Algorítimos de *machine learning* aplicado na analise de dados de EMG

1st Manoel Messias de Assis Júnior
Universidade de fortaleza
T951-09 Sistemas inteligentes
Fortaleza, Ceará, Brasil
Matricula: 2020335

***Resumo*—A Pesquisa se desenvolve através de dados capturados via EMG (Eletromiografia), perfazendo de estatísticas e algorítimos de classificação.**

***Index Terms*—EMG, Eletromiografia, Machine Learning, Estatísticas**

I. Introdução

A. Introdução a Machine Learning

Machine learing é um campo da ciência de dados devotado ao entendimento e a construção de métodos de aprendizagem, onde por meios de classificação dos dados, podemos prever dados futuros que se concentra no uso de dados e algoritmos para imitar a maneira como os humanos aprendem, melhorando gradualmente sua precisão.

B. Introdução a EMG (Eletromiografia)

A Eletromiografia (EMG) é um exame que mede a atividade elétrica dos músculos em resposta de sua própria estimulação. Na medicina é utilizada para detectar anormalidades neuromusculares, o EMG mede os estímulos do músculo, contrações leves ou fortes, músculo em repouso, os músculos em repouso geralmente não apresentam atividade elétrica.

II. Origem dos dados

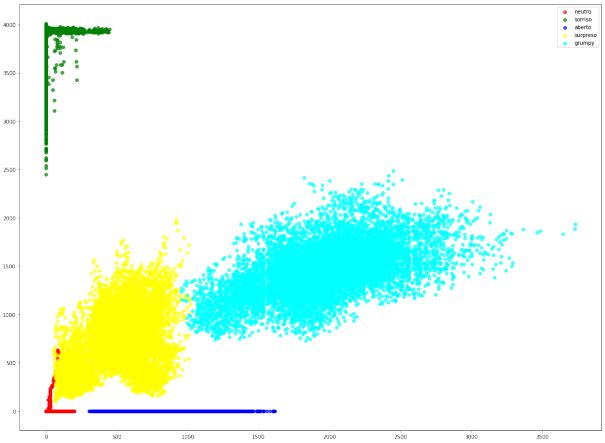
A origem dos dados foram obtidos por eletromiografia, onde foram postos dois eletrodos no rosto, posicionados no corrugador do super cílio e no zigomático maior. Cada rotulo se divide um cinco expressões faciais, neutro, sorrindo, aberto, surpreso e grumpy, os dados da EMG variam em 0khz a 1khz.

A. Conjunto dos dados

Os $dados \in \mathbb{R}^{NxP}$ onde suas dimensões, P são 2 preditores e N são 50000 observações, os $rotulos \in \mathbb{R}^{NxC}$ onde N é são amostras 5 e C são 50000 classes.

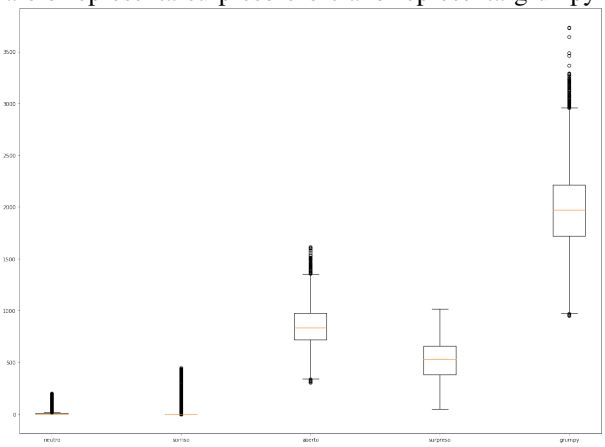
A rotina de medição foi 1000 amostras de cada expressão facial repetidas 10 vezes resultando em 50000 amostras.

