



Extração de características para Análise de Sentimentos

2º Projeto - Computação Evolutiva 2019.1

Raissa Camelo Salhab

- O que é mineração de texto?
- O que é análise de sentimento?

- O que é mineração de texto?
- O que é análise de sentimento?





Estado da Arte

- Utilização de algoritmos evolutivos para sumarização de textos. [Ramiz et al.] Algoritmo de Evolução Diferencial Discreto.
- Attribute Selection [Freitas] Determinar regras de classificação [Frank et al.].
- Clusterização de Documentos [Abraham et al.]




Utilizando Computação Evolutiva para a Otimização de Classificadores


- Otimização dinâmica de Grupos para redes neurais de feed-forward [Tang et. Al]
- Otimização de pesos de redes neurais utilizando computação evolutiva.
- Otimização de parametros de KNN e de Redes neurais [Tharwat et al.]

Base de Dados: LarissaNet




 negative.parsed



 positive.parsed




 steamUN.txt





 itunesUN.txt



 negative.parsed



 positive.parsed



Modelagem: Pré-processamento

- Tokenização / Lemmatization
- Extração de caracteres irrelevantes (números acentos, etc).
- Extração de preposições e pronomes.
- TF-IDF

$$\log(N/df)$$

$$TF = nt/n$$



Modelagem: Algoritmo Evolucionário

- Cada individuo consistirá em uma série de 1.200 valores de 0 à 1. Cada um mapeando uma característica do vetor TF-IDF (palavras).
- As características que possuírem um valor acima de 50% serão utilizadas para a validação com a Random Forest
- A população inicial será gerada aleatoriamente (20 indivíduos)

Funções de Fitness

- Utilizar a acurácia de teste para calcular o fitness de cada indivíduo.



Algoritmos Implementados

- Algoritmo Genético
- Estratégias Evolutivas
- Evolução Diferencial
- Programação Evolutiva
- Estratégias Evolutivas: Variação da Cauchy
- Estratégias Evolutivas: Variação da Cauchy + Variação de crossover
- GSO
- PSO



Crossover 1

```
def crossover(father, mother, cross):  
    crossing = np.random.uniform(low=0, high=1.0, size=(len(father)))  
    crossing = crossing > cross  
    new_born = []  
    for i in range(len(father)):  
        if crossing[i]:  
            new_born.append(father[i])  
        else:  
            new_born.append(mother[i])  
  
    return new_born
```

Crossover 2

```
def crossover(father, mother):  
    half = int(len(father)/2)  
    trues = np.ones(half)  
    falses = np.zeros(half)  
    bool_array = concatenate(trues, falses)  
    random.shuffle(bool_array)  
  
    new_born = []  
    for i in range(len(father)):  
        if bool_array[i]:  
            new_born.append(father[i])  
        else:  
            new_born.append(mother[i])  
  
    return new_born
```

Mutação

```
def mutation(hemafrodite, mut):  
    new_mutation = []  
  
    r1 = (math.sqrt(((2 * len(hemafrodite)))) ** -1  
    r2 = (math.sqrt(math.sqrt(((4 * len(hemafrodite)))))) ** -1  
    #print(r1, r2)  
    for gene in hemafrodite:  
        o = mut * np.exp((r1 * random.gauss(0, 1)) + (r2 * random.gauss(0, 1)))  
        new_gene = gene + (o * random.gauss(0, 1))  
        if new_gene > 0:  
            gene = new_gene  
        new_mutation.append(gene)  
    return new_mutation
```



GA

- Taxa de mutação: 0.3
 - Taxa de crossover: 0.7
 - Utilizando método da roleta probabilística para seleção parental
-
- Referencia: Computational Methods of Feature Selection, 2008



DE

- Taxa de mutação: 0.3
 - Taxa de Crossover: 0.6
-
- Referência: Referencia: Computational Methods of Feature Selection, 2008
 - Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Space, 1997



PE

- Taxa de mutação: 0.3
- Referência: Referencia: Computational Methods of Feature Selection, 2008
- Fast Evolutionary Programming, 1996



EE

- Taxa de mutação: 0.3
- Taxa de Crossover: 0.2 e 0.7

- Referência: Referencia: Computational Methods of Feature Selection, 2008
- Estratégias evolutivas Aplicadas à Resolução de Otimização Multimodal 1999

EE - Cauchy

- Taxa de mutação: 0.3
- Taxa de Crossover: 0.2
- Utiliza uma série Gaussiana para a realização da mutação

```
new_mutation = []

r1 = (math.sqrt((2*len(hemafrodite))))**-1
r2 = (math.sqrt(math.sqrt((4*len(hemafrodite))))))**-1
#print(r1, r2)
for gene in hemafrodite:
    o = mut * np.exp((r1*random.gauss(0, 1)) + (r2*random.gauss(0, 1)))
    new_gene = gene + (o * np.random.standard_cauchy(1))
```

