UNIVERSIDADE FEDERAL DO TOCANTINS
CAMPUS UNIVERSITÁRIO DE PALMAS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

UTILIZAÇÃO DE ALGORITMOS DE OTIMIZAÇÃO POR ENXAME APLICADOS À SELEÇÃO DE CARACTERÍSTICAS

KLEYSON MORAIS DE SOUSA

PALMAS (TO)
2018

KLEYSON MORAIS DE SOUSA

UTILIZAÇÃO DE ALGORITMOS DE OTIMIZAÇÃO POR ENXAME APLICADOS À SELEÇÃO DE CARACTERÍSTICAS

Trabalho de Conclusão de Curso II apresentado à Universidade Federal do Tocantins para obtenção do título de Bacharel em Ciência da Computação, sob a orinetação do(a) Prof.(a) Dr. Rafael Lima de Carvalho.

Orientador: Dr. Rafael Lima de Carvalho

PALMAS (TO)

2018

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da Universidade Federal do Tocantins

S725u Sousa, Kleyson Morais de.

Utilização de algoritmos de otimização por enxame aplicados à seleção de características. / Kleyson Morais de Sousa. – Palmas, TO, 2018.

52 f.

Monografia Graduação - Universidade Federal do Tocantins – Câmpus Universitário de Palmas - Curso de Ciências da Computação, 2018.

Orientador: Rafael Lima de Carvalho

1. Seleção de Características. 2. Otimização. 3. Aprendizagem de Máquina. 4. Enxame de Partículas. I. Título

CDD 004

RESUMO

A pesquisa por métodos de seleção de características tem se mostrado cada vez mais presente em aplicações de aprendizado de máquina, principalmente naquelas onde o número de atributos disponíveis está na faixa de centenas ou até mesmo milhares. Essas aplicações incluem, por exemplo, o processamento de documentos de texto, análise de expressão genética e química combinatória. Seleção de características ou variáveis é um conceito que propõe métodos que visam fornecer preditores mais rápidos e econômicos, melhorar o desempenho de previsão dos preditores e proporcionar uma melhor compreensão do processo subjacente que gerou os dados. Matematicamente, a seleção de características é formulada como um problema de otimização combinatória. Em geral, abordar problemas deste tipo de maneira a encontrar a solução exata nem sempre é viável. Dessa forma, métodos de inteligência computacional, podem ser usados para que permitam realizar a seleção de características na prática. Portanto, o objetivo deste trabalho é apresentar e propor técnicas de otimização guiadas por estratégias de seleção de características, dentre as quais pode-se destacar a otimização por enxame de partículas, otimização de enxame por competição e a combinação de ambos.

Palavra-chave: Seleção de Características. Otimização. Aprendizagem de Máquina. Enxame de Partículas.

ABSTRACT

The search for feature selection methods has been increasingly present in machine learning applications, especially in those where the number of available attributes is in the range of hundreds or even thousands. Such applications include, for example, word document processing, gene expression analysis, and combinatorial chemistry. Feature selection or selection of characteristics is a concept that proposes methods that aim to provide faster and more economical predictors, improve predictor prediction performance, and provide a better understanding of the underlying process that generated the data. Mathematically, feature selection is formulated as a combinatorial optimization problem. In general, addressing such problems in a way that finding the exact solution is not always feasible. In this way, computational intelligence methods can be used to allow the feature selection in practice. Therefore, the objective of this work is to present and propose optimization techniques guided by strategies of feature selection, among which we can highlight the optimization by swarm of particles, optimization of swarm by competition and the combination of both.

Keywords: Feature Selection. Optimization. Machine Learning. Swarm of Particles.

ABREVIATURAS

AM Aprendizado de Máquina

CSO *Competitive Swarm Optimizer*

IA Inteligência Artificial

NB *Naive Bayes*

PSO *Particle Swarm Optimization*

SC Seleção de Características

LISTA DE FIGURAS

Figura 2.1 – Representação do Mecanismo de Busca. 18
Figura 2.2 – Seleção de características utilizando filtros. 22
Figura 2.3 – Seleção de características usando *wrappers* 23

Figura 3.1 – Modelo de aprendizado das partículas no PSO. 26
Figura 3.2 – Modelo de aprendizagem das partículas no CSO. 28
Figura 3.3 – Modelo de aprendizagem do algoritmo híbrido proposto. 30

Figura 4.1 – Seleção de características utilizando o PSO para geração de subconjunto. 36
Figura 4.2 – Seleção de características utilizando o CSO para geração de subconjunto. 37
Figura 4.3 – Seleção de características utilizando proposta de algoritmo híbrido. 38

Figura 5.1 – Comparação dos resultados de *fitness* obtidos com os algoritmos para a base *Wine*. 42
Figura 5.2 – Comparação do número de novas soluções obtidas com os algoritmos para a base *Wine*. 43
Figura 5.3 – Comparação do número de características selecionadas com os algoritmos para a base *Wine*. 43
Figura 5.4 – Comparação dos resultados de *fitness* obtidos com os algoritmos para a base *Cancer*. 44
Figura 5.5 – Comparação do número de novas soluções obtidas com os algoritmos para a base *Cancer*. 44
Figura 5.6 – Comparação do número de características selecionadas com os algoritmos para a base *Cancer*. 45
Figura 5.7 – Comparação dos resultados de *fitness* obtidos com os algoritmos para a base *Ionosphere*. 45

Figura 5.8 – Comparação do número de novas soluções obtidas com os algoritmos para a base *Ionosphere*. 46
Figura 5.9 – Comparação do número de características selecionadas com os algoritmos para a base *Ionosphere*. 46
Figura 5.10 – Comparação dos resultados de *fitness* obtidos com os algoritmos para a base *Sonar*. 47
Figura 5.11 – Comparação do número de novas soluções obtidas com os algoritmos para a base *Sonar*. 47
Figura 5.12 – Comparação do número de características selecionadas com os algoritmos para a base *Sonar*. 48
Figura 5.13 – Comparação dos resultados de *fitness* obtidos com os algoritmos para a base *Madelon*. 48
Figura 5.14 – Comparação do número de novas soluções obtidas com os algoritmos para a base *Madelon*. 49
Figura 5.15 – Comparação do número de características selecionadas com os algoritmos para a base *Madelon*. 49
Figura 5.16 – Quantidade de Características Selecionadas para as bases *Wine*, *Breast-Cacer*, *Ionosphere* e *Sonar*. 51
Figura 5.17 – Quantidade de Características Selecionadas para a base *Madelon*. . 51

LISTA DE TABELAS

Tabela 4.1 – Detalhes da base de dados *Wine*. 39
Tabela 4.2 – Detalhes da base de dados *Breast-cancer*. 39
Tabela 4.3 – Detalhes da base de dados *Ionophere*. 40
Tabela 4.4 – Detalhes da base de dados *Sonar*. 40
Tabela 4.5 – Detalhes da base de dados *Madelon*. 41

Tabela 5.1 – Comparativo da Taxa de Erro Encontrada pelos Métodos 50

SUMÁRIO

1 INTRODUÇÃO 14

1.1 Objetivos 15

1.1.1 Objetivos Específicos 15

1.2 Descrição do Trabalho 16

2 SELEÇÃO DE CARACTERíSTICAS 17

2.1 O Problema da Alta Dimensionalidade 17

2.2 Principais Conceitos 18

2.2.1 Ponto(s) de Partida da Busca e Geração dos Subconjuntos 19

2.2.2 Estratégia de Busca 19

2.2.3 Estratégia de Avaliação 20

2.2.4 Critério de Parada 20

2.3 Filtros 21

2.4 *Wrappers* 22

3 ALGORITMOS UTILIZADOS 24

3.1 Algoritmos de *Wrappers* (Classificadores) 24

3.1.1 Naive Bayes 24

3.2 Algoritmos de Otimização 25

3.2.1 Otimização por Enxame de Partículas (PSO) 25

3.2.2 Otimização de Enxame por Competição (CSO) 27

3.2.3 Proposta de Algoritmo Híbrido 30

4 METODOLOGIA 32

4.1 Considerações Iniciais 32

4.1.1 Sobre *Python* 32

4.1.1.1 *Scikit-learn* 32
4.1.2 Acurácia 32
4.1.3 *F1 Score* 33
4.1.4 *Holdout* 33
4.2 Esquematização dos Algoritmos Utilizados 34
4.2.1 Implementação do PSO 34
4.2.2 Implementação do CSO 36
4.2.3 Implementação da Proposta de Algoritmo Híbrido 37
4.2.4 Avaliação do Subconjunto com o Naive Bayes 38
4.3 Base de Dados 39
5 RESULTADOS E DISCUSSÕES 42
5.1 Redução da Taxa de Erro 49
5.2 Seleção de Características 50
6 CONSIDERAÇÕES FINAIS 52
REFERÊNCIAS 53

1 INTRODUÇÃO

A utilização do aprendizado de máquina vem sendo cada vez mais explorada nas mais diversas áreas devido ao seu alto potencial de produção de conhecimento. Dois exemplos servem como base para ilustração. O primeiro é a seleção de genes a partir de dados de *microarrays* e o segundo é a categorização de texto (GUYON; ELISSEEFF, 2003).

Na Bioinformática, experimentos com *microarrays* de DNA gerando milhares de medições de expressão gênica são usados para reunir dados de amostras de tecido e célula em relação a diferenças de expressão gênica que podem ser úteis para diagnóstico de doenças, distinção de um tipo específico de tumor, etc. (INZA et al., 2004). Estudos mostraram a eficácia da utilização de algoritmos de aprendizagem de máquinas na classificação de doenças severas, como o caso da doença de Alzheimer (WALKER et al., 2004). A partir de estudos como este, pode-se tentar definir estratégias que evitem o agravamento de um problema a uma pessoa que seja suscetível à doença.

No caso da categorização de texto, documentos podem ser representados por um *bag-of-words*, que é um vetor de dimensão do tamanho do vocabulário que contém contagens de frequência de cada palavras. Vocabulários com centenas de milhares de palavras são comuns. Ao aplicar algoritmos de aprendizagem de máquina sobe um *bag-of-words*, pode-se obter, por exemplo, a análise de sentimentos, classificação automática de documentos e a detecção de e-mail não solicitados (*spam*).

Em comum, aplicações para a aprendizagem de máquina como estas citadas, requerem um custo de processamento significativo, dado o elevado número de características a serem considerados para a tarefa. Entende-se assim, que minimizar o número de características é primordial para uma maior qualidade dos resultados e para a compreensão do fenômeno considerado (CHANDRASHEKAR; SAHIN, 2014).

Para reduzir o número de características a serem utilizados em um algoritmo, com um mínimo de perda de informação, pode-se utilizar métodos de redução de dimensionalidade. Há, basicamente, duas abordagens para isso: a extração de características e a seleção de características. A primeira realiza uma série de combinações e/ou transformações das características existentes para gerarem outras novas. A segunda consiste na utilização de métodos e técnicas para selecionar um subconjunto otimizado de características, segundo algum critério pré-estabelecido. No caso dos exemplos citados, para os dados de expressão gênica, esse subconjunto seria composto apenas pelos genes considerados mais relevantes, reduzindo consideravelmente a dimensionalidade. Já para a categorização de textos, poderia ser um subconjunto que excluísse as palavras consideradas irrelevantes para o *bag-of-words*, permitindo assim também, uma diminuição da dimensionalidade.

Dessa forma, os métodos de seleção de características atuam buscando distinguir um subconjunto de características que possa representar satisfatoriamente os dados de entrada enquanto minimiza os efeitos do ruído ou características irrelevantes e ainda fornecer bons resultados de predição (CHANDRASHEKAR; SAHIN, 2014).

Blum e Langley (1997) dividiram o aprendizado de máquinas em duas subtarefas: decidir quais características usar para descrever o conceito e decidir como combinar essas características. Segundo o mesmo, a seleção de características relevantes e a eliminação de irrelevantes, é um dos problemas centrais na aprendizagem de máquina, e muitos algoritmos de predição incorporam alguma forma de tratamento para abordá-la.

Não obstante, Kohavi e John (1997) apontam que o principal problema da seleção de características é o de encontrar um subconjunto das características originais de um determinado conjunto de dados, de modo que um algoritmo de predição que é executado sob estes dados contenha apenas as características que gerem um classificador com a melhor taxa de avaliação possível. Reforça ainda, que a seleção de características escolhe um conjunto de características existentes e não cria novas, ou seja, não há construção de características, mas sim exclusão de características irrelevantes.

1.1 Objetivos

Este trabalho tem como finalidade investigar técnicas de seleção de características. Dentre as quais, pode-se citar a utilização dos método de busca de otimização por enxame de partículas e otimização de enxame por competição guiados pelo classificador Naive Bayes, bem como pela proposta de apresentar um algoritmo híbrido que explore as duas abordagens de otimização.

1.1.1 Objetivos Específicos

- Definir a abordagem de seleção de características.
- Apresentar alguns dos principais métodos de seleção de características.
- Conceituar algoritmos de otimização por enxame de partícula e otimização de enxame por competição.
- Conceituar classificadores.
- Descrever especificidades de implementação dos algoritmos.
- Aplicar os algoritmos de seleção de atributos em bases de dados.
- Interpretar e analisar os resultados de predição.
- Comparar os resultados obtidos pelos diferentes métodos utilizados.

