UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

ALEXANDRE ROSSETO LEMOS

TRABALHO COMPUTACIONAL 1

COMPUTAÇÃO NATURAL

Vitória
Março 2022

Objetivo

A solução da tarefa compreende a implementação dos algoritmos DE e ES e suas respectivas versões hibridizadas com *K-Mean*s para clusterização de dados para 3 benchmarks: Iris, wine, e breast cancer.

Os algoritmos DE e ES são utilizadas para otimização de funções. No caso deste projeto, eles foram utilizados para minimizar a função de soma das diferenças quadradas (SSE), dada pela equação abaixo.

$$SSE(X, C) = \sum_{i=1}^{K} \sum_{x_j \in C_i} \|x_j - m_i\|^2$$

Onde $X = \{x_1, x_2, ..., x_N\}$ é o conjunto de N amostras presente no conjunto de dados, $C = \{C_1, C_2, ..., C_K\}$ é o conjunto de k *clusters*, $\| \cdot \|$ é a distância euclidiana e m_i é o centroide do *cluster* C_i.

Algoritmos utilizados

No desenvolvimento do projeto foram utilizados quatro algoritmos, que serão explicados a seguir, sendo eles:

- *K-Means*
- *K-Means* com *Differential Evolution*
- *K-Means* com *Evolution Strategies*

K-Means

No algoritmo K-Means desenvolvido, primeiramente inicializa-se aleatoriamente k centroides cujas coordenadas são limitadas pelos valores máximos e mínimos de cada variável do conjunto de dados. Em seguida, para cada amostra, calcula-se a distância euclidiana entre ela e os k centroides, e adiciona-se a amostra ao *cluster* cujo centroide encontra-se mais próximo. Após todas as amostras pertencerem a um conjunto, os centroides dos *clusters* são substituídos pela média de todas as amostras pertencentes ao *cluster*, e recalcula-se os *clusters* novamente. Esses procedimentos são repetidos até que não haja mais mudanças nos grupos gerados.

K-Means com *Differential Evolution*

Para o algoritmo *K-Means* com *Differential Evolution*, utilizou-se o *K-Means* inicialmente para gerar os *clusters* e centroides referentes aos conjuntos de dados, que são utilizados pelo DE como população inicial. Em seguida, define-se os limites os quais o algoritmo DE irá buscar a melhor solução (mínimo e máximo de cada variável do conjunto de dados referente às coordenadas do centroide). Foi utilizada a biblioteca *scipy-optimize* para execução do algoritmo DE. Com os centroides inicias já obtidos (vetor inicial), realiza-se primeiramente uma operação de mutação onde, para cada elemento do vetor, calcula-se aleatoriamente um número entre 0 e 1, caso esse número supere um limite (fator de *cross-over*), esse elemento será substituído utilizando a fórmula abaixo.

$$y'_{i,G} = u_{a,G} + F * (u_{b,G} - u_{c,G})$$

Onde $y'_{i,G}$ é o chamado de vetor mutante, $u_{a,G}$, $u_{b,G}$ e $u_{c,G}$ são três elementos aleatórios (diferentes entre si) do conjunto de dados e F é o fator de amplificação (geralmente um número entre 0 e 1). Em seguida, ocorre a operação de *cross-over*, onde cada elemento do vetor solução final será calculado a partir da fórmula abaixo.

$$y_{ji.G} = \begin{cases} y'_{ji,G} \; se \; r(j) < CR \; ou \; j = d \\ u_{ji,G} \qquad caso \; contrário \end{cases}$$

Onde *r(j)* é um número aleatório entre 0 e 1 calculado para cada *j*, *CR* é o fator de *cross-over* e *d* é um índex aleatório para garantir que pelo menos um elemento do virá do vetor mutante $y'_{i,G}$. Tanto a variável *CR* quanto a variável *F* são fornecidas pelo usuário. Em seguida, o valor da função objetivo é calculado utilizando o vetor solução $y'_{i,G}$, se o resultado obtido for melhor que o resultado da solução original $u_{i,G}$, o vetor sobrevive até a nova iteração, se tornando $u_{i,G+1}$, caso contrário $u_{i,G+1}$ se torna $u_{i,G}$.

K-Means com *Evolution Strategies*

Para o algoritmo *K-Means* com *Evolution Strategies* foi implementado a versão onde substitui-se os pais pelos filhos (mu, lambda)-ES. Primeiramente, inicia-se com o *K-Means* gerando *clusters* e centroides que servirão de população inicial para o ES. Semelhante ao algoritmo DE, define-se os mesmos limites de busca para o ES. Com a população inicial obtida através do *K-Means*, calcula-se o SSE para cada elemento da população. Em seguida, ordena-se as possíveis soluções baseados no SSE obtido, como o algoritmo busca minimizar a função objetivo, a melhor solução é aquela com o menor SSE. O próximo passo é a escolha dos pais, onde o número de pais é a variável *mu*, e a criação dos filhos, onde o número de filhos por cada pai é definido pela divisão *lambda/um*, onde *lambda* é o tamanho da população. Os filhos são gerados a partir dos pais somados a uma distribuição Gaussiana cuja média e desvio padrão são fornecidos pela variável *step_size*. Se cada pai selecionado for a melhor solução já encontrada, encerra-se a iteração do algoritmo. Os filhos gerados substituem a população na próxima iteração. O algoritmo irá executar até que um determinado número de iterações, fornecido pelo usuário, tenha ocorrido.

