Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

<TÍTULO DA OBRA>

Guilherme Leite Moreira de Paiva

Trabalho de Graduação

Recife
<DATA DA DEFESA>

Universidade Federal de Pernambuco
Centro de Informática

Guilherme Leite Moreira de Paiva

<TÍTULO DA OBRA>

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: *Prof. Dr. George Darmiton*

Recife
<DATA DA DEFESA>

Dedico este trabalho à ...

Agradecimentos

<DIGITE OS AGRADECIMENTOS AQUI>

<*DIGITE AQUI A CITAÇÂO*>
—<AUTOR> (<NOTA>)

Resumo

<DIGITE O RESUMO AQUI>

Palavras-chave: <DIGITE AS PALAVRAS-CHAVE AQUI>

Abstract

Keywords: <DIGITE AS PALAVRAS-CHAVE AQUI>

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

Neste capítulo será dada uma introdução à classificação em si, à combinação de classificadores bem como uma breve análise dos algoritmos Bagging, Boosting e ICS-Bagging. Para que este assunto seja melhor compreendido, será apresentado o contexto histórico, a motivação do uso de combinação de classificadores e os desafios atuais. Após a seção de motivação e contextualização seguirá um detalhamento deste trabalho, abordando objetivo e estrutura do mesmo.

1.1 Motivação e Contextualização

1.1.1 Histórico

Os primeiros trabalhos de aprendizagem de máquina surgiram há mais de seis décadas. E de uma maneira mais ampla, esses primeiros trabalhos consistiam em dar ao computador problemas que ele pudesse resolver de forma mais genérica. Junto com esses primeiros estudos, diversos outros surgiram na qual a aprendizagem de máquina atua.

Ao dividir os problemas de aprendizagem de máquina, de uma forma até mesmo didática, pode-se segregar em: agrupamento, discriminação e generalização. O primeiro consiste em agrupar dados de acordo com suas características, de forma que seja possível extrair informação útil desses agrupamentos. No problema da discriminação, que basicamente é achar uma forma de reconhecer um conceito, dado um conjunto de conceitos exemplos. No último, o da generalização, reduz-se uma regra, tornando-a mais abrangente e menos custosa.

Reconhecimento de padrões foca principalmente no problema anteriormente mencionado da discriminação. E seu objetivo é classificar padrões, discriminando-os dentre duas ou mais classes. A base de um sistema discriminante é um classificador, que como o próprio nome sugere, classifica uma nova instância como pertencente a uma determinada classe de interesse. E essa eficácia pode ser medida pela taxa de acerto, pela variância, e pela eficiência em termos de custo computacional.

Um exemplo bastante conhecido e simples é o *K-Nearest Neighbor*, KNN [**?**]. Este algoritmo consiste basicamente em ado um padrão x a ser classificado e um conjunto de padrões conhecidos T, obter as classes dos K elementos de T mais próximos de x. A classe que obtiver maior ocorrência, ou peso, será a classe de x. A descrição do algoritmo pode ser vista em Algorithm 1.

Algorithm 1 KNN

Require: K: um número
Require: T: conjunto de treinamento
Require: x: elemento para ser classificado
Require: L: uma lista

1. **for all** $t_i \in T$ **do**
2. $d_i = distance(t_i, x)$
3. adicione $(d_i, Classe(t_i))$ em L
4. **end for**
5. $Ordene(L)$ de acordo com as distâncias
6. obtenha os K primeiros elementos de L
7. **return** a classe de maior ocorrência, ou peso, entre os K

No entanto, como podemos ver com o exemplo acima, a classificação é dada por um único classificador. Como consequência natural, a evolução nesta área seu pautou em combinar vários classificadores para melhorar os resultados. Surgiram então diversos algoritmos que combinam classificadores, dentre eles o AdaBoost e Bagging, que são muito difundidos e servem de base para uma série de algoritmos de combinação de classificadores. Vale ressaltar que existem diversas outras formas de se combinar classificadores na literatura [**?**] que não se limita apenas a evolução dos algoritmos mencionados.

