Resumo da análise para o projeto AMPARO

Rafael Bassi Stern

April 19, 2018

Este texto tem por fim resumir as análises que foram realizadas para o projeto AMPARO. A análise se divide em dois componentes que, até o momento, são relativamente independentes: o jogo do goleiro e a dupla tarefa. Cada uma destas análises é exposta a seguir.

Ainda que o banco de dados utilizado seja sigiloso, o código utilizado nesta análise está disponível aqui.

Jogo do Goleiro

Banco de dados

Participaram do jogo do goleiro pessoas saudáveis e com estágios variados da doença de Parkinson (HY de 1 a 3). Para fins deste estudo, o jogo do goleiro teve 6 fases, com os seguintes objetivos:

1. **Fase 1 - Aquecimento**: Apertar as 5 direções que eram indicadas pelo jogo.
2. **Fase 2 - Determinística explícita**: Descobrir a regra de uma sequência determinística de tamanho 8, tendo acesso a um auxílio de memória.
3. **Fase 3 - Determinística implícita**: Descobrir a regra de uma sequência determinística de tamanho 12, não tendo acesso a um auxílio de memória.
4. **Fase 4 - Aleatória I**: Descobrir a regra da sequência aleatória I de tamanho 40.
5. **Fase 5 - Aleatória II**: Descobrir a regra da sequência aleatória II de tamanho 60.
6. **Fase 6 - Memória**: Memorizar uma sequência de tamanho 15 e repetí-la.

Consideramos que jogadores que não obtiveram ao menos 3 acertos na Fase 1 não estavam preparadas para participar do jogo e retiramo-las da amostra. O número de jogadores remanescentes é indicado a seguir.

```
players <- data %>%
  filter(game == "AQ", total_correct >= 3) %>%
  select(player_alias) %>%
  distinct()
nrow(players)
```

```
## [1] 72
```

Também observamos que, em geral, os jogadores não aprenderam as regras na Fase 4 e 5. Também, a Fase 1 era muito simples, tendo como objetivo apenas verificar se o jogador conseguia interagir com o console. Por isso, mantivemos para na análise apenas as Fases 2, 3 e 6.

Extração de informação

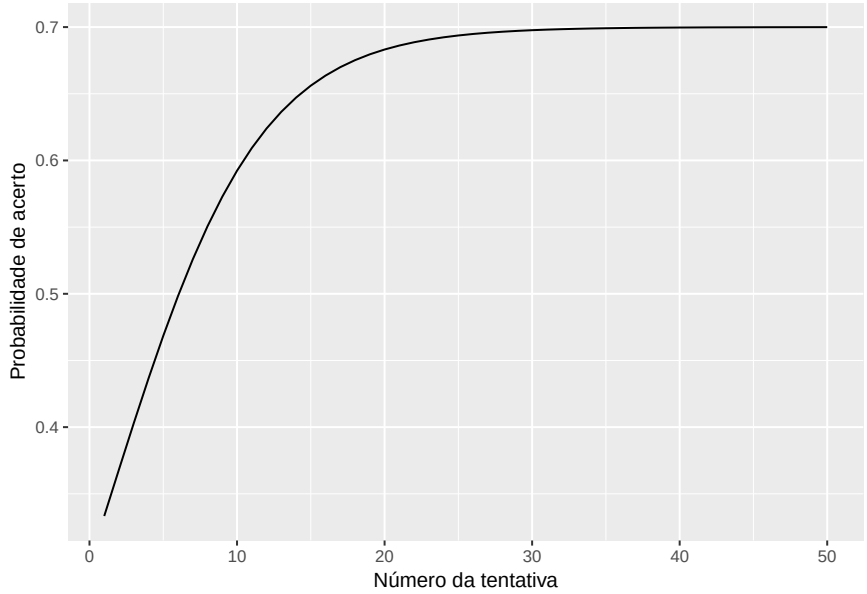
Dadas estas restrições, prosseguir para a análise do desempenho dos jogadores em cada fase. Definimos como $X_{f,j,t}$ a indicadora de que o jogador j acertou o t-ésimo elemento da sequência da fase f. Também definimos por $e(j)$ a escolaridade do jogador j. O modelo que usamos é descrito a seguir:

