldsymbol{x}}_{\cdot,j}^\top\boldsymbol{\beta}]}{1+\exp[\tilde{\boldsymbol{x}}_{\cdot,j}^\top\boldsymbol{\beta}]}\) where \(p_{\cdot,j}\) is the probability

that \((y_j,\boldsymbol{x}_{j})\) and \((y_{\cdot},\boldsymbol{x}_{\cdot})\) are in the same category, as well as \(p_{i,j}=\mathbb{P}[\tilde Y|\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{x}}_{i,j}]=\frac{\exp[\tilde{\boldsymbol{x}}_{i,j}^\top\boldsymbol{\beta}]}{1+\exp[\tilde{\boldsymbol{x}}_{i,j}^\top\boldsymbol{\beta}]}\)
Let \(\boldsymbol{p}_{\cdot}=(p_{\cdot,j})\), and similarly \(\boldsymbol{p}_{i}=(p_{i,j})\). Then several techniques can be used to predict \(y_{\cdot}\).

1. \(\widehat{y}_{\cdot}=y_{j^\star}\) where \({j^\star}=\underset{j=1,\cdots,n}{\text{argmax}}\{p_{\cdot,j}\}\): the predicted class is the one of the observation the most likely to be other same class
2. \(\widehat{y}_{\cdot}=y_{j^\star}\) where \({j^\star}=\underset{\ell=1,\cdots,k}{\text{argmax}}\{\overline{p}_{\ell}\}\), where\(\overline{p}_{\ell} = \frac{1}{n_{\ell}}\sum_{j\in\mathcal{I}_s} \boldsymbol{1}(y_j =y_{\ell}),\text{ where }\mathcal{I}_s=\lbrace i:p_{\cdot,i}>s\rbrace\)consider only probabilities sufficiently high to be considered, and predict the most important class (majority rule)
3. \(\widehat{y}_{\cdot}=y_{j^\star}\) where \({j^\star}=\underset{i=1,\cdots,n}{\text{argmax}}\{\theta_i\}\) where \(\theta_i=\cos(\boldsymbol{p}_{\cdot},\boldsymbol{p}_{i})=\displaystyle{\frac{\boldsymbol{p}_{\cdot}\cdot\boldsymbol{p}_{i}}{\|\boldsymbol{p}_{\cdot}\|\|\boldsymbol{p}_{i}\|}}\)
4. \(\widehat{y}_{\cdot}=y_{j^\star}\) where \({j^\star}=\underset{i=1,\cdots,n}{\text{argmax}}\{KL_{\cdot|i}\}\) and \(KL_{\cdot|i}=\displaystyle{\sum_{j=1}^n p_{\cdot,j}\log\frac{p_{\cdot,j}}{p_{i,j}}}\) (but one can select another metric)
5. \(\widehat{y}_{\cdot}=y_{j^\star}\) where \({j^\star}=\underset{j\in\mathcal{J}}{\text{argmax}}\{p_{\cdot,j}\}\) and \(\mathcal{J}\) is a sample of \(k\) observations, chosen randomly, one in each group (one-shot procedure): the predicted class is the one of the observation the most likely to be other same class