1.1.2 Combinação de Classificadores

Classificar algo sempre fez parte e é algo que ocorre várias vezes com os seres humanos. Um médico ao diagnosticar um paciente está classificando o mesmo como portador de uma determinada enfermidade ou não, um sommelier ao dar sua opinião sobre um vinho pratica da mesma tarefa. Como podemos ver, a tarefa de classificar algo é de suma importância e nada mais natural que essa tarefa fosse feita por uma máquina. Inicialmente, a abordagem era criar um classificador que pudesse, dado uma instância desconhecida, dizer a qual classe esta instância pertence. No entanto, novamente fazendo analogia ao comportamento humano, ao tomar uma decisão nada mais natural que uma pessoa consulte a opinião de outras pessoas. E isto é a base para se combinar classificadores, a decisão final não é baseada na resposta de um único classificador e sim nas respostas de vários.

Na literatura temos vários algoritmos de combinação de classificadores, os primordiais foram: Bagging e Boosting. O algoritmo Bagging introduziu o conceito de bootstrap agreggating, na qual cada classificador é treinado com uma amostra da base de dados, cada classificador é colocado em um conjunto de classificadores e quando uma instância desconhecida é colocada, pelo voto da maioria dos classificadores ou por outro tipo de votação, a instância é classificada como pertencente a uma determinada classe. Já os algoritmos da família Boosting, que tem como algoritmo mais conhecido o AdaBoost, partem de ideia de que se uma instância é classificada de forma errada, esta instância deve ter mais importância e como consequência se uma instância é classificada de forma correta, sua importância é diminuída. A motivação deste algoritmo é termos vários classificadores ditos fracos, mas que são especialistas em um subdomínio do problema.

A partir desses algoritmos, que poderíamos chamar de canônicos, vários outros surgiram. Um em especial é o algoritmo ICS-Bagging. Que de forma mais ampla, combina os classificadores amparado por uma função de *fitness*, de forma que um classificador só é adicionado na *pool* de classificadores se o fato deste classificador ser adicionado na *pool* de classificadores aumenta o *fitness* da *pool*.

1.1.3 Métricas de Diversidade

Combinação de classificadores é uma área de pesquisa estabelecida e os algoritmos para gerar um conjunto de classificadores baseados em Bagging e Boosting têm mostrado resultados de sucesso. A chave para o sucesso destes algoritmos é que, intuitivamente, estes algoritmos constroem um conjunto de classificadores *diversos*. Por esse motivo diversidade é uma característica importante quando se combina classificadores. No entanto, não existe uma definição estrita sobre o que é percebido intuitivamente como diversidade, dependência, ortogonalidade ou complementariadade dos classificadores.

Várias métricas que correlacionam as saídas dos classificadores são derivadas de estudos vindos da estatística. Existem métodos, fórmulas e ideias que tem como objetivo quantificar a diversidade de um conjunto de classificadores. Para este trabalho, foi utilizada na análise dos experimentos as métricas: Entropia, Kohavi Wolpert Variance e Q Statistics.

1.1.4 Bases Balanceadas e Bases Desbalanceadas

A distribuição de uma classe, ou seja, a proporção de instâncias que pertence a cada classe do conjunto de dados, desempenha um importante papel na combinação de classificadores. Uma base de dados é dita desbalanceada quando o número de instâncias de uma classe é muito maior que o da outra classe. De forma intuitiva, uma base balanceada mantém um proporção mais ou menos igual entre as classes. E este ponto é de suma importância quando se trata de combinação de classificadores pois a abordagem dos algoritmos e as métricas utilizadas diferem quando uma base é desbalanceado ou balanceada.

1.2 Objetivo

O objetivo deste trabalho é analisar os parâmetros da função de *fitness* bem como diversas métricas de diversidade para o algoritmo ICS-Bagging e avaliar seu desempenho. A medida de desempenho será a área sobre a curva ROC de 5-cross-fold validation de cada base de dados. As bases de dados a serem analisadas são: Glass1, Pima, Iris0, Yeast1, Vehicle2, Vehicle3, Ecoli1, Ecoli2, Glass6, Yeast3, Ecoli3, Vowel0, Glass4, Ecoli4 e Page-blocks-1-3 . Todas essas bases são balanceadas e foram obtidas no repositório Keel. No final do trabalho será possível identificar qual métrica obteve o melhor desempenho e qual a melhor proporção da medida de diversidade na função de *fitness* do algoritmo ICS-Bagging. Depois de obtida a melhor proporção da medida de diversidade, o algoritmo será analisado utilizando-se outras métricas de diversidade.