$$
\begin{aligned}
\alpha_{f,e(j)} &\sim N(0,1) \\
\beta_{f,j} &\sim N(\alpha_{e(j)},1) \\
\lambda_{f,j,t} &= (t-1)\beta_{f,j} - \log(3\gamma_{f,j}-1) \\
P(X_{f,j,t}=1) &= \gamma_{f,j} \cdot \frac{\exp(\lambda_{f,j,t})}{1+\exp(\lambda_{f,j,t})}
\end{aligned}
$$

Os parâmetros deste modelo indicam os diferentes efeitos que podem atuar sobre a probabilidade de acerto de um jogador. $\alpha_{f,e}$ indica o efeito que a escolaridade e tem sobre a taxa de aprendizado na fase f. $\beta_{f,j}$ é a taxa de aprendizado do jogador j na fase f. No modelo, $\beta_{f,j}$ é centrado em $\alpha_{f,e}$, isto é, a taxa de aprendizado de cada jogador é centrada na taxa de aprendizado para a escolaridade deste. Finalmente, o limite de aprendizado do jogador j na fase f é denotada por $\gamma_{f,j}$. Estes parâmetros são alinhados de tal forma que a probabilidade de acerto no primeiro lançamento é 3^{-1} e cresce em cada lançamento de acordo com $\beta_{f,j}$ até convergir para $\gamma_{f,j}$, quando o número de lançamentos vai para infinito.

Um exemplo de uma curva de aprendizado quando $\beta_{f,j} = 0.2$ e $\gamma_{f,j} = 0.7$ é apresentada a seguir. Notamos que, a probabilidade de acerto começa em 3^{-1} na primeira jogada e sobe até 0.7. A velocidade com que ocorre essa subida é dada por $\beta_{f,j}$.

```
tempos = 1:50
prob = function(t) {
  lambda = 0.2*(t-1)-log(3*0.7-1)
  0.7 * exp(lambda)/(1+exp(lambda))
}
data_frame(tempos = tempos, probs = prob(tempos)) %>%
  ggplot(aes(x = tempos, y = probs)) +
  geom_line() +
  ylab("Probabilidade de acerto") +
  xlab("Número da tentativa")
```



Para cada jogador j e fase f, obtivemos a média da posteriori para $\beta_{f,j}$ e $\gamma_{f,j}$. Assim, para cada jogador, obtivemos 6 valores, um para cada combinação de fase e tipo de parâmetro. Estes valores resumem o seu desempenho no jogo do goleiro.