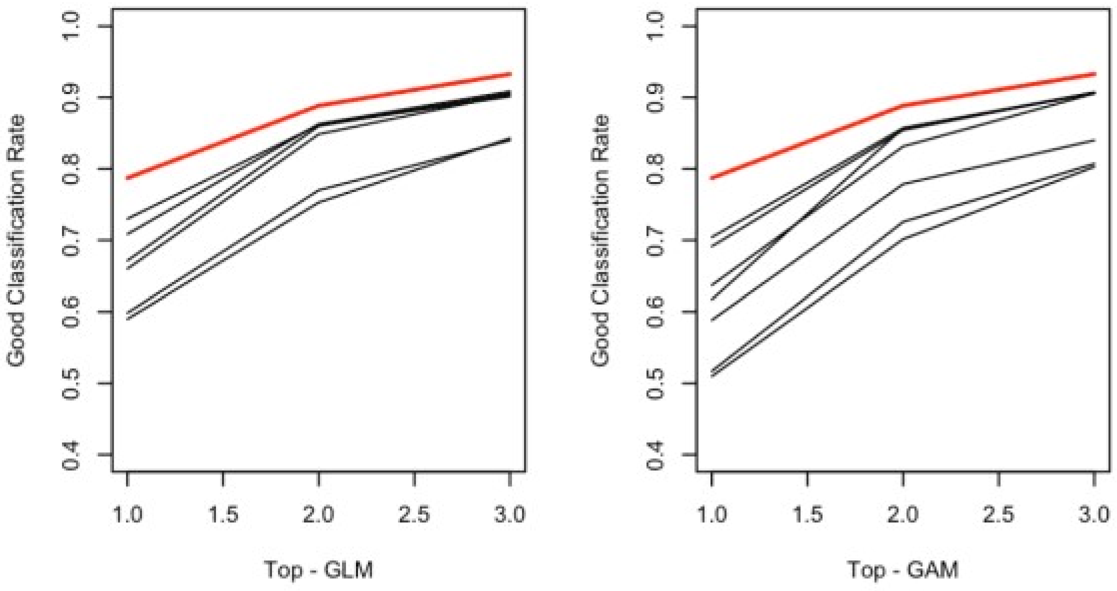
Heuristically, it can be related to some \(k\) nearest neighbors strategy: we give the attribute that most neighbors have. The total distance is a weighted sum of the componentwise distances (for the logistic regression).

- **Simulation study**

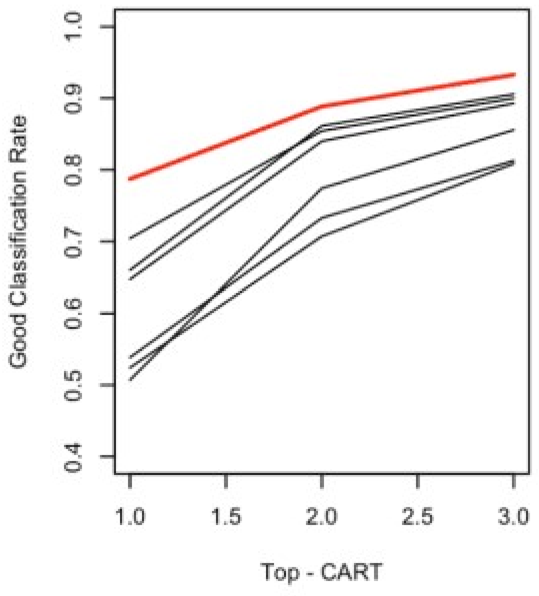
In order to test that technique, let us generate some multinomial model where \(y\) has 10 possible labels, with 6 (independent) covariates \(x_1,\cdots,x_6\), and \(\mathbb{P}[Y=A_k|\boldsymbol{X}=\boldsymbol{x}]\propto \exp[\boldsymbol{x}^\top\boldsymbol{\beta}_k]\) (where coefficients \(\boldsymbol{\beta}_k\) where generated randomly) for \(k\in\{1,2,\cdots,10\}\) (there were 10 categories) and with \(n=700\) observations.

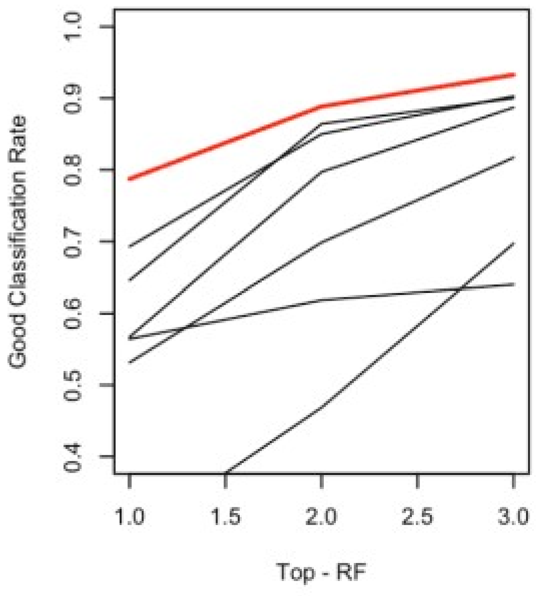
```
n=700
X1=rnorm(n)
X2=rnorm(n)
X3=rnorm(n)
X4=rnorm(n)
X5=rnorm(n)
X6=rnorm(n)
X=cbind(1,X1,X2,X3,sqrt(abs(X4)),X5*X1,X6)
k = 10
 PARAM = matrix(rnorm(k*6),k,6)
 PARAM[,1]=PARAM[,1]-1
 PARAM=cbind(PARAM,0)
 P=matrix(NA,n,k-1)
 for(j in 1:(k-1)) P[,j] = X %*% (PARAM[j,])+rnorm(n)
 P=cbind(P,0)
S=apply(exp(P),1,sum)
Pb = exp(P)/S
tirage = function(i){
       sample(1:10,size=1,prob = Pb[i,])
}
Y = LETTERS[Vectorize(tirage)(1:n)]
dbase = data.frame(Y=as.factor(Y),X1,X2,X3,X4,X5)
```