1.2 Descrição do Trabalho

O trabalho está organizado da seguinte forma: no Capítulo 2 está exposto os principais conceitos e métodos da abordagem de seleção de características, o Capítulo 3 aborda as principais fundamentações dos algoritmos utilizados neste trabalho, no Capítulo 4 estão os tópicos referentes ao desenvolvimento e implementação dos algoritmos e, por fim, no Capítulo 5 estão descritos as análises e interpretações dos principais resultados encontrados.

2 SELEÇÃO DE CARACTERÍSTICAS

2.1 O Problema da Alta Dimensionalidade

Segundo Monard e Baranauskas (2003), "Aprendizado de Máquina é uma área de IA cujo o objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado, bem como a construção de sistemas capazes de adquirir conhecimento de forma automática". Um sistema de aprendizado é um algoritmo que toma decisões baseadas em experiências acumuladas, que podem ser abstraídas como sendo vetores de características do fenômeno abordado, buscando aprender uma função de mapeamento entre os exemplos de entrada e suas saídas desejadas. Quando este processo é bem sucedido, um classificador de indução é criado.

Intuitivamente, o senso comum acredita que quanto mais características disponíveis são utilizadas para a construção do classificador de indução, mais precisa e representativa será a expressão obtida do fenômeno abordado. No entanto, a prática sugere que trabalhar com uma quantidade reduzida de característica pode ser benéfico em diversos aspectos (SOUZA, 2005).

Inicialmente, observa-se que quanto maior a quantidade de características utilizadas, maior será o processamento de casos testes na construção de um classificador indutor de bom desempenho. Esse paradigma, é conhecido como maldição da dimensionalidade (CHEN; MONTGOMERY; BOLUFÉ-RÖHLER, 2015). Em algumas situações, a quantidade necessária de testes a serem executados, pode ser exponencial em relação à quantidade de características (CHANDRASHEKAR; SAHIN, 2014). Devido a isso, em muitos casos um subconjunto do conjunto total de características pode apresentar um melhor desempenho do classificador.

Não obstante, em alguns casos, objetiva-se selecionar uma série de características apenas para visualização e/ou compreensão dos dados. Facultando que, a construção de uma classificador, não seja prioridade. Um exemplo disso são as análises do mercado financeiro. A identificação das variáveis (características) que atuam diretamente sob um cenário econômico em questão é de grande interesse prático. Economistas e investidores podem examinar as variáveis de mercado selecionadas para elaborar estratégias, planejamento de vendas etc..

Outo aspecto que torna a seleção de características uma ação benéfica, está na etapa de construção de um classificador. Nesta etapa, é comum que certas características não sejam relevantes para o desenvolvimento do mesmo (KOHAVI; JOHN, 1997). Considerando um exemplo bem simplório, observe o caso de um sistema de aprendizagem que busca gerar um classificador a partir de um banco de dados relacional, estes costumam ter em sua estrutura a característica *ID*, que trata-se apenas de um valor que representa

o índice de uma determinada instância, intuitivamente, nota-se que a remoção dessa característica não afetará o desempenho do classificador, pelo contrário, esta remoção pode, em alguns casos, ser benéfica ao mesmo. Em outras palavras, percebe-se que não se trata de uma característica tão relevante para a construção do classificador. Assim atributos ou características como estas, podem ser irrelevantes ou redundantes ou apresentarem níveis elevados de ruído. Dessa forma, o aprendizado pode ser realizado mais efetivamente desconsiderando-se tais atributos. Vale ressaltar também, que o uso de mais atributos implica em maior custo de aquisição e análise de dados.

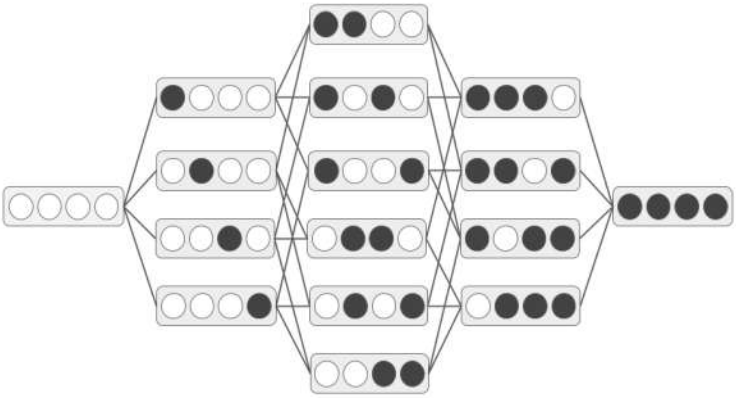
Os métodos de Seleção de Características (SC) estão entre os mais empregados, quando o objetivo é a redução da dimensionalidade. Sendo que, o princípio base, que rege a SC segundo Souza (2005) é, "dado um conjunto de n características, selecionar um subconjunto de tamanho k tal que o erro de classificação tenha a maior redução possível em relação ao conjunto completo".

A seguir, este princípio será melhor abordado, apresentando os principais conceitos dos algoritmos de SC encontrados na literatura.

2.2 Principais Conceitos

Com a necessidade de selecionar características, dentro de um conjunto de possibilidades, o conceito de seleção passou a ser entendido como um problema de busca. Assim, toda combinação possível de características, estaria no espaço da busca. Considere o exemplo apresentado na Figura 2.1, onde um grafo é gerado com todas as possibilidades para quatro características/atributos. Os círculos com cores claras representam atributos não selecionados e os círculos com cores escuras são atributos selecionados.

Figura 2.1 – Representação do Mecanismo de Busca. Adaptada de (BLUM; LANGLEY, 1997)



Percebe-se com isso, que algumas estratégias devem ser levadas em consideração para que haja uma seleção de características. Blum e Langley (1997) apresentaram alguns

dos principais conceitos para trabalhar com a busca de subconjuntos. São eles: ponto(s) de partida da busca e geração dos subconjuntos, estratégia da busca, estratégia de avaliação e o critério de parada.

2.2.1 Ponto(s) de Partida da Busca e Geração dos Subconjuntos

Motoda e Liu (2002) ressaltaram as seguintes formas de geração de subconjuntos:

1. **Geração para trás**: Esta abordagem deve ser inicializada com todas as características para que em seguida novas combinações sejam geradas a partir da remoção das características, com um decremento uniforme das características.

2. **Geração adiante**: Esta abordagem deve inicializar um conjunto vazio de características para que em seguida novas combinações sejam geradas a partir da inserção das características, com um incremento uniforme das características.

3. **Geração bidirecional**: A busca tem parte tanto pelo conjunto vazio como pelo conjunto com todas as características, fazendo inserção e remoção de características, de acordo com as extremidades do grafo.

4. **Geração randômica**: A busca nesta abordagem é destacada por escolhas aleatórias do ponto de partida e da decisão de remover ou adicionar as características.

2.2.2 Estratégia de Busca

Após a definição do ponto de partida da busca e a forma como os subconjuntos serão gerados, é necessário definir que algoritmos utilizar para percorrer o grafo. Motoda e Liu (2002) resumiram os algoritmos nas seguintes categorias:

1. **Busca não-determinística**. Este sistema de busca, é compreendido como sendo um processo estocástico. É a contraparte da busca determinística, em que os nós a serem visitados já são conhecidos. Por se tratar de uma busca randômica, essa abordagem possui duas características principais: não é necessário esperar até que termine a busca para se encontrar uma solução satisfatória e não é possível saber quando o subconjunto ótimo vai aparecer.

2. **Busca completa**. Compreende-se por busca completa, o algoritmo que consegue avaliar todas as possíveis combinações de características, nos subconjuntos ótimos, caso haja mais de um. Dessa forma, entende-se a princípio que a única forma de fazer isso é através de uma busca exaustiva, visitando todos os nós.

3. **Busca heurística**. Em síntese, entende-se uma heurística como sendo um conjunto de regras e métodos que conduzem à resolução de um problema, utilizando

a seu favor informação e intuição a respeito da instância do problema e da sua estrutura para resolvê-lo de forma rápida. Com essa abordagem, a utilização de heurísticas pode reduzir significativamente a complexidade da busca, e ainda assim ter bons resultados. No entanto, diferente da abordagem anterior, a implementação de heurísticas em algoritmos de busca não pode garantir que o resultado encontrado seja um resultado ótimo, embora geralmente a utilização de heurísticas apresentem resultados satisfatórios.

2.2.3 Estratégia de Avaliação

Há várias definições divergentes na literatura de SC que buscam conceituar o que torna uma variável "relevante". O motivo dessa variedade é que geralmente depende da questão: "relevante para o quê?". O consenso, no entanto, é que diferentes definições podem ser mais apropriadas dependendo dos objetivos. Nessa perspectiva, a forma como os subconjuntos gerados serão avaliados é de extrema importância, visto que, optar por uma métrica inconsistente com o modelo de dados, pode levar à geração de um subconjunto aquém do esperado. Motoda e Liu (2002) destacaram duas categorias de critérios de avaliação.

1. **Independente**. Os critérios independentes tentam avaliar o quão importante é uma característica ou um subconjunto de características, sem necessariamente aplicar algum algoritmo de aprendizagem de máquina na implementação. Geralmente verificando algum grau de similaridade entre as características, como por exemplo, o quão uniformes são duas características.

2. **Dependente**. Os critérios dependentes utilizam algoritmos de aprendizagem como estratégia de avaliação, assim cada vez que um novo subconjunto é gerado, ele é testado no algoritmo escolhido para que haja um mérito de avaliação em relação ao subconjunto de proposto.

2.2.4 Critério de Parada

Exceto para a busca completa, onde todos os nós serão visitados, é necessário definir um critério de parada. Alguns dos critérios sugeridos são (SOUZA, 2005):

1. Definir previamente um número de características a ser atingido.

2. Parar quando, após suscetivas iterações, o algoritmo não apresentar nenhuma avaliação melhor, mesmo após realizar mais inserções e/ou remoções de características do subconjunto.

3. Definir previamente um número de geração de subconjuntos a ser atingido.

4. Parar quando todas as possibilidades forem testadas.

2.3 Filtros

As primeiras abordagens de seleção de características no aprendizado de máquina foram os métodos de filtro. Em vez de um algoritmo de aprendizado para avaliar o mérito dos subconjuntos de características, os métodos de filtro usam heurísticas com base nos traços gerais dos dados, possibilitando assim, a filtragem de características irrelevantes e/ou redundantes. Como consequência, os métodos de filtro são geralmente muito mais rápidos que os métodos de *wrappers* (Seção 2.4) e, como tal, são mais práticos para redução da dimensionalidade (HALL, 1999).

Dentre as diversas definições de relevância para uma característica, a mais famosa se deve a John, Kohavi e Pfleger (1994). Eles conceituaram dois graus de relevância, são eles: características que são fortemente relevantes e características que são fracamente relevantes. Assumindo uma característica x_i, ela será considerada fortemente relevante se dado um conjunto completo de características S, haja alguma alteração na distribuição de probabilidade dos valores das classes quando x_i for excluída. A característica x_i será fracamente relevante quando ela não atender o primeiro grau, ou seja, não ser fortemente relevante e quando, dado um subconjunto de características $s(s \subset S)$, do qual x_i é membro, haja alguma alteração na distribuição de probabilidade dos valores das classes quando x_i for excluída. Com isso, aquelas características que não estão em um destes dois graus, são consideradas irrelevantes.

Para maior clareza sobre como os métodos de filtro trabalham, observe a Figura 2.2. Ao se trabalhar com conjunto de dados, visando aplicar aprendizagem, é comum que haja uma divisão do conjunto de dados em dois grupos: conjunto de teste e conjunto de treinamento. Assim, seguindo os conceitos apresentados na Seção 2.2, os métodos de filtro terão a seguinte estratégia, utilizando os exemplos do conjunto de treinamento:

1. A busca é iniciada seguindo algum critério pré-estabelecido, gerando um subconjunto de características.

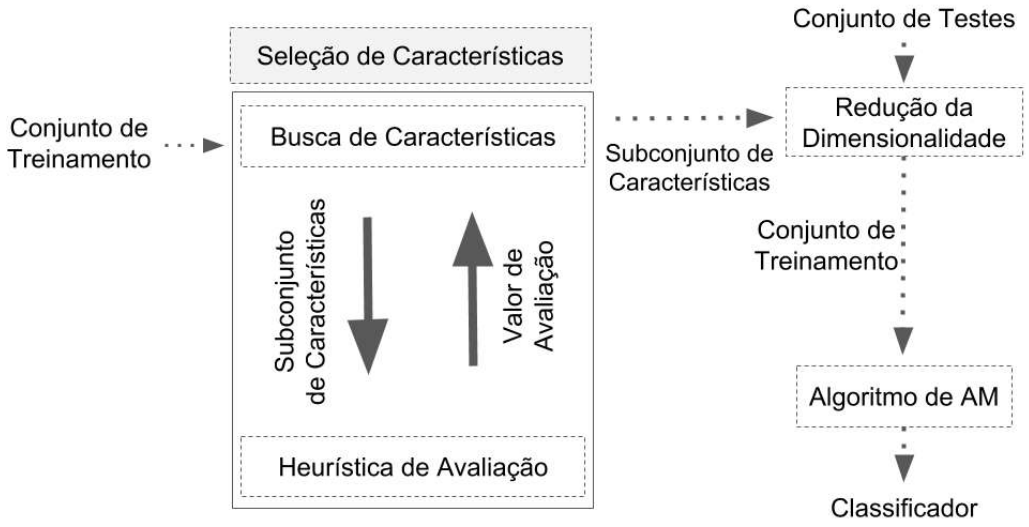
2. É aplicada alguma heurística, referente ao tipo de filtro que está utilizando-se, sob o subconjunto selecionado pelo passo anterior.