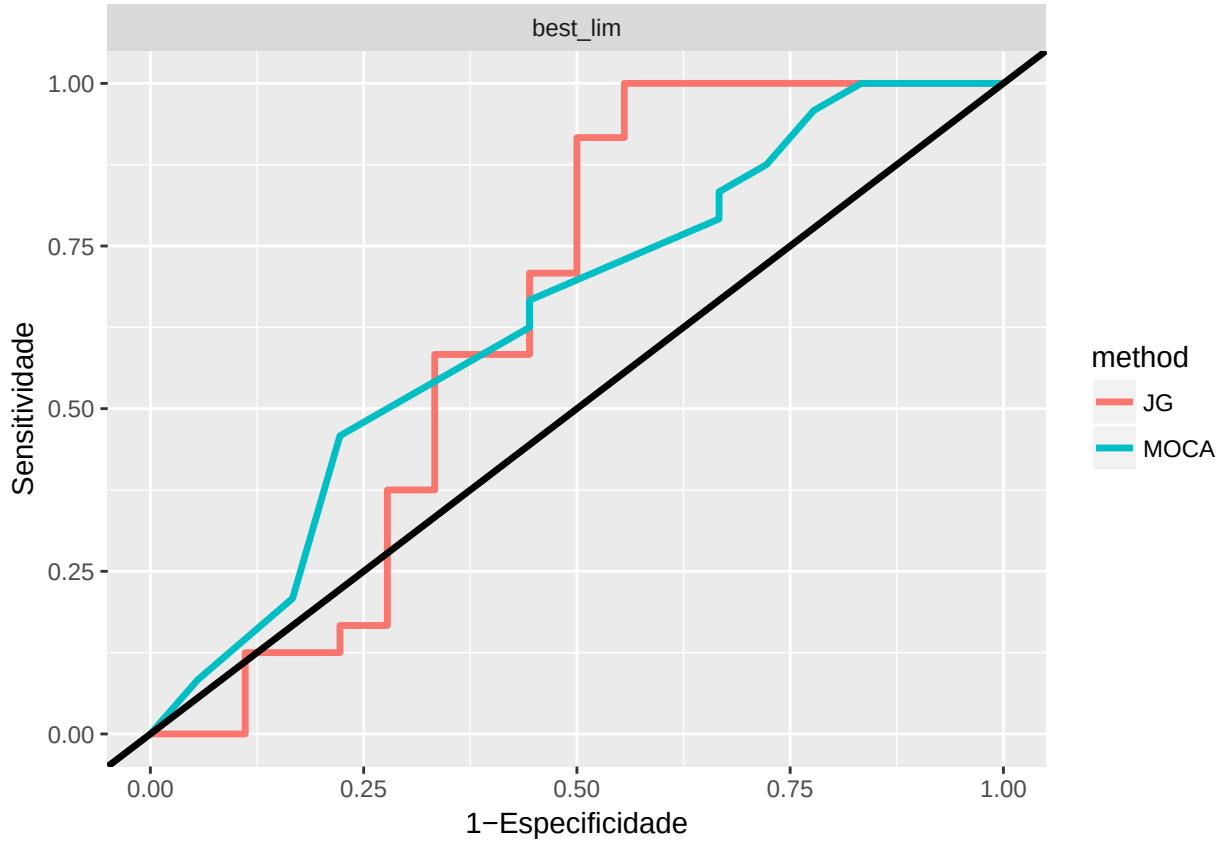
Classificação

A seguir, utilizamos estas medidas de performance no Jogo do Goleiro para construir preditores de outras características clínicas dos jogadores dadas pelos testes BEST, UPDRS e MOCA. Para cada característica destes testes, desejamos predizer o jogador está acima ou abaixo da mediana para esta característica. Para ajustar estes preditores, utilizamos uma regressão logística com penalização nos coeficientes dada por uma rede elástica e seleção do parâmetro de penalização por validação cruzada. Comparamos os preditores utilizando o Jogo do Goleiro com preditores que usavam o MOCA total.

A figura abaixo apresenta a curva ROC para os preditores ajustados usando o MOCA total e o Jogo do Goleiro para predizer a característica "best_lim". Verificamos que, em relação ao preditor baseado no MOCA total, o preditor baseado no Jogo do Goleiro é capaz de obter alta sensitividade sacrificando menos especificidade.

```
data_roc_all = read_rds("../data/amparo/amparo-JG-classify.rds")
data_roc_all %<>%
  filter(method=="GLMNET" | method=="MOCA")
data_roc_all$method[data_roc_all$method == "GLMNET"] <- "JG"

data_roc_all %>%
  filter(variable=="best_lim") %>%
  ggplot()+
  geom_line(aes(x = espec, y = sens, color = method), size = 1.2)+
  xlab("1-Especificidade")+
  ylab("Sensitividade")+
  geom_abline(size = 1.2)+
  facet_wrap( ~ variable, ncol = 4)
```