1.3 Estrutura do Trabalho

O restante deste trabalho possui um capítulo com detalhes sobre diferentes algoritmos de combinação de classificadores bem como uma abordagem mais detalhada do algoritmo ICS-Bagging. Durante o capítulo, será abordado o conceito de métrica de diversidade, bem como várias métricas de diversidade que serão analisadas neste trabalho. Logo após, segue um capítulo mostrando os resultados das analises das diversas métricas de diversidade bem como sua proporção na função de *fitness* bem como uma análise comparativa com algoritmos clássicos. Por fim, será concluído como se dá o comportamento dessas métricas de diversidade e a importância das mesmas na função de fitness.

CAPÍTULO 2

Combinação de Classificadores

Neste capítulo, será mostrado um conceito e análise de combinação de classificadores e os principais algoritmos desta área, Bagging e Boosting. Cada seção aborda os conceitos básicos do tema e de cada algoritmo, o seu pseudo-código e suas características principais, bem como as variações mais conhecidas. Ao final, de maneira mais detalhada, será analisado o algoritmo ICS-Bagging com foco nas possíveis métricas de diversidade que este algoritmo pode usar bem como uma análise a respeito de métricas de diversidade.

2.1 Selecionar Classificadores

Antes de se combinar classificadores de fato é preciso selecioná-los. Em uma análise menos acurada, parece ser algo intuitivo. No entanto, neste assunto reside alguns aspectos importantes que servem de base para o entendimento dos algoritmos que serão tratados nas próximas seções. A ideia base de se selecionar classificadores é de que existe uma espécie de "oráculo"que pode selecionar o melhor classificador para uma dada entrada **x**. A decisão deste melhor classificador é tida como sendo a decisão do conjunto de classificadores. A ideia de usar diferentes classificadores para diferentes entradas foi primeiramente sugerida por Dasarathy e Sheela em 1978 [ref 118]. Os autores combinaram um classificador linear e o *K-Nearest Neighbor*. A combinação dos classificadores identifica o domínio de conflito dentro do espaço de características e utiliza o *K-Nearest Neighbor* neste domínio, enquanto que o classificador linear é utilizado nos demais domínios. E no ano de 1981 Rastrigin e Erenstein [ref 119] propuseram uma metodologia de seleção de classificadores como é conhecida atualmente. Posteriormente, o interesse em seleção de classificadores foi impulsionado pela publicação de Woods et al. [ref 120]. Os autores introduziram o termo *seleção dinâmica de classificadores* para denotar o processo de escolher um membro do conjunto de classificadores para tomar uma decisão baseada na entrada **x**. Partindo desta proposta, para se construir a seleção do conjunto de classificadores, algumas perguntas precisam ser feitas:

- Como construir os classificadores individuais?
- Como avaliar a competência de cada classificador dado uma entrada **x**?
- Uma vez que a competência de um classificador foi avaliada, qual estratégia deverá ser utilizada?

2.1.1 Regras de Combinação

Como será visto adiante, alguns algoritmos de combinação de classificadores possuem regras de combinação embutidas no seu próprio algoritmo. Por exemplo, o algoritmo Bagging utiliza uma maioria dos votos simples, já no AdaBoost é utilizado uma maioria dos votos com pesos. No entanto, um conjunto de classificadores pode ser treinado simplesmente usando um subconjunto do conjunto de treinamento, diferentes parâmetros dos classificadores ou até mesmo diferentes subconjuntos de características.

Os classificadores podem ser combinados usando uma ou mais regras de combinação. Algumas dessas regras de combinação operam apenas nos *labels* das classes, outras possuem mecanismos mais sofisticados. Mas, de forma prática, pode-se dividir as regras de combinação entre métodos baseados em votos, métodos algébricos e outras formas de combinação.

A regra baseada em votos, é bastante simples, e como o próprio nome já diz, cada classificador dá um voto ao classificar uma nova instância. Duas formas de votos são muito conhecidas, o voto da maioria e o voto com pesos. No primeiro caso, cada classificador, ao receber uma nova instância a ser classificada, provê seu voto como sendo o resultado da sua classificação. A classe que foi apontada como a maioria dos votos é a classe na qual a instância será classificada. Analogamente, nos votos com pesos, alguns classificadores ou até mesmo instância de treinamento, possuem um peso maior. De forma que a votação se dá da mesma forma, apenas algumas instâncias ou classificadores detêm um peso maior no seu voto.

Outras regras são conhecidas na literatura, porém não é o objetivo de estudo deste trabalho. Estes incluem *Borda count,behavior knowledge space* e *decision templates* como rol apenas exemplificativo, já que a lista não se exaure dentro destes.