3. Um valor de avaliação é gerado, ou seja, um mérito que reflita o quão bom o é subconjunto selecionado.

4. Com algum critério de parada estabelecido, é verificado se o mérito encontrado na etapa anterior atende à este critério, se o critério é satisfeito, então o subconjunto é selecionado e as características não inclusas serão consideradas irrelevantes e/ou redundantes e consequentemente serão removidas.

Em relação às heurísticas utilizadas, os filtros costumam ser distintos de duas maneiras (SOUZA, 2005):

Figura 2.2 – Seleção de características utilizando filtros. Figura adapada (SOUZA, 2005).



1. Filtros que avaliam cada característica isoladamente. Geralmente essa categoria trabalha com um *ranking* das características, identificado as melhores características para o modelo e acusando aquelas que são irrelevantes e/ou redundantes.

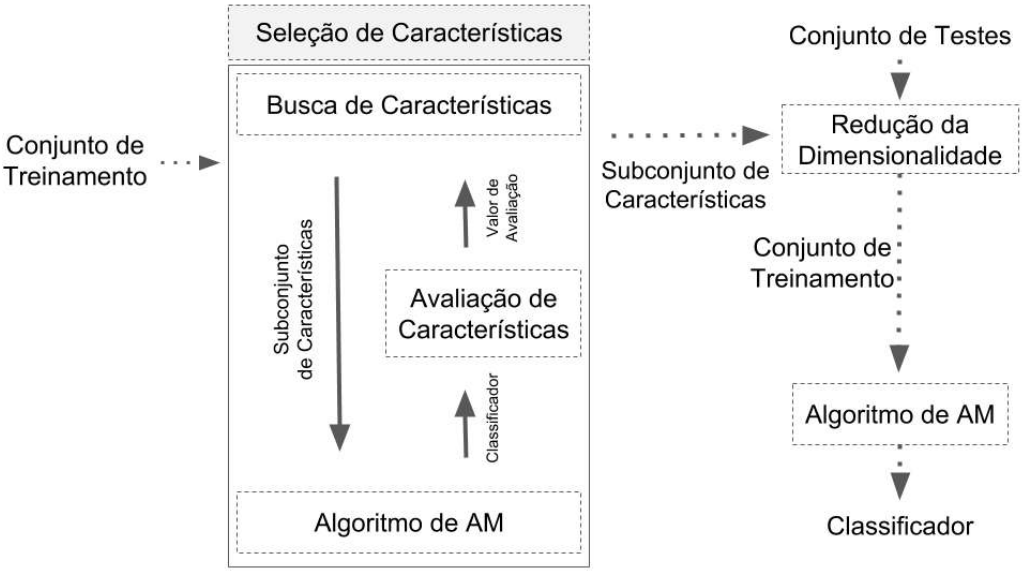
2. Filtros que avaliam subconjuntos de características. Essa categoria considera as características em interação com outras. Isto corrobora para encontrar subconjuntos bons como um todo, mas que não tenham obrigatoriamente características com boas posições se estas fossem avaliadas isoladamente.

2.4 *Wrappers*

No problema de seleção de características, um algoritmo de aprendizado é confrontado com o problema de selecionar algum subconjunto de características para focar sua atenção, enquanto ignora o restante. Na abordagem *wrapper*, o algoritmo de seleção de subconjunto de características age como uma cápsula em torno do algoritmo de classificação. Assim, este algoritmo realiza uma busca, utilizando os conceitos apresentados na Seção 2.2, por um bom subconjunto usando o próprio algoritmo de classificação como parte da função de avaliação (KOHAVI; JOHN, 1997).

O modelo esquemático da abordagem *wrappers* é mostrado na Figura 2.3. A ideia básica desta abordagem é simples: os métodos *wrappers* são considerados como uma caixa preta. Visto que, não é de interesse do método saber qual algoritmo de AM e qual o critério avaliador das características estão sendo aplicados, contanto que, dada uma entrada de um subconjunto de características ao algoritmo de AM (encapsulado), haja o retorno de um classificador referente ao subconjunto aplicado, esse classificador será usado para

Figura 2.3 – Seleção de características usando *wrappers*. Figura adaptada (SOUZA, 2005)



.

avaliação das características presentes no subconjunto utilizado e, finalmente, um valor de avaliação será gerado (KOHAVI; JOHN, 1997).

3 ALGORITMOS UTILIZADOS

Este Capítulo apresenta a parte conceitual dos algoritmos abordados no desenvolvimento deste trabalho. O Capítulo em questão, destaca duas categorias distintas de algoritmos. Na Seção 3.1 um algoritmo de classificação é apresentado, já na Seção 3.2 três métodos de otimização serão abordados.

3.1 Algoritmos de *Wrappers* (Classificadores)

Os métodos de mineração de dados são divididos, tradicionalmente, em aprendizado supervisionado (preditivo) e o não-supervisionado (descritivo) (CIOS et al., 2007). Sendo que a principal diferença entre os métodos de aprendizado supervisionado e não-supervisionados está na característica de que os métodos não-supervisionados não necessitam de uma pré-categorização para os registros, ou seja, não é preciso um atributo alvo. Normalmente, esses métodos utilizam alguma medida de similaridade entre os atributos (MCCUE, 2014). Tarefas como agrupamento e associação, são consideradas como não-supervisionadas. Já no aprendizado supervisionado, os métodos são dotados com um conjunto de dados que possuem uma variável alvo pré-definida e os registros são categorizados em relação a ela. E uma das tarefas mais comuns de aprendizado supervisionado é a classificação.

3.1.1 Naive Bayes

O algoritmo de classificação bayesiana, é uma técnica estatística (probabilidade condicional), fundamentada no teorema de Thomas Bayes (JOYCE, 2003). Segundo o teorema, é possível encontrar a probabilidade de um certo evento ocorrer, dada a probabilidade de um outro evento que já ocorreu (CAMILO; SILVA, 2009). Formalmente, dada uma instância $A = \{a_1, a_2...a_n\}$, em que deseja-se predizer sua classe, o teorema é definido como:

$$P(classe|A) = \frac{P(classe|A) \cdot P(classe)}{P(A)} \tag{1}$$

Como $A = \{a_1, a_2...a_n\}$, tem-se:

$$P(classe|a_1...a_n) = \frac{P(a_1...a_n|classe) \cdot P(classe)}{P(a_1...a_n)} \tag{2}$$

Onde:

- $P(classe|A)$ é a probabilidade futura da classe.
- $P(classe)$ é a probabilidade original da classe.

- $P(A|classe)$ é o valor que representa a probabilidade de preditor, dada a classe.
- $P(A)$ é a probabilidade original do preditor.

Para calcular a classe mais provável da nova instância, calcula-se a probabilidade de todas as possíveis classes e, no fim, escolhe-se a classe com a maior probabilidade como rótulo da nova instância.

3.2 Algoritmos de Otimização

3.2.1 Otimização por Enxame de Partículas (PSO)

O algoritmo de busca de otimização por enxame de partículas ou *Particle Swarm Optimization* (PSO), utiliza uma técnica de computação estocástica baseada em enxames, introduzida a *priori* por Kennedy e Eberhart (1995), que apresenta uma metáfora do comportamento social da interação entre indivíduos (partículas) de um grupo (enxame), onde a movimentação (aprendizagem) de cada indivíduo é definida a partir do conhecimento adquirido por cada partícula, assim como pelo conhecimento compartilhado entre o grupo (RODRIGUES, 2013).

A ideia da metáfora desenvolvida, originou-se a partir da observação de pássaros e cardumes (enxames) de peixes em busca de alimento em uma determinada região. Imaginando-se um cenário em que peixes de um cardume estejam dispostos aleatoriamente em um espaço e estes estão em busca de alimento e, considerando também que eles não sabem onde está a fonte para o alimento e que esta é única. Um problema é concebido, qual o melhor comportamento que os peixes do cardume terão de realizar para conseguir atingir seu objetivo, o mais simples é que eles sigam o peixe que estiver mais próximo do alimento.

O algoritmo PSO, trabalha com o conceito que cada candidato a solução do problema corresponde a uma posição no espaço de busca (MEDEIROS, 2005). Cada candidato a solução, chamado de partícula, possui em sua estrutura um valor, que é avaliado individualmente para cada partícula e que indica a correspondência da partícula como solução do problema, e uma velocidade levando em conta a melhor posição da partícula e a melhor posição do grupo. Assim, ao longo do tempo o grupo acaba alcançando o alimento. Matematicamente, esse processo pode ser definido como:

$$v_i^{t+1} = v_i^t + R_1^t \alpha_1 (\hat{g}^t - x_i^t) + R_2^t \alpha_2 (\hat{x}_i^t - x_i^t) \tag{3}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{4}$$

Onde:

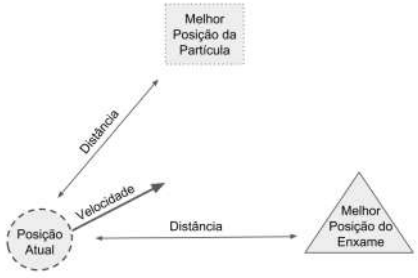
- v_i^t é a velocidade da partícula i na iteração t.

- x_i^t é a posição da partícula i na iteração t.
- $\hat{x}_i^t$ é a melhor posição por onde a partícula já esteve.
- $\hat{g}^t$ é a melhor posição entre todas as posições onde as partículas da população já estiveram.
- R_1^t e R_2^t são coeficientes de aceleração que ajustam como as partículas irão acelerar em direção a nova posição.
- α_1 e α_2 são escalares escolhidos randomicamente a cada iteração, sendo responsáveis pela capacidade de exploração das partículas.

No PSO, as partículas se movimentam baseadas na experiência acumulada individual e compartilhada, ou seja, quando alguma partícula alcança uma boa posição, todo o enxame é beneficiado. A Figura 3.1 ilustra as equação (3) e (4). Na imagem, o círculo mais a esquerda representa a posição atual da partícula (x_i^t), o quadrado na parte superior, representa a melhor posição por onde a partícula já esteve ($\hat{x}_i^t$) e o triângulo mais a direita representa a melhor posição já encontrada por todas as partículas ($\hat{g}^t$). Com isso, a nova posição (x_i^{t+1}) será igual a posição atual mais a velocidade (equação 4). Esta velocidade aplicada, é o resultado da soma dos três argumentos apresentados na equação (3):

1. v_i que representa a velocidade atual da partícula.
2. $R_1^t\alpha_1(\hat{g}^t - x_i^t)$, simboliza a distância entre a melhor posição já encontrada por todas as partículas ($\hat{g}^t$) e a posição atual (x_i^t), multiplicada pela constante de ajuste de aceleração (R_1^t) e por um valor aleatório (α_1), geralmente entre 0 e 1.
3. $R_2^t\alpha_2(\hat{x}_i^t - x_i^t)$ corresponde a distância entre a melhor posição já encontrada pela partícula em questão ($\hat{x}_i^t$) e a posição atual (x_i^t), multiplicada pela constante de ajuste de aceleração (R_2^t) e por um valor aleatório (α_2), geralmente entre 0 e 1.

Figura 3.1 – Modelo de aprendizado das partículas no PSO.



O funcionamento do PSO é resumido no Algoritmo 1.