Outras variáveis para as quais construímos preditores foram:

```
 data_roc_all$variable %>% table() %>% names()
```

```
##  [1] "best_lim"    "best_marcha" "best_orient" "best_reat"   "best_rest"
##  [6] "best_tot"    "best_trans"  "moca_abs"    "moca_ate"    "moca_evoc"
## [11] "moca_lin"    "moca_nom"    "moca_ori"    "moca_tot"    "moca_vis"
## [16] "updrs_post"  "updrs_rig"   "updrs_tot"   "updrs_trem"
```

Dupla tarefa

Banco de Dados

Neste banco de dados, cada paciente realizou 3 atividades:

1. **Tarefa simples motora**: Andar o máximo possível em linha reta em 30s.
2. **Tarefa simples verbal**: Dizer o maior número possível de palavras com letra inicial especificada em 30s.
3. **Tarefa dupla** Realizar ambas as tarefas anteriores ao mesmo tempo em 30s.

Assim, foram obtidas quatro medidas para cada paciente A distância percorrida em tarefa simples e dupla, ts_m e ts_d, e o número de palavras ditas em tarefa simples e dupla, ts_v e ts_d.

Agrupamento

Para obter intuição sobre estas medições, realizamos agrupamentos dos pacientes segundo elas. Para tal, utilizamos a técnica do **agrupamento espectral**. Esta consiste em realizar uma análise de componentes principais nas medições e aplicar uma técnica de agrupamento nas projeções nos componentes principais. A técnica de agrupamento escolhida foi a das k-médias.

Agrupamento espectral no banco de dados inteiro

Interpretamos os componentes principais obtidos a seguir:

```
dt_pca = dt_pure %>%
  as.matrix() %>%
  prcomp(center = TRUE, scale = TRUE)
dt_pca
```

```
## Standard deviations (1, .., p=4):
## [1] 1.5017474 1.1916863 0.4664539 0.3271989
##
## Rotation (n x k) = (4 x 4):
##            PC1        PC2         PC3        PC4
## td_m 0.5465002 -0.4366568  0.09505923 -0.7082599
## td_v 0.4599918  0.5415748  0.69430268  0.1142281
## ts_m 0.5124465 -0.5000837 -0.06379779  0.6951580
## ts_v 0.4765959  0.5156974 -0.71051909 -0.0455545
```

```
dt_pca$sdev
```

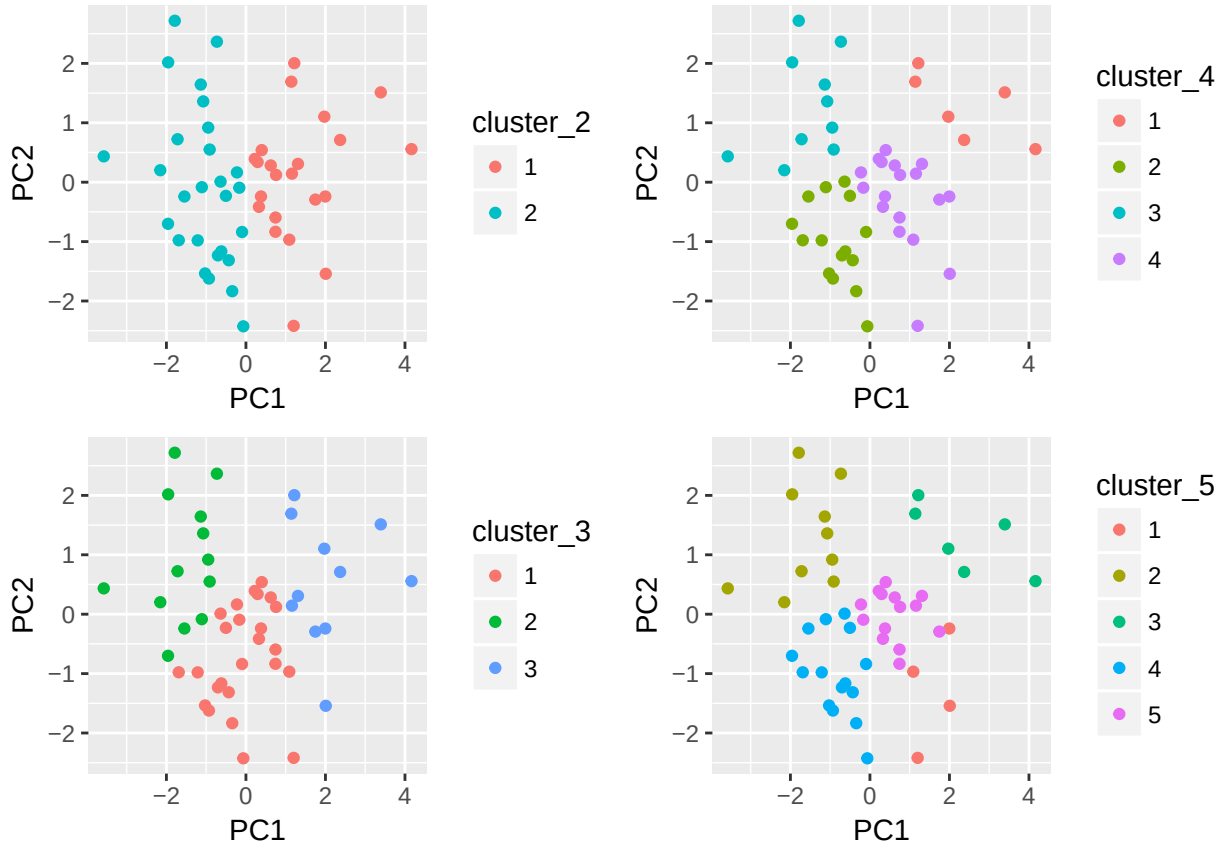
```
## [1] 1.5017474 1.1916863 0.4664539 0.3271989
```