B. Visualizando os dados



Scatter Plot

Cada cor refere-se a uma expressão facial. Verde representa o sorriso, vermelho representa neutro, azul representa aberto, amarelo representa surpreso e o ciano representa grumpy



Box Plot

III. Fundamentação Teórica

Classificadores de dados matriciais

Classificadores de dados matriciais, são ótimos para grandes volumes de dados como este em que está sendo discutido, trazendo confiabilidade, velocidade e simplicidade nos modelos.

A. OLS (Ordinary Least Squares)

O primeiro modelo a ser discutido, OLS (Ordinary Least Squares) para o português MQO (Mínimos quadrados ordinários), é uma técnica de estimação dos coeficientes de uma regressão linear aos quais descreve o relacionamento com uma ou mais variáveis sendo elas quantitativas ou não.

O modelo é dado pela seguinte equação linear:

$$\beta = (X^T X)^{-1} X^T y \tag{1}$$

onde X, é a matriz de dados e y é os rotulos

B. Regularização por Tikhonov

Um método usado para evitar mau-condicionamento. Em geral, o método fornece maior eficiência em problemas de estimativa de parâmetros em troca de uma quantidade tolerável de viés

Representada somente pela adição do termo de regularização λ onde varia de $0 \leqslant \lambda \leqslant 1$ multiplicado por uma matriz identidade $I \in \mathbb{R}^{PxP}$ e somada a primeira parte da equação

Depois da adição λI ganha a seguinte forma:

$$\beta = (X^T X + \lambda I)^{-1} X^T y \tag{2}$$

C. PDF (Probability Density Function)

A PDF (Probability Density Function), para o português FDP (Função de Densidade e Probabilidade), também conhecida como classificador Navie Bayers, distribuição gaussiana, multivariada distribuição normal, ou somente como normal, método de classificação a qual base-a-se na probabilidade da variável está dentro da integral.

PDF, Navie Bayes, Distribuição Gaussiana:

$$p(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-(x-\mu)^2}{2\sigma^2} \tag{3}$$

Multivariada Distribuição Normal:

$$p(x|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp -\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu) \tag{4}$$

onde μ é o vetor de médias, $\mathbf{x}$ matriz de dados separado por classe, Σ representa a matriz de covariância.

D. Multivariada Distribuição Normal regularizada por Friedman

Método usado para melhorar a degradação da função discriminante (5) para dados de alta-dimensibilidade, precisão em poucos dados ou não invertibilidade da matriz.

A função discriminante é dada pela seguinte equação:

$$g_i(\mathbf{x}_n) = (\mathbf{x}_n - \mu_i)^T\Sigma^{-1}(\mathbf{x}_n - \mu_i) \tag{5}$$

Tal método faz combinação linear da matriz agregada das classes:

$$\begin{aligned}\Sigma_{agregada} &= \frac{n_1}{N}\Sigma_1 + \frac{n_2}{N}\Sigma_2 + ... + \frac{n_c}{N}\Sigma_c \\ &= P(y_1)\Sigma_1 + P(y_2)\Sigma_2 + ... + P(y_c)\Sigma_c \\ &= \sum_{i=1}^{C} P(y_i)\Sigma_i\end{aligned} \tag{6}$$

Regularização por Friedman:

$$\Sigma_i^\lambda = \frac{(1-\lambda)(n_i \cdot \Sigma_i) + (\lambda \cdot N \cdot \Sigma_{agregada})}{(i-\lambda)n_i + \lambda \cdot N} \tag{7}$$

De maneira que λ receba $0 \leqslant \lambda \leqslant 1$ e resulte em uma nova matriz de covariância Σ.

E. Multivariada Distribuição Normal Pooled

Este método faz uma abordagem com a matriz de covariância agregada $\Sigma_{agregada}$ (6).

Todos esses métodos dependem do resultado do calculo da matriz de covariância Σ.

IV. Metodologia

Primeiro fazemos um algorítimo de pre-processamento nos dados, depois segmentamos os dados 80/20 sendo 80% para o treinamento do modelo e 20% para o teste do modelo.