- Referência: Referência: Computational Methods on Feature Selection, 2008
- Estratégias evolutivas Aplicadas à Resolução de Otimização Multimodal,

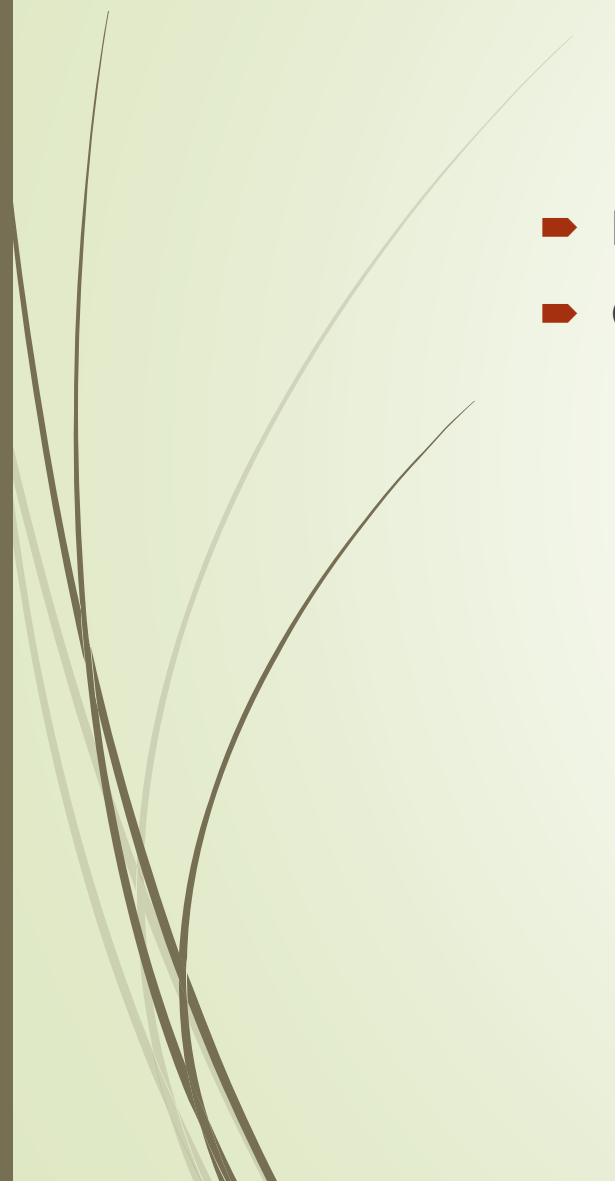


EE – Cauchy + mutação meio à meio

- Taxa de mutação: 0.3
 - Utiliza uma série Gaussiana para a realização da mutação
 - O crossover obrigatoriamente pega metade dos genes da mãe e a outra metade do pai.
-
- Referência: Referencia: Computational Methods of Feature Selection, 2008
 - Estratégias evolutivas Aplicadas à Resolução de Otimização Multimodal,



GSO e PSO

- ▶ Limites Inferiores e Superiores (0 – 1) GSO e PSO
 - ▶ $C1$ e $C2 = 2$, (PSO)
- 

Resultados

Média, Desvio Padrão e Mediana do Tempo de Execução			
Algoritmo	Média	Desvio Padrão	Mediana
Algoritmo Genético	0.642	0.022	0.65
Estratégias Evolucionárias	0.706	0.114	0.75
Evolução Diferencial	0.735	0.053	0.73
Programação Evolutiva	0.622	0.081	0.64
Estratégias Evolucionárias - C	0.642	0.034	0.65
Estratégias Evolucionárias -CC	0.6427	0.034	0.65
Random Forrest	0.6154	0.094	0.65
GSO	0.746	0.056	0.77
PSO	0.709	0.064	0.69

Resultados

Comparação		
Comparação	t-value	p-value
Normal e GA	-0.899	0.3790
Normal e EP	-0.193	0.848
Normal e EE	-2.0340	0.0554
Normal e DE	-3.667	0.0015
Normal e EE_Cauchy	-0.899	0.379
Normal e EE_Cauchy2	-0.899	0.3790
Normal e GSO	-4.609	9.414
Normal e PSO	-2.625	0.0166
GA e EP	0.746	0.4642
GA e EE	-1.764	0.8848
GA e DE	-4.797	0.6047
GA e ee_cauchy	0.0	1.0
GA e ee_cauchy2	0.0	1.0
GA e GSO	0.7461	0.4642
GA e PSO	-1.764	0.0929
EP e EE	0.1510	0.8848