Dados utilizados

Os dados necessários para o desenvolvimento do projeto (Iris, Wine e Breast Cancer presentes no repositório *UCI Machine Learning Repository* da Universidade da Califórnia Irvine) foram obtidos através da biblioteca *scikit-learn*. Cada conjunto de dados possui características diferentes que serão explicitadas nesta sessão.

O conjunto de dados Iris possui informações a respeito de diferentes plantas do gênero Iris, com 150 amostras, quatro colunas de características das pétalas e uma coluna de rótulo que informa qual subgrupo a amostra pertence (existem três possíveis categorias). O conjunto de dados possui 50 amostras em cada uma das três classes, sendo totalmente balanceado.

O conjunto de dados Breast Cancer contém informações a respeito de imagens dos núcleos celulares extraídos de pacientes, como raio, textura, perímetro e área, por exemplo. Também possui campo de identificação do paciente e uma coluna de rótulo que informa se o tumor é benigno ou maligno (duas classes). O conjunto de dados possui 357 amostras pertencentes à classe de câncer benigno e 212 amostras pertencentes à classe de câncer maligno.

O conjunto de dados Wine contém informações sobre análises químicas realizadas em 178 vinhos da mesma região, porém de três localidades diferentes (três classes). Ele possui características como concentração de álcool, magnésio e intensidade da cor do vinho, por exemplo. O conjunto de dados possui 59 amostras pertencentes à classe 1, 71 pertencentes à classe 2 e 48 pertencentes à classe 3.

Desenvolvimento

Primeiramente foi realizada a separação dos dados de cada dataset em conjuntos de treino e teste, utilizando 70% dos dados para treino e 30% para teste. Essa separação foi realizada para que possa se calcular a performance do modelo (treinado com o conjunto de treino) sobre o conjunto de teste. Como métrica de desempenho foi utilizada a acurácia, cuja fórmula é dada seguir:

$$Acurácia = \frac{VP + VN}{VP + FP + VN + FN}$$

Onde *VP*, *VN*, *FP* e *FN* são a quantidade de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos, respectivamente.

O desenvolvimento do projeto foi dividido em três partes. Na primeira parte, foi desenvolvido um algoritmo *K-Means* para servir como *baseline*, utilizando o modelo disponibilizado pela biblioteca *scikit-learn.*

Na segunda parte, foi desenvolvido um modelo híbrido de K-Means que utiliza *Differential Evolution* para o cálculo dos centroides dos clusters. Primeiro foi inicializado uma população cuja quantidade de indivíduos varia seguindo a fórmula: $tamanho_{populacao} = 10 * dimensao_{dados}$. Como parâmetros utilizou-se um fator de mutação de 0,88 (valor entre 0,5 e 1,0, escolhido aleatoriamente), uma taxa de crossover de 0,7 e um número de iterações de 50.

Na terceira parte, utiliza-se o *K-Means* juntamente com o algoritmo (mu, lambda)-*Evolution Strategies*. Novamente, em cada iteração do algoritmo, selecionou-se aleatoriamente um centroide dos valores gerados pelo *K-Means* para servir de população inicial.

Para analisar a robustez dos modelos desenvolvidos e comparar as diferentes técnicas utilizadas, executou-se cada algoritmo 5 vezes. A cada execução, calculou-se o SSE de cada modelo assim como também a acurácia obtida pelo modelo quando executado no conjunto de teste.

Resultados

A seguir, tem-se os resultados obtidos pelos quatro algoritmos desenvolvidos para todos os conjuntos de dados.

Conjunto de dados	Algoritmo	Média	Desvio Padrão	Mediana	Acurácia (%)
Wine	K-Means	**1082,89**	184,34	1058,82	21,48
	K-Means-DE	2212,92	**16,74**	2208,67	**32,22**
	K-Means-(mu, lambda)-ES	3165,15	1050,36	2858,91	28,15
Breast Cancer	K-Means	**11923,76**	5059,09	15618,41	57,66
	K-Means-DE	15467,36	**355,24**	15653,91	35,20
	K-Means-(mu, lambda)-ES	38624,60	24938,87	33401,16	**64,44**
Iris	K-Means	324,54	207,12	259,16	29,33
	K-Means-DE	**317,22**	**15,00**	309,84	30,67
	K-Means-(mu, lambda)-ES	401,17	93,20	379,39	**31,56**

Conclusão

Analisando os resultados, para os dados Wine e Breast Cancer o modelo *baseline* foi o que obteve a menor média de erro, porém, o modelo K-Means-DE foi o mais robusto (menor desvio padrão) entre os três conjuntos de dados e obteve a melhor média para o conjunto Iris. Considerando a acurácia, os modelos híbridos foram superiores ao modelo *baseline*, onde para o conjunto Wine o K-Means-DE obteve a melhor acurácia e para os conjuntos Breast Cancer o Iris o modelo K-Means-ES obteve a melhor acurácia.

Portanto, o modelo que obteve a melhor média foi o *baseline*, o mais robusto foi o K-Means-DE e o que obteve a melhor acurácia foi o K-Means-ES.

É importante ressaltar que os experimentos que utilizaram DE e ES necessitaram de um maior tempo para execução. Porém é provável que exista espaço para melhoria na performance, uma vez que o modelo *baseline* é proveniente de uma biblioteca que recebe contribuições de milhares de desenvolvedores.