Algorithm 1 Algoritmo PSO

$t \leftarrow 0$
Para cada Partícula $p_i^t = \langle x_i^t, v_i^t \rangle$ **no** Enxame S **Faça**
    Inicializar Posição x_i^t
    Inicializar Velocidade v_i^t
Fim Para
Enquanto Critério de Parada não é Atingido **Faça**
    Para cada Partícula $p_i^t = \langle x_i^t, v_i^t \rangle$ **no** Enxame S **Faça**
        Calcular *fitness* $f(x_i^t)$
        Se $f(x_i^t)$ é melhor que $\hat{x}_i^t$ **Então**
            $\hat{x}_i^t \leftarrow f(x_i^t)$
        Fim Se
    Fim Para
    Para cada Partícula $p_i^t = \langle x_i^t, v_i^t \rangle$ **no** Enxame S **Faça**
        Se $\hat{x}_i^t$ é melhor que $\hat{g}^t$ **Então**
            $\hat{g}^t \leftarrow \hat{x}_i^t$
        Fim Se
    Fim Para
    Para cada Partícula $p_i^t = \langle x_i^t, v_i^t \rangle$ **no** Enxame S **Faça**
        $v_i^{t+1} = v_i^t + R_1^t \alpha_1 (\hat{g}^t - x_i^t) + R_2^t \alpha_2 (\hat{x}_i^t - x_i^t)$
        $x_i^{t+1} = x_i^t + v_i^{t+1}$
        $p_i^{t+1} = \langle x_i^{t+1}, v_i^{t+1} \rangle$
    Fim Para
    $t \leftarrow t + 1$
Fim Enquanto

Precisamente, o PSO, assume um enxame S que contenha m partículas que são inicializadas randomicamente, onde m é o tamanho do enxame e t é o índice de geração. Cada partícula possui uma posição *n-dimensional*, $x_i^t = \langle x_{i1}^t, x_{i2}^t, ..., x_{in}^t \rangle$, representando uma possível solução para o problema, e um vetor de velocidade *n-dimensional*, $v_i^t = \langle v_{i1}^t, v_{i2}^t, ..., v_{in}^t \rangle$ (CHENG; JIN, 2015). Em cada iteração, uma função de aptidão é aplicada sobre a posição de cada partícula, dependendo de quão boa é esta aptidão, a partícula pode ganhar o *status* de líder do enxame, atraindo as demais partículas para si, ou a partícula pode ter uma aptidão inferior à outras partículas, sendo então liderada e atraída por uma partícula mais apta. Essa dinâmica é caracterizada por sempre permitir o aprendizado das partículas a cada iteração, sem que elas sejam punidas ou removidas do enxame.

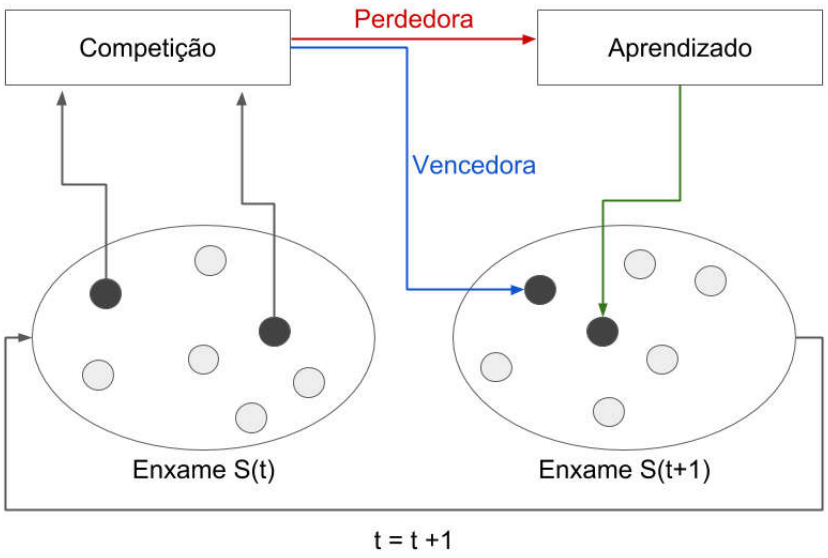
3.2.2 Otimização de Enxame por Competição (CSO)

O algoritmo de otimização de enxame por competição ou *Competitive Swarm Optimizer* (CSO), assim como o PSO abordado anteriormente, utiliza a metáfora de enxames para abstração do seu mecanismo de otimização. No entanto, distingue em alguns aspectos significativos que tornam o CSO um algoritmo que, apesar de compartilhar de um

fundamento semelhante, é um algoritmo distinto e não apenas uma variação do PSO.

O CSO, foi proposto por Cheng e Jin (2015) com o intuito de trabalhar com eficiência para otimização de larga escala (GU; CHENG; JIN, 2018). No CSO, as partículas aprendem de competidores aleatoriamente selecionados. Em cada iteração, o enxame é aleatoriamente dividido em dois grupos e são realizadas competições em pares entre as partículas de cada grupo. Depois de cada competição, a partícula vencedora é passada diretamente para a próxima iteração, enquanto a perdedora irá atualizar sua posição e velocidade aprendendo com a partícula vencedora (Figura 3.2).

Figura 3.2 – Modelo de aprendizagem das partículas no CSO. Figura adaptada (CHENG; JIN, 2015).



Matematicamente, essa proposta foi definida como (GU; CHENG; JIN, 2018):

$$v_l^{t+1} = R_1^t v_l^t + R_2^t(x_w^t - x_l^t) + \Theta R_3^t(\bar{x}^t - x_l^t) \tag{5}$$

$$x_l^{t+1} = x_l^t + v_l^{t+1} \tag{6}$$

Onde, t é o contador de iteração, R_1^t, R_2^t e R_3^t são três valores gerados aleatoriamente no domínio de [0, 1], x_w^t e x_l^t representam as posições da partícula vencedora e da perdedora, respectivamente, $\bar{x}^t$ denota a posição média do enxame atual na iteração t, e Θ controla a influência de $\bar{x}^t$ (GU; CHENG; JIN, 2018).

Formalmente, na resolução do problema de otimização trabalhada pelo CSO, um enxame S^t que contenha m partículas é inicializado aleatoriamente e iterativamente atualizado, onde m é o tamanho do enxame e t é o índice de geração (CHENG; JIN, 2015). Cada partícula possui uma posição *n-dimensional*, $x_i^t = \langle x_{i1}^t, x_{i2}^t, ..., x_{in}^t \rangle$, representando uma possível solução para o problema, e um vetor de velocidade *n-dimensional*, $v_i^t = \langle v_{i1}^t, v_{i2}^t, ..., v_{in}^t \rangle$. Em cada geração, as partículas p_i^t são alocadas aleatoriamente em

pares $m/2$ (considerando um número par de partículas), e depois, uma competição é feita entre as duas partículas de cada par. Como resultado de cada competição, a partícula que possuir uma melhor aptidão, denominada de vencedora, será passada diretamente para a próxima geração do enxame, S^{t+1}, enquanto que a partícula perdedora, atualiza sua posição e velocidade aprendendo com a vencedora. Depois de realizada a aprendizagem, por parte da partícula perdedora, a mesma é passada para a próxima geração do enxame, S^{t+1} (CHENG; JIN, 2015). Em outras palavras, para um tamanho de enxame de m, $m/2$ competições ocorrem para que todas as m partículas participem da competição, para que a posição e a velocidade de $m/2$ partículas sejam atualizadas. O funcionamento do CSO é resumido no Algoritmo 2 (GU; CHENG; JIN, 2018).

Algorithm 2 Algoritmo CSO

$t \leftarrow 0$
Para cada Partícula $p_i^t = \langle x_i^t, v_i^t \rangle$ **no** Enxame S^t **Faça**
    Inicializar Posição x_i^t
    Inicializar Velocidade v_i^t
Fim Para
Enquanto Critério de Parada não é Atingido **Faça**
    Para cada Partícula $p_i^t = \langle x_i^t, v_i^t \rangle$ **no** Enxame S^t **Faça**
        Calcular *fitness* $f(x_i^t)$
    Fim Para
    $S^{t+1} \leftarrow \varnothing$
    Enquanto $S^t \neq \varnothing$ **Faça**
        Selecione Aleatoriamente duas Partículas p_{r1}^t e p_{r2}^t de S^t
        Se $f(x_1^t)$ é melhor que $f(x_2^t)$ **Então**
            $p_w^t \leftarrow p_{l1}^t$
            $p_l^t \leftarrow p_{l2}^t$
        Senão
            $p_w^t \leftarrow p_{l2}^t$
            $p_l^t \leftarrow p_{l1}^t$
        Fim Se
        $v_l^{t+1} = R_1^t v_l^t + R_2^t(x_w^t - x_l^t) + \Theta R_3^t(\bar{x}^t - x_l^t)$
        $x_l^{t+1} = x_l^t + v_l^{t+1}$
        $S^{t+1} \leftarrow S^{t+1} \cup \{p_w^t, p_l^{t+1}\}$
        $S^t \leftarrow S^t \setminus \{p_{r1}^t, p_{r2}^t\}$
    Fim Enquanto
    $t \leftarrow t + 1$
Fim Enquanto

Para reforçar a dinâmica do CSO e distingui-lo do PSO, os autores Cheng e Jin (2015) destacam as seguintes diferenças entre ambos:

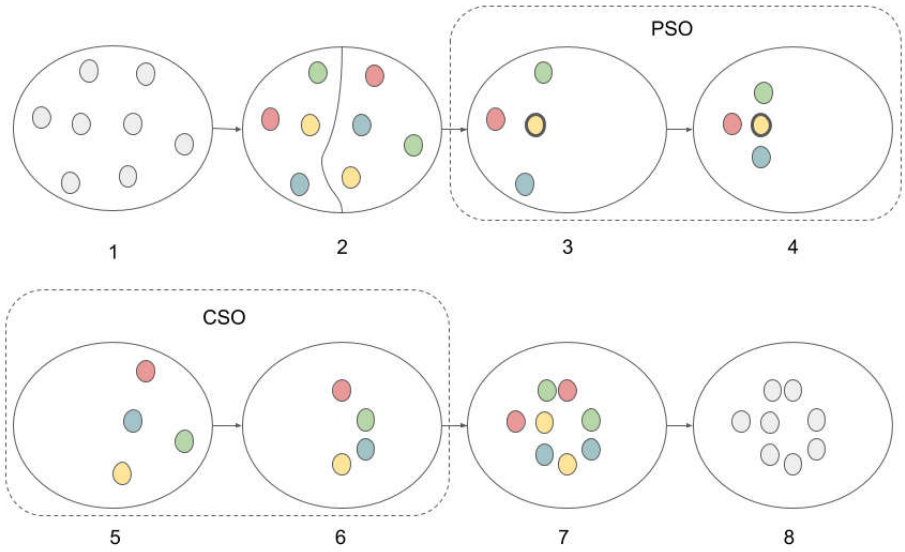
1. No PSO, o sistema dinâmico é liderado principalmente pela melhor posição global ($\hat{g}^t$) e a melhor posição individual ($\hat{x}_i^t$), enquanto que no CSO, não há mais $\hat{g}^t$ ou $\hat{x}_i^t$. Em vez disso, o sistema dinâmico é impulsionado por um mecanismo de competição aleatória, onde qualquer partícula poderia ser um líder em potencial.

2. No PSO, as melhores posições históricas são registradas, enquanto que no CSO, não há memória usada para registrar o histórico de posições. Em contraste, as partículas que perdem uma competição aprendem com os vencedores no enxame atual.

3.2.3 Proposta de Algoritmo Híbrido

Visando explorar ainda mais os métodos de otimização de enxame por partículas, aplicados à seleção de características, um algoritmo híbrido foi modelado e implementado. Objetivando utilizar ambas as estratégias de buscas descritas anteriormente, essa proposta, permite que todas as partículas aprendam a cada nova iteração. A grande diferença do método proposto para os métodos já descritos está na forma como as novas posições serão encontradas. Onde, após uma competição, a metade do enxame com as partículas vencedoras será guiada por uma partícula líder que possui a melhor posição, enquanto que, a outra metade será guiada por vários líderes, compostos pela metade anterior. A Figura 3.3 ilustra este modelo de aprendizagem.

Figura 3.3 – Modelo de aprendizagem do algoritmo híbrido proposto.



A descrição de cada passo apresentado na figura 3.3 é relatado a seguir:

1. Enxame original com M partículas.

2. *M/2* competições são realizadas com as partículas do enxame, separando as partículas vencedoras das partículas perdedoras.

3. Enxame vencedor. Seleciona-se uma partícula líder com a melhor posição do enxame de partículas vencedoras (na Figura 3.3 esta partícula está em destaque).

4. Enxame vencedor com novas posições. Realiza-se a etapa de aprendizagem utilizando a otimização por enxame de partículas.

5. Enxame perdedor. Seleciona-se as partículas líderes, que são as partículas com as novas posições encontradas no passo anterior.

6. Enxame perdedor com as novas posições. As partículas serão movimentas utilizando a otimização de enxame por competição.

7. Novo enxame. Os enxames resultados de 4 e 6 são reagrupados.

8. O novo enxame pode ser usado para realizar uma nova iteração buscando melhores posições.

Categoricamente, na resolução do problema de otimização trabalhada pelo algoritmo proposto, um enxame S^t que contenha m partículas é inicializado aleatoriamente e iterativamente atualizado, onde m é o tamanho do enxame e t é o índice de geração. Cada partícula possui uma posição *n-dimensional*, $x_i^t = \langle x_{i1}^t, x_{i2}^t, ..., x_{in}^t \rangle$, representando uma possível solução para o problema, e um vetor de velocidade *n-dimensional*, $v_i^t = \langle v_{i1}^t, v_{i2}^t, ..., v_{in}^t \rangle$. Em cada geração, as partículas p_i^t são alocadas aleatoriamente em pares $m/2$, e depois, uma competição é feita entre as duas partículas de cada par. Como resultado de cada competição, as partículas que possuírem uma melhor aptidão, denominadas de vencedoras, passaram para a etapa de aprendizagem, utilizando o PSO, enquanto que as partículas perdedoras, atualizarão suas posições e velocidades aprendendo com as vencedoras utilizando o CSO. Depois de realizada a etapa de aprendizagem, por parte da partícula dos subgrupos de partículas, as mesmas são reagrupadas em um mesmo enxame, S^{t+1}.