Algorítimos no final do artigo

V. Resultados

A. OLS

Media	Min	Max	Desvio-Padrão
-0.60	-2.34	2.27	0.55

B. OLS regularizada por Tikhonov

λ	Media	Min	Max	Desvio-Padrão
0.00	-0.60	-2.39	2.35	0.55
0.10	-0.60	-2.39	2.35	0.55
0.20	-0.60	-2.39	2.35	0.55
0.30	-0.60	-2.39	2.35	0.55
0.40	-0.60	-2.39	2.35	0.55
0.50	-0.60	-2.39	2.35	0.55
0.60	-0.60	-2.39	2.35	0.55
0.70	-0.60	-2.39	2.35	0.55
0.80	-0.60	-2.39	2.35	0.55
0.90	-0.60	-2.39	2.35	0.55

VI. Conclusões

A. OLS

Algorítimo fácil de implementar, muito veloz, se-comporta bem com muitos dados, porem menos precisa.

B. OLS com regularização por Tikhonov

Fácil de implementar como o anterior, não houve diferenças significativas

C. FDP

Complexo de implementar e trabalhoso, exige um poder de computação maior, apesar de não conseguir mostrar os resultados. A equação acaba dependendo muito da matriz de covariância.

D. FDP pooled

Complexo como o anterior, porem demanda mais poder, por agregar a matriz de todas as classes.

E. FDP regularizada por friedman

Vantajoso quando se tem muitos dados, mas demanda de um processamento bom.

Algorithm 1: OLS sem regularização

Input: $X_{treino} \in \mathbb{R}^{NxP}, Y_{treino} \in \mathbb{R}^{NxC}$
Output: $\beta \in \mathbb{R}^{NxC}$
Function `Estimação(`X, y`)`**:**
  X = concatene uma vetor coluna de 1's a X
  return $(X^T X)^{-1} X^T y$

Input: $X_{teste} \in \mathbb{R}^{NxP}, \beta \in \mathbb{R}^{NxC}$
Output: $Y_{previsto}$
Function `Previsão(`X, β`)`**:**
  X = concatene uma vetor coluna de 1's a X
  return $X \cdot \beta$

Input: $X_{treino} \in \mathbb{R}^{NxP}, Y_{treino} \in \mathbb{R}^{NxC}, X_{teste} \in \mathbb{R}^{NxP}$
Function `OLS()`**:**
  X = X_{treino}
  Y = Y_{treino}
  β = `Estimação(`*X,Y*`)`
  $Y_{previsto}$ = `Previsão(`X_{teste},β`)`
  return $Y_{previsto}$

Algorithm 2: OLS com regularização

Input: $X_{treino} \in \mathbb{R}^{NxP}, Y_{treino} \in \mathbb{R}^{NxC}, \lambda \in \mathbb{R}$
Output: $\beta \in \mathbb{R}^{NxC}$
Function `Estimação(`X, y,λ`)`**:**
  X = concatene uma vetor coluna de 1's a X
  return $(X^T X + \lambda I)^{-1} X^T y$

Input: $X_{teste} \in \mathbb{R}^{NxP}, \beta \in \mathbb{R}^{NxC}$
Output: $Y_{previsto}$
Function `Previsão(`X, β`)`**:**
  X = concatene uma vetor coluna de 1's a X
  return $X \cdot \beta$

Input: $X_{treino} \in \mathbb{R}^{NxP}, Y_{treino} \in \mathbb{R}^{NxC}, X_{teste} \in \mathbb{R}^{NxP}$
Function `OLS()`**:**
  X = X_{treino}
  Y = Y_{treino}
  β = `Estimação(`*X,Y*`)`
  $Y_{previsto}$ = `Previsão(`X_{teste},β`)`
  return $Y_{previsto}$

Algorithm 3: Multivariada Distribuição Normal

Input: $X_{treino} \in \mathbb{R}^{NxP}$
Output: $\beta \in \mathbb{R}^{NxC}$
Function `Classificar(`*X*`)`**:**
  Crie um dicionario que armazene as classes
  Percorra o X em linhas
  Acada 1000 linhas reserve para uma classe
  return dicionario

Input: $X_{treino} \in \mathbb{R}^{NxP}$,
Output: $p(x|\mu, \Sigma) \in \mathbb{R}^{NxN}$
Function `Estimação(`X`)`**:**
  Percorra cada classe em X
  Estime as μ de cada classe
  Estime as Σ cada classe
  Estime para cada classe
  $g_i(x, \Sigma, \mu) - \frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)$
  $p(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(g_i(x, \Sigma, \mu)\right)$
  return lista de $p(x|\mu, \Sigma)$

Input: $X_{treino} \in \mathbb{R}^{NxP}$
Function `FDP()`**:**
  X = X_{treino}
  Y = Y_{treino}
  X_c^{PxC} = `Classificar(`*X*`)`
  lista de $Y_{previsto}$ = `Estimação(`X`)`
  return lista de $Y_{previsto}$

Algorithm 4: Multivariada Distribuição Normal (pooled)

Input: $X_{treino} \in \mathbb{R}^{NxP}$
Output: $\beta \in \mathbb{R}^{NxC}$
Function `Classificar(X))`**:**
  Crie um dicionario que armazene as classes
  Percorra o X em linhas
  Acada 1000 linhas reserve para uma classe
  return dicionario

Input: $X_{treino} \in \mathbb{R}^{NxP}$,
Output: $p(x|\mu, \Sigma) \in \mathbb{R}^{NxN}$
Function `Estimação(X)`**:**
  Percorra cada classe em X
  Estime as μ de cada classe
  Estime as $\Sigma_{agregada}$ com todas classe
  Estime para cada classe
  $g_i(x, \Sigma_{agregada}, \mu) - \frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}_{agregada}(\mathbf{x} - \mu)$
  $p(x|\mu, \Sigma) =$
  $\frac{1}{\sqrt{(2\pi)^d |\Sigma_{agregada}|}} \exp\left(g_i(x, \Sigma_{agregada}, \mu)\right)$
  return lista de $p(x|\mu, \Sigma)$

Input: $X_{treino} \in \mathbb{R}^{NxP}$
Function `FDP()`**:**
  X = X_{treino}
  Y = Y_{treino}
  X_c^{PxC} = `Classificar(X)`
  lista de $Y_{previsto}$ = `Estimação(X)`
  return lista de $Y_{previsto}$

Algorithm 5: Multivariada Distribuição Normal (regularizada)

Input: $X_{treino} \in \mathbb{R}^{NxP}$
Output: $\beta \in \mathbb{R}^{NxC}$
Function `Classificar(X))`**:**
  Crie um dicionario que armazene as classes
  Percorra o X em linhas
  Acada 1000 linhas reserve para uma classe
  return dicionario

Input: $X_{treino} \in \mathbb{R}^{NxP}$,
Output: $p(x|\mu, \Sigma) \in \mathbb{R}^{NxN}$
Function `Estimação(X)`**:**
  Percorra cada classe em X
  Estime as μ de cada classe
  Estime as $\Sigma_{agregada}$ com todas classe
  Estime as $\Sigma_i^\lambda = \frac{(1-\lambda)(n_i \cdot \Sigma_i) + (\lambda \cdot N \cdot \Sigma_{agregada})}{(i-\lambda)n_i + \lambda \cdot N}$ com todas classe
  Estime para cada classe
  $g_i(x, \Sigma_i^\lambda, \mu) = \frac{1}{2}(x - \mu)^T {\Sigma_i^\lambda}^{-1}(x - \mu)$
  $p(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_i^\lambda|}} \exp\left(g_i(x, \Sigma_i^\lambda, \mu)\right)$
  return lista de $p(x|\mu, \Sigma)$

Input: $X_{treino} \in \mathbb{R}^{NxP}$
Function `FDP()`**:**
  X = X_{treino}
  Y = Y_{treino}
  X_c^{PxC} = `Classificar(X)`
  lista de $Y_{previsto}$ = `Estimação(X)`
  return lista de $Y_{previsto}$