In the paragraph previously, I suggested to take the most likely one. Being wrong means that it was not the first choice. But perhaps being the second or the third choice is not that bad, actually. So in my simulations, I look at the proportion of predictions were our prediction is the good one (top 1), if the true one is either the most likely or the second most likely (top 2), or in the top 3. That will be on my \(x\)-axis. I draw some *line,* but we simply have three points (top 1, top 2 and top 3). I compute the proportion of good prediction, using cross-validation techniques (10-fold). The black *lines* are one of the methods described above. The red one is the standard multinomial model (with a logistic link function). For the Siamese model, I tried several models. I tried a logistic regression, and some smooth version (GAM) on top



and a classification tree, on the left, as well as some random forest on the right, below.





It looks like the multinomial approach performs always better than any Siamese one… and to be honest, I am disappointed.

Here is the code I did use when I considered a logistic regression on the Siamese dataset,

```
set.seed(1)
kfold = sample(rep(1:10,n/10))

KFOLDglm = function(i){
i_test=which(kfold==i)
i_calibration=which(kfold!=i)
y=credit[i_calibration,"Y"]
tirage = function(){
v=c(sample(i_calibration[y==levels(y)[1]],size=1),
    sample(i_calibration[y==levels(y)[2]],size=1),
    sample(i_calibration[y==levels(y)[3]],size=1),
    sample(i_calibration[y==levels(y)[4]],size=1),
    sample(i_calibration[y==levels(y)[5]],size=1),
    sample(i_calibration[y==levels(y)[6]],size=1),
    sample(i_calibration[y==levels(y)[7]],size=1),
    sample(i_calibration[y==levels(y)[8]],size=1),
    sample(i_calibration[y==levels(y)[9]],size=1),
    sample(i_calibration[y==levels(y)[10]],size=1))
names(v)=levels(y)
return(v)
}

LogisticModel <- multinom(Y ~ ., data = credit[i_calibration,],
trace=FALSE)

comparaisonx = function(base,x=base[1,]){
  mix_base = base
  for(j in 1:ncol(base)){
    xj = as.numeric(x[j])-base[,j]
    mix_base[,j] = (xj)
  }
  mix_base
}
comparaisony = function(base,y=base[1]){
  as.factor(base == y)
}
creditx = credit[,-which(names(credit) == "Y")]
nc=length(i_calibration)
B=comparaisonx(base = creditx[i_calibration[2:nc],],
x=creditx[i_calibration[1],])
B$Y=comparaisony(base = credit[i_calibration[2:nc],"Y"
],y=credit[i_calibration[1],"Y"])
for(i in 2:(nc-1)){
  B0=comparaisonx(base = creditx[i_calibration[(i+1):
nc],],x=creditx[i_calibration[i],])
  B0$Y=comparaisony(base = credit[i_calibration[(i+1):nc]
,"Y"],y=credit[i_calibration[i],"Y"])
  B=rbind(B,B0)
}
credit_mix = B

OneShotLogisticModel <- glm(Y ~ ., data = credit_mix, family=binomial)
A_ref = table(credit[i_calibration,"Y"])/length(i_calibration)

vect_oneshot = function(i){
  B2=comparaisonx(base = creditx[i_calibration,],x=creditx[i,])
  predict(OneShotLogisticModel,type="response",newdata=B2)
}

prediction_oneshot = function(i,type=1){
B2=comparaisonx(base = creditx[i_calibration,],x=creditx[i,])