4 METODOLOGIA

4.1 Considerações Iniciais

4.1.1 Sobre *Python*

A linguagem de programação de alto nível *Python*, foi projetada com a filosofia de priorizar a legibilidade do código sobre a velocidade ou expressividade. É uma linguagem interpretada, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. *Python* atualmente é bastante popular pelo seu modelo de desenvolvimento comunitário, a linguagem é aberta e gerenciada pela organização sem fins lucrativos *Python Software Foundation* [1].

4.1.1.1 *Scikit-learn*

Criada para conduzir os desenvolvedores a construírem aplicações cada vez mais inteligentes e complexas de maneira simplória. A *scikit-learn* é uma biblioteca de código aberto para a linguagem de programação *Python* e trabalha com aprendizagem de máquina. Dentro do seu escopo de implementações, estão algoritmos de classificação, regressão e agrupamento. Além disso, a biblioteca foi projetada para interagir com as bibliotecas *Python* científicas e numéricas *SciPy* e *NumPy* [2].

4.1.2 Acurácia

A acurácia é uma métrica de avaliação de resultados bastante utilizada, em detrimento a sua boa representação da taxa de acerto de um classificador qualquer. Para compreender o cálculo da acurácia, considere que um classificador está sendo avaliado. Para isso, são geradas quatro informações, que juntas constroem a chamada "Matriz de confusão", sobre os resultados desta avaliação inicial. São elas:

- **Verdadeiros Positivos (VP)**: Quando a predição do classificador é uma classe positiva e a resposta esperada também é positiva.
- **Falsos Positivos (FP)**: Quando a predição do classificador é uma classe positiva, mas a resposta esperada era uma classe negativa.
- **Falsos Negativos (FN)**: Quando a predição do classificador é uma classe negativa, mas a resposta esperada era uma classe positiva.

[1] <https://docs.python.org/3/faq/general.html>
[2] <http://scikit-learn.org/stable/faq.html>

- **Verdadeiros Negativos (VN)**: Quando a predição do classificador é uma classe negativa e a resposta esperada também é negativa.

Com todas essas informações contabilizadas, o cálculo da acurácia será:

$$A = \frac{VP + VN}{VP + VN + FP + FN} \tag{7}$$

4.1.3 *F1 Score*

F1 Score ou Pontuação F1 é uma métrica de avaliação que busca informar a média harmônica entre *precision* e *recall*. Ela quantifica a precisão do classificador, através de quantas instâncias o mesmo classificou corretamente. Assim, quanto maior a pontuação F1, melhor é o desempenho do classificador (MISHRA, 2018). Formalmente, pode ser expresso como:

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}} \tag{8}$$

Onde, para as pontuações obtidas através da matriz de confusão descrita na Seção 4.1.2:

- *Precision* é o número de resultados positivos corretos dividido pelo número de resultados positivos previstos pelo classificador.

$$presicion = \frac{VP}{VP + FP} \tag{9}$$

- *Recall* é o número de resultados positivos corretos dividido pela quantidade de amostras relevantes.

$$recall = \frac{VP}{VP + FN} \tag{10}$$

4.1.4 *Holdout*

Para a construção de um classificador eficiente, algumas técnicas de mineração de dados se tornam necessárias. Na técnica *holdout*, os dados fornecidos são divididos aleatoriamente em dois conjuntos independentes, um conjunto de treinamento e um conjunto de teste. Normalmente, dois terços dos dados são alocados para o conjunto de treinamento e o terceiro restante é alocado para o conjunto de teste. O conjunto de treinamento é usado para derivar o classificador, cuja acurácia é estimada com o conjunto de testes (HAN; PEI; KAMBER, 2011).

4.2 Esquematização dos Algoritmos Utilizados

A implementação dos algoritmos utilizados neste trabalho foi desenvolvida totalmente em Python. Estes algoritmos, dividem-se em dois fundamentos discutidos nos Capítulos anteriores. São eles: método de *wrappers* (Seção 2.4) e algoritmos de otimização (Seção 3.2). O grupo de algoritmos codificados trabalha visando selecionar as características mais relevantes de um modelo de dados. A *priore*, serão descritos os traços que são genéricos aos algoritmos implementados e, logo após, os mesmos serão abordados individualmente.

As maiores variações de possibilidades de como selecionar as características mais relevantes devem-se a dois parâmetros. O primeiro, é a estratégia de busca que será aplicada para encontrar subconjuntos. Optou-se aqui, por utilizar a otimização de por enxames de partículas para encontrar bons subconjuntos. O algoritmo de busca, no entanto, deve atuar juntamente com algum *fitness* para guia-lo nessa busca, é nesse momento que aparece o segundo parâmetro, que é a estratégia de avaliação. A estratégia de avaliação, compreende duas grandes categorias, as que consideram cada característica independente (filtro) e tentam avaliar o quão boa é uma característica ou um subconjunto e as que enxergam cada característica de forma dependente (*wrappers*), ou melhor dizendo, aquela que possui uma "dependência" a algum algoritmo de aprendizagem de máquina classificador.

Perceba que isso possibilita uma codificação modularizada. Onde será utilizado um algoritmo de seleção de características no primeiro módulo e um algoritmo de avaliação no segundo módulo.

4.2.1 Implementação do PSO

O cenário de busca de seleção de características é discreto, ou seja, existe um número finito de possibilidades a serem testadas, em quanto que, o PSO foi proposto para otimização contínua. Dado isto, foram necessárias diversas adaptações na implementação para adequar o algoritmo ao papel que desempenha na seleção. O aspecto mais relevante a ser comentado é a necessidade de binarização de características, ação denominada de *Binary Particle Swarm Optimization* (BPSO) pelos autores Nezamabadi-pour, Rostami-Shahrbabaki e Maghfoori-Farsangi (2008). Assim, admitindo um domínio no qual as instâncias do modelo são definidas por N características. Cada partícula do enxame é codificada com dois vetores de N índices com valores binários, referentes a velocidade e a posição da partícula, este último representando um possível subconjunto solução de características. De forma que, se um índice i do vetor de posição, sendo $1 \leq i \leq N$ for 0, a característica correspondente não faz parte do subconjunto, e se for 1, faz parte do subconjunto.

O questionamento que se segue é, o quê fazer para que partículas com apenas valores binários movimentem-se em um espaço de N dimensões. Ainda segundo os autores

Nezamabadi-pour, Rostami-Shahrbabaki e Maghfoori-Farsangi (2008), as partículas estão posicionadas em um espaço de busca no BPSO considerado como um hipercubo, no qual cada partícula pode ser vista movendo-se para os cantos mais próximos e mais distantes do hipercubo, invertendo vários números de bits. A velocidade de movimento é definida em termos de mudanças de probabilidades de que um bit estará em um estado ou outro. Possibilitando dessa forma, que uma partícula se mova em um espaço de estado restrito a 0 e 1 em cada dimensão, onde cada índice i do vetor v_i representa a probabilidade do bit de índice i do vetor x_i ser 1. Com esta definição, $\hat{g}$ e $\hat{x}$ são integrados em {0, 1} e v_i^{t+1}, uma vez que é uma probabilidade, deve ser restrito ao intervalo [0, 1].

$$v_i^{t+1} = v_i^t + R_1^t \alpha_1 (\hat{g}^t - x_i^t) + R_2^t \alpha_2 (\hat{x}_i^t - x_i^t) \tag{11}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{12}$$

Onde a posição é atualizada de acordo com:

$$x_i^{t+1} = \begin{Bmatrix} 0 : if & random() & \geqslant & S(v_i^{t+1}) \\ 1 : if & random() & < & S(v_i^{t+1}) \end{Bmatrix}$$

Sendo $random()$ um número selecionado entre [0.1, 1] e $S(v_i^{t+1})$ uma transformação limitadora sigmoide definida na equação a seguir.

$$S(v_i^{t+1}) = Sigmoide(v_i^{t+1}) = \frac{1}{1 + e^{-v_i^{t+1}}} \tag{13}$$

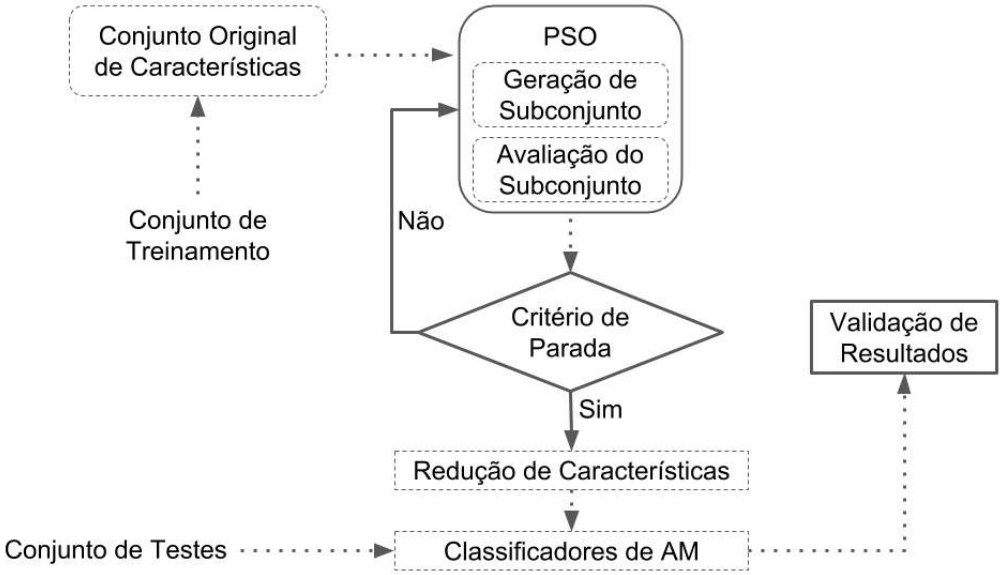
Na implementação do PSO em sua versão discreta, um limite de alcance de $[-v_{max}$, $v_{max}]$ para v_i^{t+1} também é utilizado. Sendo v_{max}, geralmente defindo para ser 6. Embora essa configuração limite a probabilidade de estar entre [0,0025, 0,9975], na prática ela resultará em melhores características de convergência (NEZAMABADI-POUR; ROSTAMI-SHAHRBABAKI; MAGHFOORI-FARSANGI, 2008). A esquematização disto, pode ser observada genericamente na Figura 4.1.

O algoritmos realizarão a busca pelo melhor subconjunto de atributos usando, para direcionar a busca, o mérito de cada partícula feita pelo método avaliador escolhido. Com isso, de forma geral, o algoritmo implementado neste trabalho possui a seguinte estrutura:

1. Inicialmente são geradas M partículas aleatoriamente posicionadas em um enxame, a posição de cada partícula gerada representa um subconjunto $s(s \in S)$, com S sendo o conjunto completo.

2. É calculado um valor de mérito utilizando o avaliador para cada partícula e as melhores posições locais e globais são registradas e atualizadas.

Figura 4.1 – Seleção de características utilizando o PSO para geração de subconjunto.



3. A velocidade é redefinida em detrimento às avaliações das posições geradas pelo avaliador no passo anterior.

4. O PSO movimenta as partículas do enxame, um movimento que é liderado pelas melhores posições já encontradas.

5. Caso o critério de parada seja satisfeito, a melhor posição encontrada é escolhida como o subconjunto ótimo e é então passada para etapa de validação. Caso o critério ainda não seja satisfeito, a execução retorna ao passo 2.

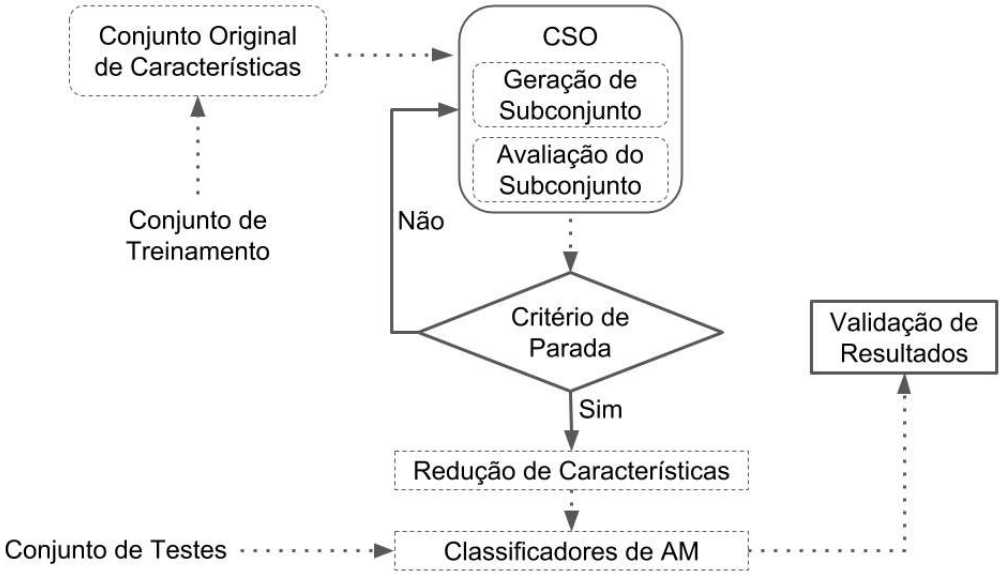
4.2.2 Implementação do CSO

A modelagem do algoritmo de otimização de enxame por competição, não é muito diferente da abordagem apresentada anteriormente no PSO. No entanto, pode-se encontrar uma clara remodelagem na estratégia de busca utilizada por este algoritmo. Agora no CSO, tem-se que devido a competição, apenas metade do enxame é atualizado a cada iteração, o que diminui o número de novas posições encontradas pelo enxame. O modelo esquemático geral pode ser visto na Figura 4.2.

Com base em implementações encontradas na literatura (GU; CHENG; JIN, 2018), foram utilizados seguintes valores para parâmetros: Θ igual a 0,1 e *threshold* igual a 0,5. Com isso, o CSO deverá realizar a busca pelo melhor subconjunto de atributos usando, para direcionar a busca, a competição entre o enxame possuindo como valor determinante o mérito de cada partícula feita pelo método avaliador escolhido.

Um ponto interessante a ser comentado na utilização desta abordagem está na redução do número de novas soluções encontradas a cada iteração, isto em comparação

Figura 4.2 – Seleção de características utilizando o CSO para geração de subconjunto.



com o PSO. Este fenômeno ocorre devido a competição realizada, que só privilegia apenas metade do enxame à realização da aprendizagem. Por fim, de forma resumida, a implementação segue os seguintes passos:

1. Inicialmente são geradas M partículas aleatoriamente posicionadas em um enxame, a posição de cada partícula gerada representa um subconjunto $s(s \in S)$, com S sendo o conjunto completo.

2. É calculado um valor de mérito utilizando o avaliador para cada partícula.

3. O enxame é divido em dois e realizam-se $M/2$ competições entre as partículas.

4. As partículas vencedoras permanecem inalteradas, enquanto que, a velocidade das partículas perdedoras é redefinida em detrimento às informações encontradas no passo anterior.

5. O CSO movimenta as partículas do enxame, um movimento que é liderado pelas partículas vencedoras.

6. Caso o critério de parada seja satisfeito, a melhor posição encontrada é escolhida como o subconjunto ótimo e é então passada para etapa de validação. Caso o critério ainda não seja satisfeito, a execução retorna ao passo 2.

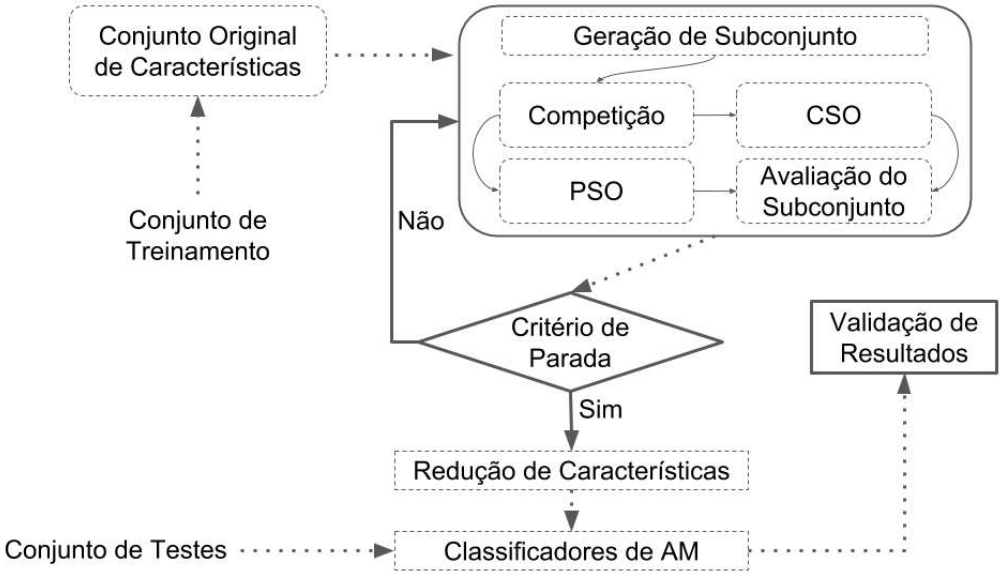
4.2.3 Implementação da Proposta de Algoritmo Híbrido

A principal ideia por traz desta nova proposta é esta: a realização de um novo modelo de aprendizagem do enxame, em que partículas ruins sejam guiadas pelas melhores

partículas e as melhores partículas sejam guiadas pela partícula com a melhor posição, ou seja, a melhor partícula do enxame.

Como pode ser observado na Figura 4.3, os métodos PSO e CSO foram utilizados e cada um recebe parte do enxame original. Suas parametrizações mantiveram-se as mesmas que em sua utilização individual e as métricas de classificação, assim como o classificador, são as mesmas para os dois métodos.

Figura 4.3 – Seleção de características utilizando proposta de algoritmo híbrido.



4.2.4 Avaliação do Subconjunto com o Naive Bayes

O classificador Naive Bayes é um modelo pouco complexo de construir e particularmente útil para grandes volumes de dados. O modelo trabalha com a probabilidade de determinado evento ocorrer, dado a probabilidade de um outro evento já ocorrido.

Após um processo de busca dos algoritmos de otimização, um subconjunto de atributos é selecionado. A classificação do Naive Bayes é usada como a função de adequação para avaliar os subconjuntos de características selecionados. Depois de várias gerações de movimentações no enxame, a partícula com a maior pontuação na classificação é selecionada. Li, Ding e Li (2014), chamaram o trabalho do PSO com o Naive Bayes de PSO-NB, estendendo esta nomenclatura para os outros métodos, tem-se o CSO-NB, para o CSO e o Hibrido-NB para o algoritmo hibrido proposto.

Em relação à codificação do avaliador, foi realizado com o auxílio da biblioteca scikit-learn (PEDREGOSA et al., 2011) para Python. Sobre a avaliação do desempenho do classificador gerado, foi utilizado a técnica de mineração de dados denominada de "Validação Cruzada" com 10 *folds* e a métrica de desempenho *f1-score*.

4.3 Base de Dados

Para analisar os algoritmos abordados, foram utilizadas cinco bases de dados que contêm apenas atributos numéricos, são elas: *Wine*, *Breast-Cancer*, *Ionosphere*, *Sonar* e *Madelon* extraídas de *UCI Repository of Machine Learning Databases* [3]. As peculiaridades das bases de dados serão resumidas a seguir.

Wine

Tabela 4.1 – Detalhes da base de dados *Wine*.

Características do conjunto de dados:	Multivariada
Características do atributo:	Inteiro, real
Tarefas Associadas:	Classificação
Número de instâncias:	178
Número de Atributos:	13
Valores faltantes:	Não
Área:	Física
Data:	01/07/1991

Os dados desta base são os resultados de uma análise química de vinhos cultivados em uma mesma região na Itália, mas derivados de três diferentes cultivares. A base contém uma determinada análise das quantidades de 13 constituintes encontrados em cada um dos três tipos de vinhos.

Breast-Cancer

Tabela 4.2 – Detalhes da base de dados *Breast-cancer*.

Características do conjunto de dados:	Multivariada
Características do atributo:	Real
Tarefas Associadas:	Classificação
Número de instâncias:	569
Número de Atributos:	32
Valores faltantes:	Não
Área:	Vida
Data:	01/11/1995

[3] <http://archive.ics.uci.edu/ml/index.php>

As características desta base são detalhes relativos a 569 diagnósticos referentes a uma massa mamária nos pacientes. Definidas como: benigna ou maligna.

Ionosphere

Tabela 4.3 – Detalhes da base de dados *Ionophere*.

Características do conjunto de dados:	Multivariada
Características do atributo:	Inteiro, real
Tarefas Associadas:	Classificação
Número de instâncias:	351
Número de Atributos:	34
Valores faltantes:	Não
Área:	Física
Data:	01/01/1989

Os dados desta base foram coletados por um sistema de radar. Este sistema consiste em um arranjo faseado de 16 antenas de alta frequência com uma potência total transmitida da ordem de 6,4 quilowatts. Com alvos em elétrons livres na ionosfera. Classificados como "bons" quando alguma evidência de algum tipo de estrutura na iosfera era mostrada ou "ruins" quando não tinham um retorno satisfatório.

Sonar

Tabela 4.4 – Detalhes da base de dados *Sonar*.

Características do conjunto de dados:	Multivariada
Características do atributo:	Real
Tarefas Associadas:	Classificação
Número de instâncias:	208
Número de Atributos:	60
Valores faltantes:	N/A
Área:	Física
Data:	N/A

Esta base de dados contém 111 padrões obtidos por meio de saltos de sinais de sonar de um cilindro de metal em vários ângulos e sob várias condições e 97 padrões obtidos de rochas em condições similares. O rótulo associado a cada registro contém a letra "R" se o objeto for uma rocha e "M" se for uma mina (cilindro de metal).

Madelon

Os dados desta base formam um conjunto de dados artificial que contém pontos de dados agrupados em 32 grupos colocados nos vértices de um hipercubo de cinco dimensões e rotulados aleatoriamente com +1 ou -1. As cinco dimensões constituem 5 características informativas.

Tabela 4.5 – Detalhes da base de dados *Madelon*.

Características do conjunto de dados:	Multivariada
Características do atributo:	Real
Tarefas Associadas:	Classificação
Número de instâncias:	4400
Número de Atributos:	500
Valores faltantes:	N/A
Área:	N/A
Data:	29/02/2008

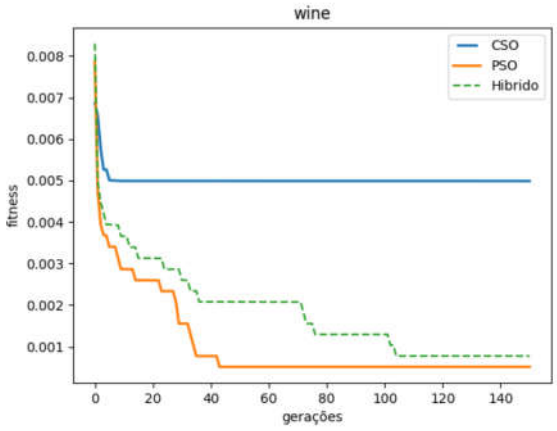
5 RESULTADOS E DISCUSSÕES

Os experimentos foram realizados utilizando um enxame de 100 partículas, com 150 gerações. O tamanho da partícula é igual o total de atributos de cada base de dados e a representação adotada foi a binária. Devido a característica estocástica dos métodos utilizados, os mesmos foram executados 15 vezes e os resultados expostos a seguir apresentam o valor médio de tais execuções. O *fitness* é dado pela taxa de erro encontrada usando a métrica de avaliação *f1-score* abordada na Seção 4.1.3

Wine

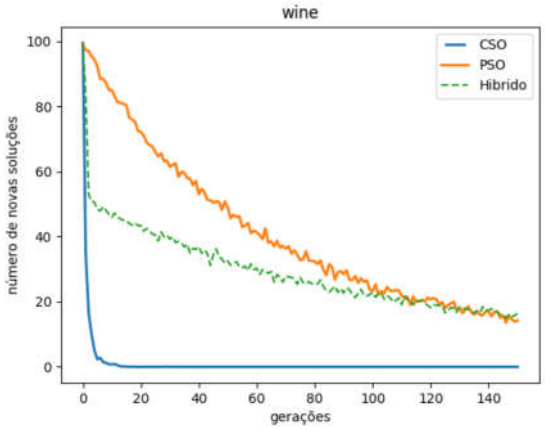
De todas as bases utilizadas, a *wine* é a que possui a menor quantidade de características e instâncias. A Figura 5.1 descreve a minimização na taxa de erro de cada método de otimização. Após as 150 gerações, a taxa de erro que antes era de 0,031, caiu para: 0,0049 com o CSO, 0,00051 com o PSO e 0,00077 com o Híbrido. Como também pode ser observado na Figura, o CSO teve sua convergência limitada após poucas gerações, já o PSO e o Híbrido obtiveram resultados próximos, apesar do PSO ter tido uma convergência mais acelerada.

Figura 5.1 – Comparação dos resultados de *fitness* obtidos com os algoritmos para a base *Wine*.