Comparação		
Comparação	t-value	p-value
EP e DE	-0.5460	0.6047
EP e ee_cauchy	-0.746	0.464
EP e GSO	-47.427	6.6278
EP e PSO	-2.664	0.0153
EE e ee_cauchy2	1.764	0.092
EE e DE	-0.6632	0.5318
EE e GSO	-1.238	0.226
EE e PSO	-0.0641	0.9495
DE e ee_cauchy	4.7976	0.0001
DE e ee_cauchy2	4.7976	0.0001
DE e GSO	0.512	0.612
DE e PSO	1.0244	0.3184
GSO e PSO	1.576	0.1273



Dificuldades

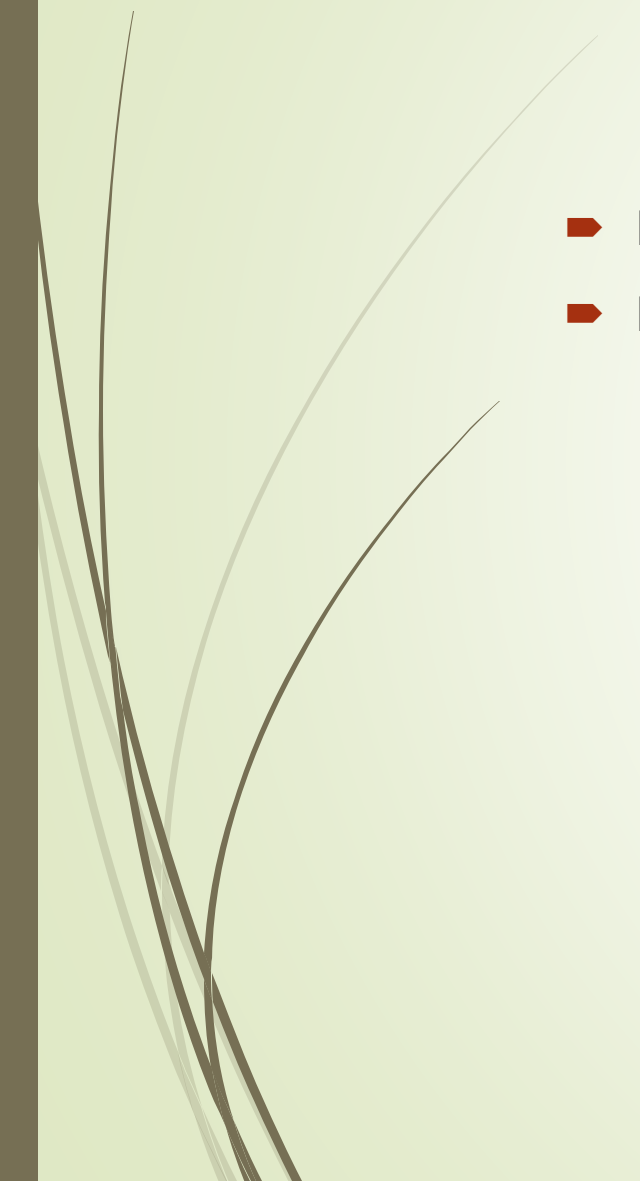
- ▶ Demora para rodar os experimentos

Num vai dar tempo...





Projeto IV

- ▀ Enxame de Abelhas (Artificial Bee Colony)
 - ▀ FireFly (Vagalumes)
- 



Referências



- A. Freitas, "A Review of Evolutionary Algorithms for Data Mining".
- Tang et al., "Dynamic group optimisation algorithm for training feed-forward neural networks"
- Alejandro et al., "Evolutionary optimization of convolutional neural networks for cancer miRNA biomarkers classification"
- Rasim et al., "Evolutionary Algorithm for Extractive Text Summarization"
- Sebastiani. Fabrizio, "Machine Learning in Automated Text Categorization"
- Black et al., "Evolutionary Computation: Comments on the History and Current State"
- Atkinson et al., "Combining Information Extraction with Genetic Algorithms for Text Mining"
- Tharwat et al., "Recognizing human activity in mobile crowdsensing environment using optimized k-NN algorithm"
- Abraham et al., "Document Clustering Using Differential Evolution"
- Evolutionary Computation Algorithms for Feature Selection of EEG-based Emotion Recognition using Mobile Sensors, 2017
- Group Search Optimizer: An Optimization Algorithm Inspired by Animal Searching Behavior, 2009
- Particle swarm optimization (**PSO**), 1995