2.1.2 Diversidade

O sucesso de um sistema que combina classificadores reside na diversidade dos seus membros. De forma que se todos os classificadores fornecessem a mesma saída, nada estaria sendo feito, e nenhum erro poderia ser corrigido. Por isso cada classificador individual precisa de diferente, de certa forma, dos demais classificadores. Isso leva ao raciocínio de que erros cometidos por alguns classificadores são compensados por acertos dos outros classificadores, assim obtêm-se uma diminuição do erro global. Outra forma de se obter ganho nessa abordagem, é que alguns classificadores são bons em determinados domínios do problemas, e a diversidade destes classificadores faz com que uma maior região do problema possa ser resolvida de uma forma maior acurada.

2.1.3 Pontos Fracos

A contrapartida do aumento da taxa de acerto ao se combinar classificadores ocorre um principalmente três pontos: é necessário mais armazenamento, mais capacidade computacional e a complexidade de entendimento aumenta. O primeiro ponto fraco, necessidade de mais armazenamento, é consequência direta de que vários classificadores e não apenas um precisa ser armazenado após a fase de treinamento. O segundo problema decorre naturalmente do aumento da capacidade computacional requerida dado que vários classificadores precisam ser processa-

dos e não apenas um. O último, complexidade no entendimento, envolve uma maior dificuldade para se compreender de forma intuitiva ou até mesmo trivial o que realmente incorpora melhores resultados nos algoritmos.

2.2 Bagging

Bagging é um algoritmo [**?**] de combinação de classificadores que foi desenvolvido com o intuito de melhorar a acurácia na tarefa de classificação. O principal conceito deste algoritmo é o *bootstrap aggregation*, na qual o conjunto de treinamento para cada classificador é construído a partir de uma amostra escolhida aleatoriamente do conjunto de treinamento. Os classificadores são então combinados e ao se apresentar uma nova instância, cada classificador fornece como resultado a classe na qual essa nova instância pertence. Normalmente, a classe que obteve mais votos, é dita como sendo a classe desta instância.

Algorithm 2 Bagging

Dado o conjunto de treinamento T
for all t = 1, ..., n **do**
  Para cada amostra S do conjunto de treinamento T selecione m exemplos aleatórios com reposição
  Considere ht o resultado do classificador t para o conjunto de treinamento S
end for
$Resultado \leftarrow$ a maioria dos votos de $(h1(x), ..., hT(x))$
return $Resultado$

Assim, como pode ser observado melhor acima, para cada n interações uma réplica do conjunto de treinamento é criada. E um classificador é treinado com essa réplica, este processo continua até uma quantidade de interações desejada. Após uma quantidade de interações previamente escolhida, uma combinação de classificadores é gerada e a maioria dos votos desses classificadores determina a classe de uma nova instância.

2.3 Boosting

Boosting é um método geral para melhorar a precisão de uma classificação. O objetivo deste método é produzir um classificador forte a partir de um conjunto de vários classificadores fracos. Um classificador é dito fraco quando este tem uma taxa de acerto boa, mas somente para uma porção da base de dados. Por isso a ideia de combinar vários classificadores ditos fracos para formar uma combinação de classificadores que tem uma precisão melhor. Um analogia favorável ao entendimento é que um classificador fraco é um bom classificador para um determinado domínio, e a junção de vários classificadores fracos forma uma combinação de classificadores forte.

O método *boosting* treina iterativamente cada classificador atribuindo a cada instância um peso após este treinamento. Os pesos de cada instância são acrescidos quando esta instância

Algorithm 3 Boosting

Dado o conjunto de treinamento T
for all t = 1, ..., n **do**
Para cada amostra S do conjunto de treinamento T selecione m exemplos aleatórios com reposição
Considere ht o resultado do classificador t para o conjunto de treinamento S
end for
$Resultado \leftarrow$ a maioria dos votos de $(h1(x), ..., hT(x))$
return $Resultado$

é classificada incorretamente e analogamente este peso é decrementado quando classificado corretamente. Isto faz com que cada classificador foque nos exemplos mais difíceis. Depois que a cominação de classificadores for gerada, uma regra de escolha é usada, maioria dos votos por exemplo.