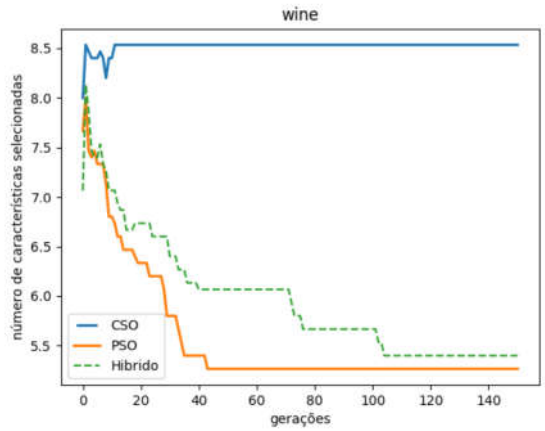
A Figura 5.2 ressalta bem os resultados apresentados na Figura 5.1. Visto que, os algoritmos que criam um maior número de soluções a cada geração, possuem uma maior probabilidade de encontrar as partículas com as melhores posições do enxame, permitindo assim, uma convergência mais acelerada. Vale ressaltar, que na geração 0, todos os algoritmos são inicializados com 100 partículas, ou seja, 100 novas soluções.

Figura 5.2 – Comparação do número de novas soluções obtidas com os algoritmos para a base *Wine*.



O principal propósito deste trabalho é a seleção de característica. Assim, este propósito é bem atendido quando um menor número de características é encontrado, sem que isto prejudique a precisão do classificador. Este objetivo foi exposto na Figura 5.3, onde está apresentado o número de características selecionadas a cada geração. A base de dados abordada aqui, possui por padrão 13 características, após a aplicação dos métodos a quantidade média de características selecionadas foi: 8,53 para o CSO, 5,26 para o PSO e 5,4 para o Híbrido.

Figura 5.3 – Comparação do número de características selecionadas com os algoritmos para a base *Wine*.

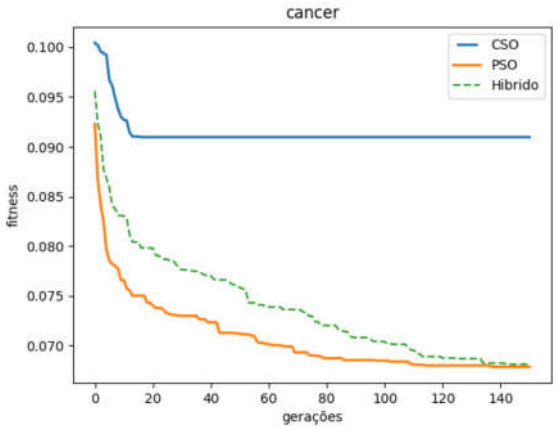


Breast-Cancer

As taxas de erros encontrados sob esta base, podem ser observadas na Figura 5.4. Onde a taxa de erro para a base completa era de 0,606, após aplicar os métodos de otimização a taxa de erro encontrada foi de: 0,09 para o CSO, 0,067 para o PSO e 0,068

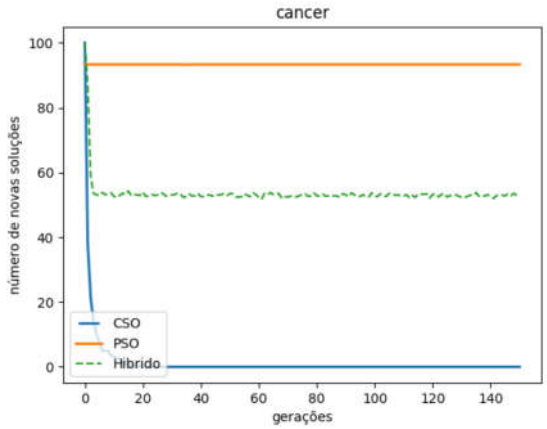
para o Híbrido. Nota-se uma diferença muito pequena entre o resultado encontrado pelo PSO e pelo Híbrido.

Figura 5.4 – Comparação dos resultados de *fitness* obtidos com os algoritmos para a base *Cancer*.



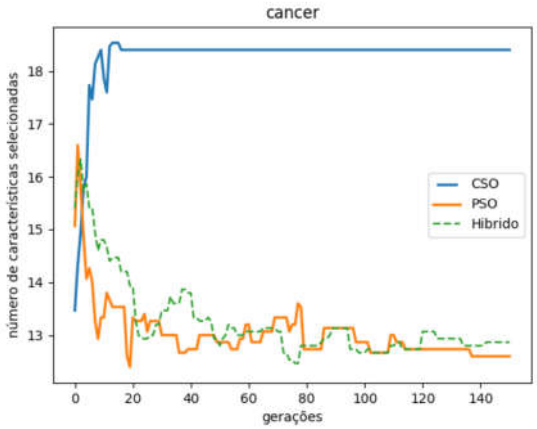
A Figura 5.5 ressalta de forma interessante a estratégia de busca utilizada por cada método de otimização. Enquanto que, as partículas no CSO convergem rapidamente para um ponto, no PSO o oposto acontece, onde o enxame é sempre guiado a descobrir novas posições para as partículas. No centro destas duas abordagens, está o método proposto, o algoritmo híbrido apresenta um equilíbrio satisfatório para a geração de novas soluções.

Figura 5.5 – Comparação do número de novas soluções obtidas com os algoritmos para a base *Cancer*.



A base de dados abordada aqui, possui por padrão 31 características, após a aplicação dos métodos a quantidade média de características selecionadas foi: 18,4 para o CSO, 12,6 para o PSO e 12,8 para o Híbrido. A Figura 5.6 apresenta mais detalhes.

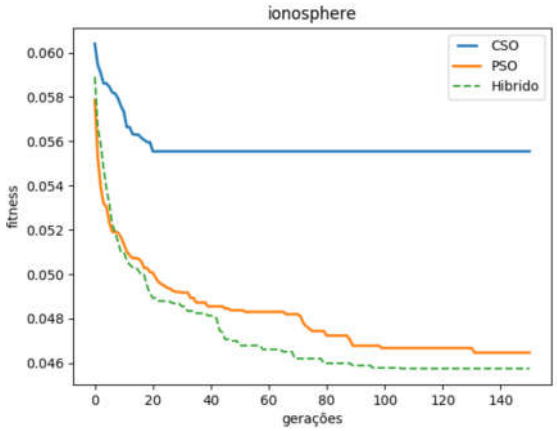
Figura 5.6 – Comparação do número de características selecionadas com os algoritmos para a base *Cancer*.



Ionosphere

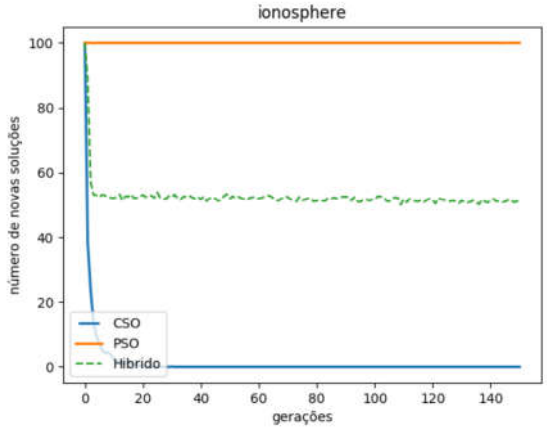
As taxas de erros encontrados sob esta base, podem ser observadas na Figura 5.7. A taxa de erro para a base completa era de 0,078, após aplicar os métodos de otimização a taxa de erro encontrada foi de: 0,055 para o CSO, 0,046 para o PSO e 0,045 para o Híbrido.

Figura 5.7 – Comparação dos resultados de *fitness* obtidos com os algoritmos para a base *Ionosphere*.



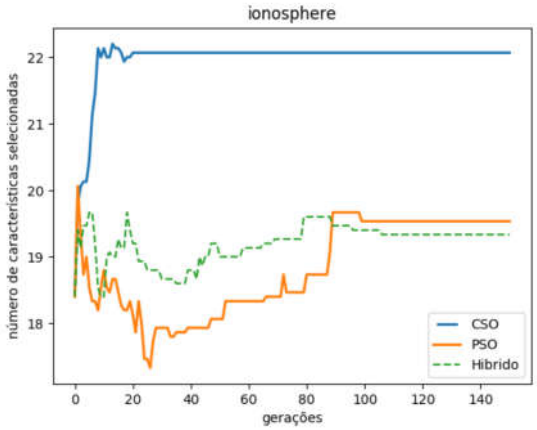
A Figura 5.8 descreve o número de novas soluções encontradas a cada geração utilizando os métodos de otimização. Como já mencionado, a baixa natureza estocástica do CSO o limita a encontrar novas soluções, diferentemente da abordagem do PSO, que possui uma natureza altamente estocástica.

Figura 5.8 – Comparação do número de novas soluções obtidas com os algoritmos para a base *Ionosphere*.



A base de dados abordada aqui, possui por padrão 34 características, após a aplicação dos métodos a quantidade média de características selecionadas foi: 22,06 para o CSO, 19,53 para o PSO e 19,33 para o Hibrido. A Figura 5.9 apresenta mais detalhes.

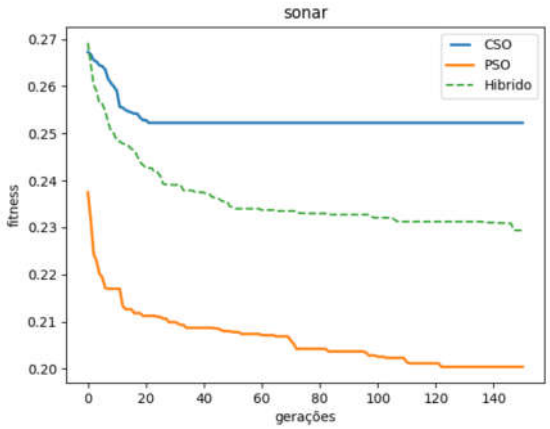
Figura 5.9 – Comparação do número de características selecionadas com os algoritmos para a base *Ionosphere*.



Sonar

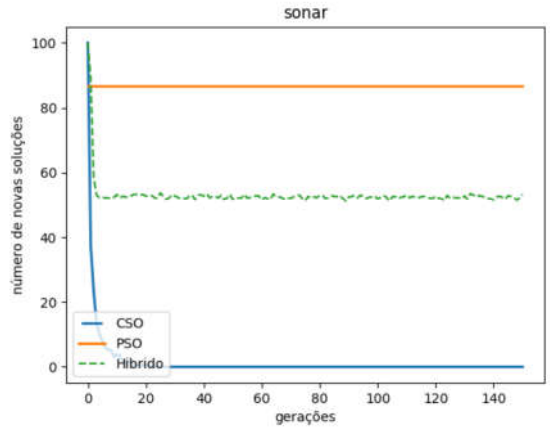
As taxas de erros encontrados sob esta base, podem ser observadas na Figura 5.10. Em que a taxa de erro para a base completa era de 0,338, após aplicar os métodos de otimização a taxa de erro encontrada foi de: 0,25 para o CSO, 0,20 para o PSO e 0,22 para o Híbrido.

Figura 5.10 – Comparação dos resultados de *fitness* obtidos com os algoritmos para a base *Sonar*.



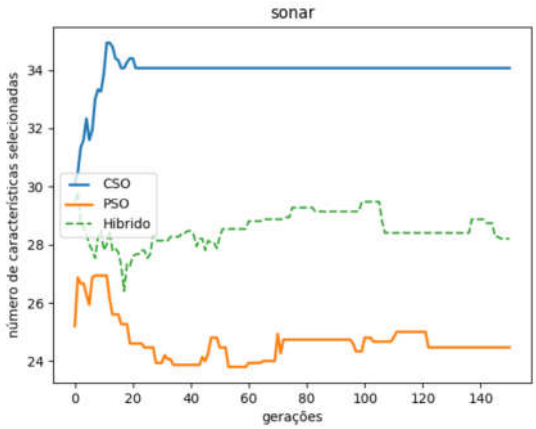
Mais uma vez o algoritmo proposto se encontra no centro das duas abordagens principais. Como pode ser observado na Figura 5.11.

Figura 5.11 – Comparação do número de novas soluções obtidas com os algoritmos para a base *Sonar*.



A base de dados abordada aqui, possui por padrão 60 características, após a aplicação dos métodos a quantidade média de características selecionadas foi: 34,06 para o CSO, 24,46 para o PSO e 28,20 para o Hibrido. A Figura 5.12 apresenta mais detalhes.

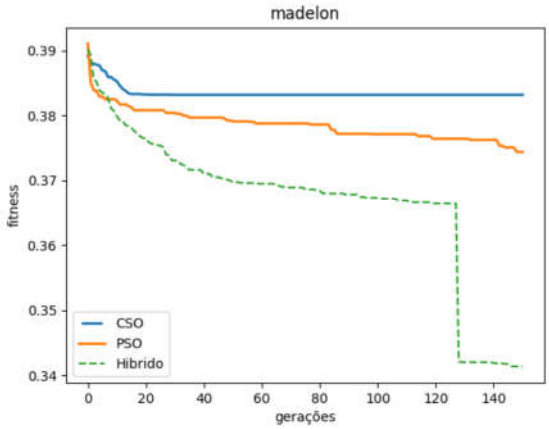
Figura 5.12 – Comparação do número de características selecionadas com os algoritmos para a base *Sonar*.