p=predict(OneShotLogisticModel,type="response",newdata=B2)
y=credit[i_calibration,"Y"]
base = data.frame(p,y)
base = base[rev(order(base$p)),]
if(type==1){T = table(base$y[1:11])
return(names(which.max(T)))}
if(type==2){return(base$y[1])}
if(type==3){A=table(base$y[1:10])/10
T=A/A_ref
return(names(which.max(T)))}
if(type==4){
  costheta = rep(NA,length(i_calibration))
  for(j in 1:length(i_calibration)){
    vecteur_proba = vect_oneshot(i_calibration[j])
    costheta[j] = sum(vecteur_proba*p)/(sqrt(
sum(vecteur_proba^2))*sqrt(sum(p^2)))
  }
  return(y[which.max(costheta)])}
if(type==5){
  kl = rep(NA,length(i_calibration))
  for(j in 1:length(i_calibration)){
    vecteur_proba = vect_oneshot(i_calibration[j])
    kl[j] = sum(p*log(vecteur_proba/p))
  }
  return(y[which.max(as.vector(kl))])}
if(type==6){ ## one shot : tirer au hasard un de chaque, et dire lequel
est plus credible !

y=credit[i_calibration,"Y"]
tirage = function(){
    v=c(sample(i_calibration[y==levels(y)[1]],size=1),
        sample(i_calibration[y==levels(y)[2]],size=1),
        sample(i_calibration[y==levels(y)[3]],size=1),
        sample(i_calibration[y==levels(y)[4]],size=1),
        sample(i_calibration[y==levels(y)[5]],size=1),
        sample(i_calibration[y==levels(y)[6]],size=1),
        sample(i_calibration[y==levels(y)[7]],size=1),
        sample(i_calibration[y==levels(y)[8]],size=1),
        sample(i_calibration[y==levels(y)[9]],size=1),
        sample(i_calibration[y==levels(y)[10]],size=1))
    names(v)=levels(y)
    return(v)
  }
  pd=rep(NA,101)
for(ix in 1:101){
ids = tirage()
B2=comparaisonx(base = creditx[ids,],x=creditx[i,])
p=predict(OneShotLogisticModel,type="response",newdata=B2)
pd[ix]=levels(y)[which.max(p)]
}
levels(y)[which.max(table(pd))]
}
}
PRED0=as.character(credit[i_test,"Y"])
PRED1=as.character(predict(LogisticModel,type = "class",
                           newdata=credit[i_test,]))
PRED21=as.character(Vectorize(function(i) prediction_oneshot(i,type=1))
                    (i_test))
PRED22=as.character(Vectorize(function(i)
prediction_oneshot(i,type=2))(i_test))
PRED23=as.character(Vectorize(function(i)
prediction_oneshot(i,type=3))(i_test))
PRED24=as.character(Vectorize(function(i)
prediction_oneshot(i,type=4))(i_test))
PRED25=as.character(Vectorize(function(i)
prediction_oneshot(i,type=5))(i_test))
PRED26=as.character(Vectorize(function(i)
prediction_oneshot(i,type=6))(i_test))
B=data.frame(PRED0,PRED1,PRED21,PRED22,PRED23,PRED24,PRED25,PRED26)
B
}

s=1/100;setTxtProgressBar(pb, s*2)
```

```
for(i in 2:10){
  PREDICTION = rbind(PREDICTION,KFOLDglm(i))
s=s+1/100;setTxtProgressBar(pb, s*2)}
for(j in 1:5) PREDICTION[,j]=as.character(PREDICTION[,j])
L=list()
v=mean(PREDICTION[,1]!=PREDICTION[,2])
names(v)="logistic"
L[["logistic"]]=v
v=c(mean(PREDICTION[,1]!=PREDICTION[,3]),
  mean(PREDICTION[,1]!=PREDICTION[,4]),
  mean(PREDICTION[,1]!=PREDICTION[,5]),
  mean(PREDICTION[,1]!=PREDICTION[,6]),
  mean(PREDICTION[,1]!=PREDICTION[,7]),
  mean(PREDICTION[,1]!=PREDICTION[,8]))
names(v)=c("top10","max","10norm","cos","KL","OS")
L[["glm"]]=v
```