2.3.1 AdaBoost

Um dos mais famosos algoritmos da família *boosting* é o *AdaBoost*, este algoritmo foi a primeira abordagem prática do *Boosting* e hoje é apontando como um dos top dez dos algoritmos de aprendizagem de máquina[94 review] . Este algoritmo utiliza todo o conjunto de dados para treinar cada classificador, mas a cada iteração é dado um foco maior a instâncias difíceis de classificar. O objetivo é classificar corretamente na próxima iteração uma instância que foi classificada de forma errada na iteração atual. Com isso, é dado um foco maior em instâncias que são difíceis de se classificar. Depois de cada iteração, os pesos das instâncias que foram classificadas de forma errada são aumentando; e em contraste, os pesos das instâncias que foram classificadas corretamente são decrementados. Além disso outro peso é atribuído a cada classificador individual, dependendo da sua taxa de acerto na fase de treinamento. Finalmente, quando uma nova instância é apresentada, cada classificador dá um voto, e a classe selecionada foi a que obteve a maioria dos votos. Lembrando que os classificadores possuem pesos diferentes nas votações. O algoritmo pode ser conferido abaixo.

2.4 ICS-Bagging

O algoritmo ICS-Bagging, acrônimo de *Iteractive Classifier Selection Baggin*, é o ponto central e objetivo de estudo deste trabalho. Por isso um maior grau de detalhamento se faz necessário. O ICS-Bagging é baseado em uma inicialização iterativa para formar o conjunto de classificadores. Cada iteração gera um conjunto de classificadores e seleciona o melhor classificador deste conjunto. A amostragem de inicialização usa uma probabilidade de amostragem a partir de cada classe, sendo esta probabilidade derivada a partir da taxa de erro da classe. O algoritmo abaixo descreve o funcionamento do ICS-Bagging.

Algorithm 4 AdaBoost

1. **ENTRADA:**
1. Conjunto de Treinamento: $\mathscr{D} = \{(x_i, y_i), i = 1, \ldots, N\}$, onde $x_i \in \mathbf{R}^d$ e $y_i \in \{-1, +1\}$.
1. O número de amostras em cada iteração: m
1. Classificador Fraco: $\mathscr{L}$ that automatically learns a binary classifier $h(x) : \mathbf{R}^d \mapsto \{-1, +1\}$ de um conjunto de treinamento.
1. O número de iterações: T
2. **SAÍDA:**
2. O Classificador Final: $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$
3. **Algorithm**
4. Inicialize a distribuição $D_0(i) = 1/N, i = 1, \ldots, N$
5. **for** $t = 1$ até T **do**
6. Sample m examples with replacement from $\mathscr{D}$ de acordo com a distribuição $D_{t-1}(i)$.
7. Treine o classificador $h_t(x)$ usando os exemplos da amostra
8. Compute a taxa de erro $\varepsilon_t = \sum_{i=1}^{N} D_{t-1}(i) I(h_t(x_i) \neq y_i)$ onde $I(z)$ outputs 1 quando z é verdadeiro e zero caso contrário.
9. Saia do loop se $\varepsilon > 0.5$.
10. Compute o peso como α_t em

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$$

11. Atualize a distribuição em as

$$D_t(i) = \frac{1}{Z_t} D_{t-1}(i) \exp(\alpha_t I(y_i \neq h_t(x_i))$$

where $Z_t = \sum_{i=1}^{N} D_{t-1}(i) \exp(\alpha_t I(y_i \neq h_t(x_i))$.

12. **end for**
13. Construct the final classifier $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$.

Algorithm 5 ICS-Bagging

Require: $\mathscr{T}$: conjunto de treinamento.
Require: $\mathscr{V}$: conjunto de validação.
Require: N: tamanho da pool.
Require: K: número de classificadores a serem adicionados a cada iteração.
Require: α: parâmetro da função de fitness.
Require: *diversity*: métrica de diversidade a ser usada.