Madelon

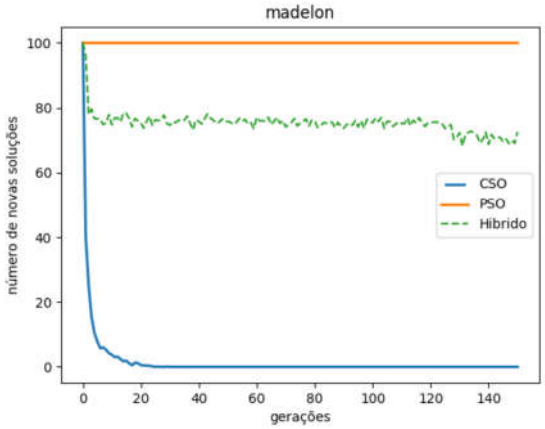
Dentre as bases utilizadas, a *Madelon* é a que possui a maior quantidade de características e instâncias. A Figura 5.13 descreve a minimização na taxa de erro para cada método de otimização. Após as execuções, a taxa de erro que antes era de 0,409, caiu para: 0,38 com o CSO, 0,37 com o PSO e 0,34 com o Híbrido. O algoritmo híbrido teve um resultado significativamente melhor para esta base, chamando a atenção também para o grande declínio da curva entre as gerações 120 e 140 do gráfico.

Figura 5.13 – Comparação dos resultados de *fitness* obtidos com os algoritmos para a base *Madelon*.



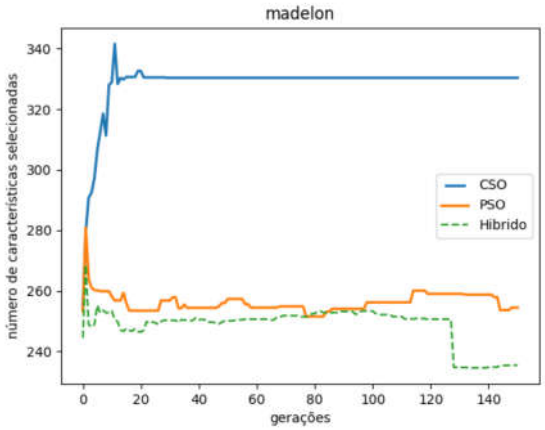
Aqui também o algoritmo proposto se encontra no centro das duas abordagens principais. Como pode ser observado na Figura 5.14.

Figura 5.14 – Comparação do número de novas soluções obtidas com os algoritmos para a base *Madelon*.



A base de dados abordada aqui, possui por padrão 500 características, após a aplicação dos métodos a quantidade média de características selecionadas foi: 330,33 para o CSO, 254,46 para o PSO e 235,46 para o Híbrido. Observa-se que o algoritmo híbrido foi o que encontrou a menor quantidade de características ao final de sua execução. A Figura 5.15 apresenta mais detalhes.

Figura 5.15 – Comparação do número de características selecionadas com os algoritmos para a base *Madelon*.



5.1 Redução da Taxa de Erro

A tabela 5.1 sintetiza um comparativo entre a taxa de erro obtida por cada método de otimização testado sob as bases de dados. Para cada base apresentada na tabela, existe um valor destacado, que refere-se ao método que obteve a menor taxa de erro em comparação às demais.

Tabela 5.1 – Comparativo da Taxa de Erro Encontrada pelos Métodos

	Base Completa	CSO	PSO	Hibrido
Wine	3,1%	0,49%	**0,051%**	0,077%
Cancer	60,6%	9%	**6,7%**	6,8%
Ionosphere	7,8%	5,5%	4,6%	**4,5%**
Sonar	33,8%	25%	**20%**	22%
Madelon	40,9%	38%	37%	**34%**

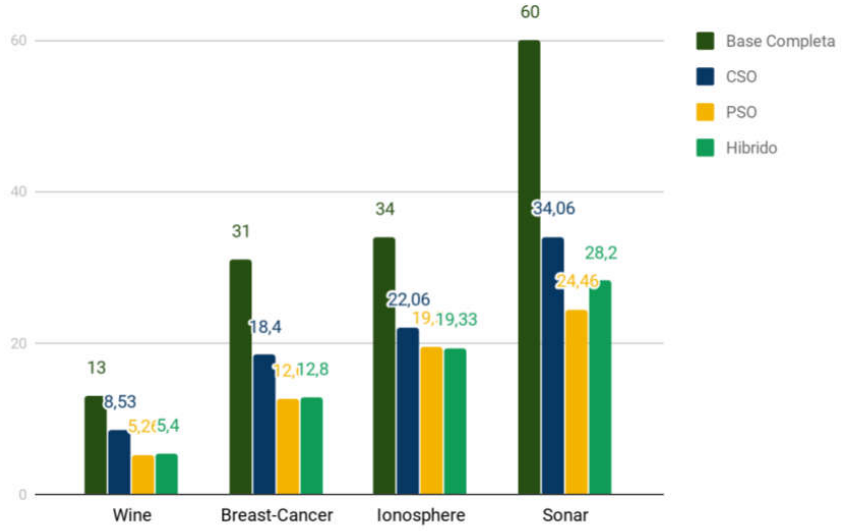
Alguns pontos na análise destes resultados merecem destaque:

- Independente do método utilizado, todos as taxas de erros encontradas são menores que a taxa de erro na base completa.
- Em alguns casos, houve uma redução consideravelmente alta na taxa de erro, como no caso das bases *Brest-Cancer* e *Sonar*.
- Os resultados dos métodos PSO e Híbrido são bem próximos e ambos disputam os melhores resultados.
- No total, para as bases experimentadas, o PSO obteve os melhores resultados em 60% dos casos, enquanto que a proposta híbrida ficou com 40%. O método CSO não obteve um resultado melhor do que os anteriores.

5.2 Seleção de Características

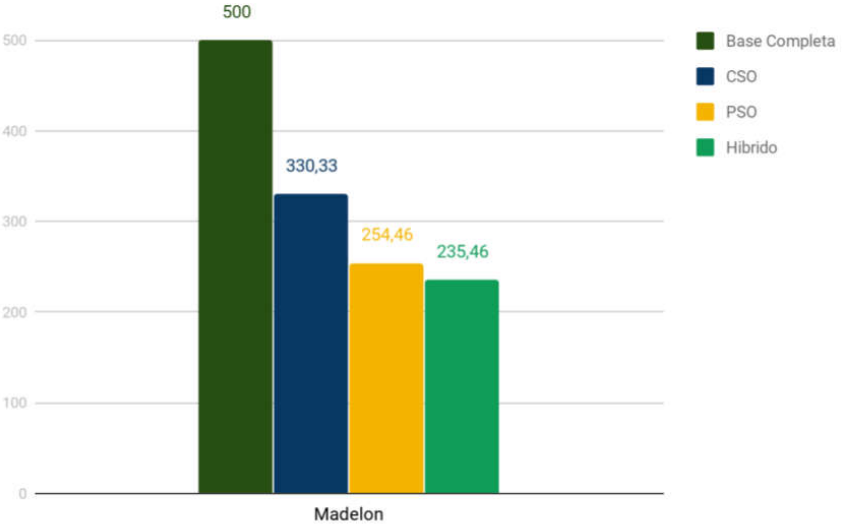
As Figuras 5.16 e 5.17 abordam dados comparativos do total de características selecionadas entre os métodos de otimização. A taxa média total de redução de características selecionadas foi de 48,23%. Um resultado bastante interessante, levando em consideração que, a redução na quantidade de característica proporcionou uma ganho na precisão do classificador.

Figura 5.16 – Quantidade de Características Selecionadas para as bases *Wine*, *Breast-Cacer*, *Ionosphere* e *Sonar*.



Individualmente, a taxa média de redução de características selecionadas para os métodos foi de: 37,46% para o CSO, 54% para o PSO e 53,24% para o Híbrido.

Figura 5.17 – Quantidade de Características Selecionadas para a base *Madelon*.



6 CONSIDERAÇÕES FINAIS

O desenvolvimento do presente trabalho, possibilitou uma investigação de como algoritmos de otimização por enxame atuam na seleção de características sendo guiados por métodos *wrappers*. Com os experimentos realizados e os resultados obtidos, é possível afirmar que, as técnicas de seleção de características proporcionam a construção de classificadores com um desempenho, em grande parte das vezes, superior em relação ao conjunto de características original, reduzindo consideravelmente a taxa de erro.

O algoritmo proposto, denominado até aqui de híbrido, respondeu satisfatoriamente aos experimentos realizados, possuindo em alguns casos, resultados superiores aos dos métodos quando executados individualmente. Ressaltando assim, que a combinação dos dois métodos em um, possibilitou uma movimentação diferenciada no enxame de partículas.

Para continuidade deste estudo, propõe-se uma revisão dos parâmetros utilizados nos métodos PSO e CSO, visando uma convergência mais maleável para o número de novas soluções encontradas a cada geração. Assim como, a realização de testes com outros algoritmos de classificação, buscando obter resultados ainda mais promissores. Por hora, baseado na fundamentação teórica identificada na literatura e nos resultados expostos até aqui, o objeto de estudo deste trabalho foi alcançado, exprimindo a utilização de algoritmos de otimização por enxame aplicados à seleção de características.

REFERÊNCIAS

BLUM, A. L.; LANGLEY, P. Selection of relevant features and examples in machine learning. **Artificial intelligence**, Elsevier, v. 97, n. 1-2, p. 245–271, 1997.

CAMILO, C. O.; SILVA, J. C. d. Mineração de dados: Conceitos, tarefas, métodos e ferramentas. **Universidade Federal de Goiás (UFC)**, p. 1–29, 2009.

CHANDRASHEKAR, G.; SAHIN, F. A survey on feature selection methods. **Computers & Electrical Engineering**, Elsevier, v. 40, n. 1, p. 16–28, 2014.

CHEN, S.; MONTGOMERY, J.; BOLUFÉ-RÖHLER, A. Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. **Applied Intelligence**, Springer, v. 42, n. 3, p. 514–526, 2015.

CHENG, R.; JIN, Y. A competitive swarm optimizer for large scale optimization. **IEEE transactions on cybernetics**, IEEE, v. 45, n. 2, p. 191–204, 2015.

CIOS, K. J. et al. **Data mining: a knowledge discovery approach**. [S.l.]: Springer Science & Business Media, 2007.

GU, S.; CHENG, R.; JIN, Y. Feature selection for high-dimensional classification using a competitive swarm optimizer. **Soft Computing**, Springer, v. 22, n. 3, p. 811–822, 2018.

GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. **Journal of machine learning research**, v. 3, n. Mar, p. 1157–1182, 2003.

HALL, M. A. Correlation-based feature selection for machine learning. University of Waikato Hamilton, 1999.

HAN, J.; PEI, J.; KAMBER, M. **Data mining: concepts and techniques**. [S.l.]: Elsevier, 2011.

INZA, I. et al. Filter versus wrapper gene selection approaches in dna microarray domains. **Artificial intelligence in medicine**, Elsevier, v. 31, n. 2, p. 91–103, 2004.

JOHN, G. H.; KOHAVI, R.; PFLEGER, K. Irrelevant features and the subset selection problem. In: **Machine Learning Proceedings 1994**. [S.l.]: Elsevier, 1994. p. 121–129.

JOYCE, J. Bayes' theorem. 2003.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: **Encyclopedia of machine learning**. [S.l.]: Springer, 1995. p. 760–766.

KOHAVI, R.; JOHN, G. H. Wrappers for feature subset selection. **Artificial intelligence**, Elsevier, v. 97, n. 1-2, p. 273–324, 1997.

LI, J.; DING, L.; LI, B. A novel naive bayes classification algorithm based on particle swarm optimization. **The Open Automation and Control Systems Journal**, v. 6, n. 1, 2014.

MCCUE, C. **Data mining and predictive analysis: Intelligence gathering and crime analysis**. [S.l.]: Butterworth-Heinemann, 2014.

MEDEIROS, J. Enxame de partículas como ferramenta de otimização em problemas complexos de engenharia nuclear. **Universidade Federal do Rio de Janeiro**, 2005.

MISHRA, A. **Metrics to Evaluate your Machine Learning Algorithm**. 2018. Último acesso em 19/11/2018. Disponível em: <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. **Sistemas inteligentes-Fundamentos e aplicações**, v. 1, n. 1, p. 32, 2003.

MOTODA, H.; LIU, H. Feature selection, extraction and construction. **Communication of IICM (Institute of Information and Computing Machinery, Taiwan) Vol**, v. 5, p. 67–72, 2002.

NEZAMABADI-POUR, H.; ROSTAMI-SHAHRBABAKI, M.; MAGHFOORI-FARSANGI, M. Binary particle swarm optimization: challenges and new solutions. **CSI J Comput Sci Eng**, v. 6, n. 1, p. 21–32, 2008.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

RODRIGUES, A. P. d. S. P. Uma metodologia híbrida de otimização aplicada às pás de turbinas hidráulicas axiais. 2013.

SOUZA, B. F. d. **Seleção de características em SVMs aplicadas a dados de expressão gênica**. Tese (Doutorado) — Universidade de São Paulo, 2005.

WALKER, P. R. et al. Data mining of gene expression changes in alzheimer brain. **Artificial intelligence in medicine**, Elsevier, v. 31, n. 2, p. 137–154, 2004.