1. **PC1**: Aproximadamente a soma de todas as variáveis. É uma medida de performance total.
2. **PC2**: Aproximadamente a soma das medidas verbais subtraídas das medidas motoras. Indica se o paciente tem um desempenho melhor nas característica verbais ou motoras.
3. **PC3**: Aproximadamente a dupla tarefa verbal subtraída da tarefa simples verbal. Indica o custo de dupla tarefa verbal.
4. **PC4**: Aproximadamente a tarefa simples motora subtraída da dupla tarefa motora. Indica o custo de dupla tarefa motora.

Os auto-valores indicados mostram que existe muito mais variabilidade dos dados em **PC1** e **PC2** do que em **PC3** e **PC4**.

Os gráficos abaixo indicam os agrupamento obtidos usando de 2 a 5 grupos nos componentes principais.

```
multiplot(c2, c3, c4, c5, cols=2)
```



Agrupamento espectral nos custos

Como as medições no banco de dados variam mais em relação a performances totais do que em relação ao custos em cada tarefa, estes são pouco utilizados no agrupamento acima. A seguir, fizemos um agrupamento utilizando apenas os custos motor (idx_m) e verbal (idx_v), definidos a seguir:

```
dt_idx = dt_pure %>%
  mutate(idx_m = (ts_m-td_m)/ts_m,
         idx_v = (ts_v-td_v)/ts_v) %>%
  select(idx_m, idx_v)
```

Os componentes principais são apresentados a seguir. Verificamos que ambos são responsáveis por variações semelhantes e que **PC1** indica custo total e **PC2** a diferença entre custo verbal e motor.

```
dt_pca_idx = dt_idx %>%
  as.matrix() %>%
  prcomp(center = TRUE, scale = TRUE)
dt_pca_idx
```

```
## Standard deviations (1, .., p=2):
## [1] 1.0991321 0.8898925
##
## Rotation (n x k) = (2 x 2):
##               PC1        PC2
## idx_m -0.7071068 -0.7071068
## idx_v -0.7071068  0.7071068
```

```
dt_pca_idx$sdev
```

```
## [1] 1.0991321 0.8898925
```

Os gráficos de agrupamento para 2 a 5 grupos estão exibidos a seguir:

```
multiplot(c2, c3, c4, c5, cols = 2)
```