1. $\mathscr{P} \leftarrow$ lista vazia de classificadores.
2. Gera o classificador usando uma inicialização aleatória das amostras adicionando-as em $\mathscr{P}$.
3. **while** $|\mathscr{P}| < N$ **do**
4. $pesos \leftarrow Probabilidade_{classe_i} = 1 - \frac{FN_{classe_i}}{\sum_{classe_j \in classes} FN_{classe_j}}$.
5. $C = K$ classificadores usando *pesos* para executar a inicialização das amostras.
6. **for all** classificador $c_i \in C$ **do**
7. Adicione c_i em $\mathscr{P}$.
8. $acc \leftarrow AUC(\mathscr{P})$
9. $div \leftarrow diversity(\mathscr{P})$
10. $fitness(c_i) \leftarrow \alpha \times acc + (1 - \alpha) \times div$
11. Remova c_i de $\mathscr{P}$.
12. **end for**
13. $melhor \leftarrow argmax(fitness, C)$

Abaixo, uma explicação formulada de maneira textualmente mais livre:

Pré-processamento: Antes de gerar os classificadores, alguma técnica de pré-processamento pode ser aplicada ao conjunto de treinamento. Este pré-processamento consiste em remover ruídos, dados redundantes ou gerar novos dados. No entanto, o algoritmo ICS-Bagging não tem essa etapa.

Gerar K classificadores: Este passo gera K classificadores usando uma amostra de inicialização (com reposição). No primeiro passo os pesos são os mesmos para todas as classes; depois do primeiro passo os peses são atualizados para priorizar a classe que possui maior taxa de erro. A motivação de se usar pesos para guiar o processo de inicialização é focar nos exemplos que são mais difíceis de serem classificados. E a motivação de se utilizar K >1 classificadores para que se aumente a região de busca, aumentando a probabilidade de se achar um classificador que consideravelmente aumente a diversidade e a acurácia na classificação.

Adicionar o melhor classificador na *pool*: Para cada um dos K classificadores gerados os seguintes passos são executados para achar o melhor classificador. C é a lista dos K classificadores gerados, $\mathscr{V}$ é o conjunto de validação (neste trabalho foi utilizado o conjunto de treinamento como sendo o conjunto de validação) e $\mathscr{P}$ é a *pool* atual de classificadores.

Para todos os classificadores em C, o classificador é adicionado à *pool* (Linha 4), e então o *fitness* da *pool* (Linha 5) é calculado pelo fórmula abaixo:

$$fitness = \alpha \times ACC + (1-\alpha) \times DIV \tag{2.1}$$

onde ACC é a acurácia de classificação da *pool*, DIV é a métrica de diversidade, e α é o parâmetro que regula a proporção que a diversidade ou a acurácia da classificação possui na função de *fitness*, este valor varia entre 0.01 e 0.99.

Se a *pool* alcança o maior *fitness* com esse classificador, então o índice desse classificador é salvo em $melhor_{index}$ (Linha 6 - 9). O classificador é removido da *pool* (Linha 10) e o processo começa novamente utilizando outro classificador até que o melhor classificador seja retornado (Linha 12).

Para a classificação de uma amostra de teste, qualquer regra de combinação pode ser utilizada. Para este trabalho foi utilizado a regra da maioria dos votos [**?**] para combinar as saídas dos classificadores na *pool*.

Quaisquer métricas de classificação ou de diversidade podem ser usadas na Equação 2.1, mas ambas precisam ser normalizadas (entre 0 1). Neste trabalho foi utilizado a área sobre a curva ROC - AUC, e como medida de diversidade foi utilizada a Entropia E [**?**], a métrica tal e a métrica tal...

$$E = \frac{1}{N}\sum_{j=1}^{N} \frac{1}{(L - \lceil \frac{L}{2} \rceil)} min\{l(z_j), L - l(z_j)\} \tag{2.2}$$

|pool| = N: Se a *pool* já contém o número desejado de classificadores, então a *pool* é retornada.I

Atualiza o peso de cada classe: Como a precisão de cada classe pode mudar depois que um novo classificador for inserido na pool, os pesos precisam ser atualizados utilizando a Equação 2.3,

$$peso_{classe} = 1.0 - \frac{acuracia_{classe}}{\sum_{c \in classes} acuracia_c} \tag{2.3}$$

onde $peso_{classe}$ é o peso da classe, $acurcia_{classe}$ é a taxa de acerto da classe, e $\sum_{c \in classes} acurcia_c$ é a soma da taxa de acerto de todas as classes.

E como foi mencionado anteriormente, a motivação de se atualizar os pesos é aumentar a probabilidade de treinar um novo classificador K com amostras da classe com uma baixa taxa de acerto na *pool*.

Retorne a *pool*: A *pool* final de classificadores é retornada.

CAPÍTULO 3

Análise dos Experimentos

3.1 Primeira Seção

CAPÍTULO 4

Análise das Métricas de Diversidade

4.1 This is a section

CAPÍTULO 5

Conclusão

Este volume foi tipografado em LaTeX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis).