Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

Combinação de Classificadores e Análise do Algoritmo ICS-Bagging

Guilherme Leite Moreira de Paiva

Trabalho de Graduação

Recife
31 de julho de 2015

Universidade Federal de Pernambuco
Centro de Informática

Guilherme Leite Moreira de Paiva

Combinação de Classificadores e Análise do Algoritmo ICS-Bagging

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
31 de julho de 2015

Dedico este trabalho a Deus por tudo. Aos meus familiares, todos eles, por terem me dado incentivo a educação. E à minha esposa por ter me dado todo o suporte e compreensão.

Agradecimentos

Agradeço primeiramente a Deus por possibilitar que tudo seja feito, e que esteve ao meu lado durante essa jornada. Agradeço ao meu pai, Leônidas Santos de Paiva por me servir de modelo e exemplo, agradeço a minha mãe Maria Augusta Leite Moreira por sempre me incentivar a estudar cada vez mais. Agradeço muito a minha avó paterna Maria Leonor Santos de Paiva por sempre me dar forças e me lembrar, a partir do seu exemplo, que se pode chegar aonde quiser. Agradeço a minha esposa, Verônica da Silva Cordeiro de Paiva por sempre estar do meu lado e ser minha companheira em todos os momentos. Agradeço ao professor George Darmiton por despertar em mim o interesse pela área de Aprendizagem de Máquina e por me despertar a sede pela pesquisa e pela ciência. Agradeço muito ao Dayvid Victor por ter me ajudado, orientado e liderado pelo exemplo.

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

Resumo

Em muitas situações de tomada de decisão é mais confiável consultar a opinião de várias pessoas para que uma decisão seja tomada de forma mais acertada. Analogamente, esse conceito foi transportado para aprendizagem de máquina sob a forma de combinação de classificadores. Várias maneiras de se combinar classificadores foram propostas, cada uma com suas particularidades, pontos fortes e fracos. Este trabalho tem como objetivo ter uma visão geral dos algoritmos de combinação de classificadores canônicos que servem de base para diversos outros algoritmos, bem como em especial o algoritmo ICS-Bagging. Este terá seus parâmetros analisados e comparados com os principais algoritmos de combinação de classificadores.

Palavras-chave: Combinação de Classificadores, Diversidade, Bagging, Boosting, Adaboost, ICS-Bagging.

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 **??**.

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

2.2.1 Origem do Algoritmo 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 AGGregatING*, 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. A diversidade necessária para fazer com que o conjunto funcione é obtida pelo fato de se utilizar diferentes conjuntos de treinamento. Idealmente, os conjuntos de treinamento devem ser criados randomicamente do conjunto de treinamento. E abaixo o algoritmo pode ser entendido, de forma que pode ser notado que este é um algoritmo paralelo nas fases de treinamento e operacional.

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. A figura **??** ilustra o funcionamento do algoritmo.

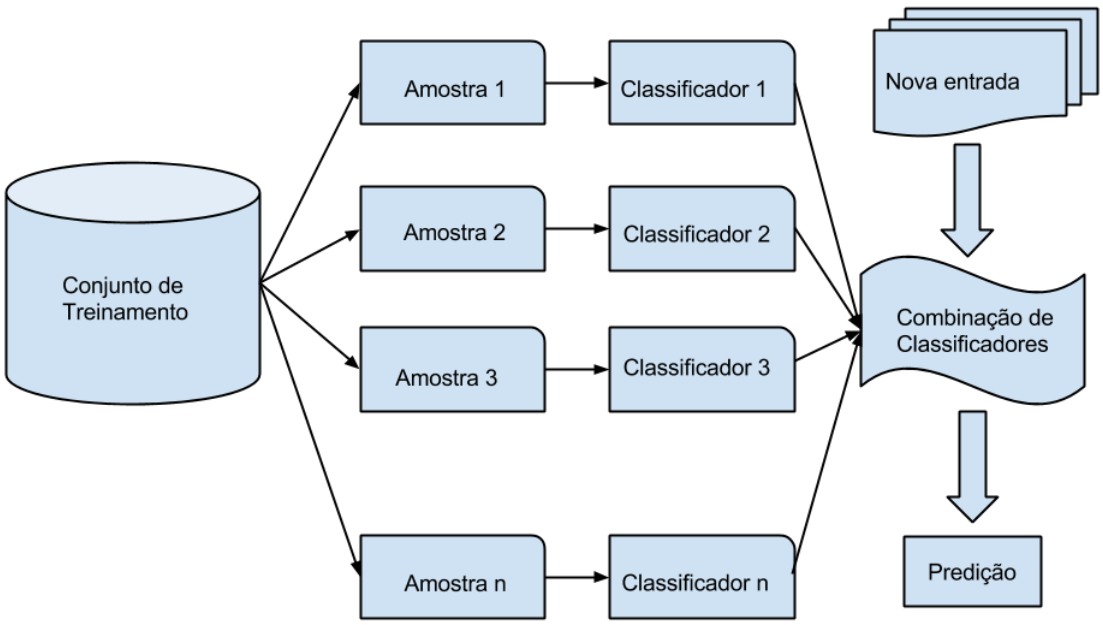

Figura 2.1 Bagging

2.2.2 A Razão do Bagging Funcionar

Se as saídas dos classificadores fossem independentes e os classificadores tivessem a mesma acurácia p, então a maioria do votos simples dos classificadores já garante um ganho com relação a um único classificador. O algoritmo Bagging tem como objetivo treinar classificadores independentes utilizando réplicas de amostras da base de treinamento. Estas réplicas são pseudo-independentes pelo fato de terem sido obtidas do mesmo conjunto de treinamento. No entanto, este fato não anula o fato das respostas dos classificadores individuais serem independentes.

2.2.3 Variantes do Algoritmo Bagging

As seções abaixo mostra algumas variantes do algoritmo têxtilbagging, bem como o seu funcionamento e definição.

2.2.3.1 Random Forest

Breiman propôs uma variante do algoritmo têxtilbagging [214] que o chamou de *random forest*. Este algoritmo é uma classe geral para os métodos de combinação de classificadores que

utilizam uma árvore de decisão como classificador base. Para ser rotulado como um "*random forest*"um conjunto de árvores de decisão deve ser construído gerando vetores aleatórios independentes e distribuídos de maneira idêntica, e utilizar cada vetor para crescer a árvore de decisão. Uma definição formal pode ser obtida abaixo.

Definição: *Random Forest* é um classificador consistindo de uma coleção de classificadores do tipo árvore de decisão. Cada árvore de decisão cresce respeitando um vetor aleatório onde cada elemento deste vetor é independente e identicamente distribuído. Cada árvore de decisão provê um único voto dado um entrada **x** e o resultado é a saída **y** que é a maioria simples dos votos. A figura **??** ilustra a definição.

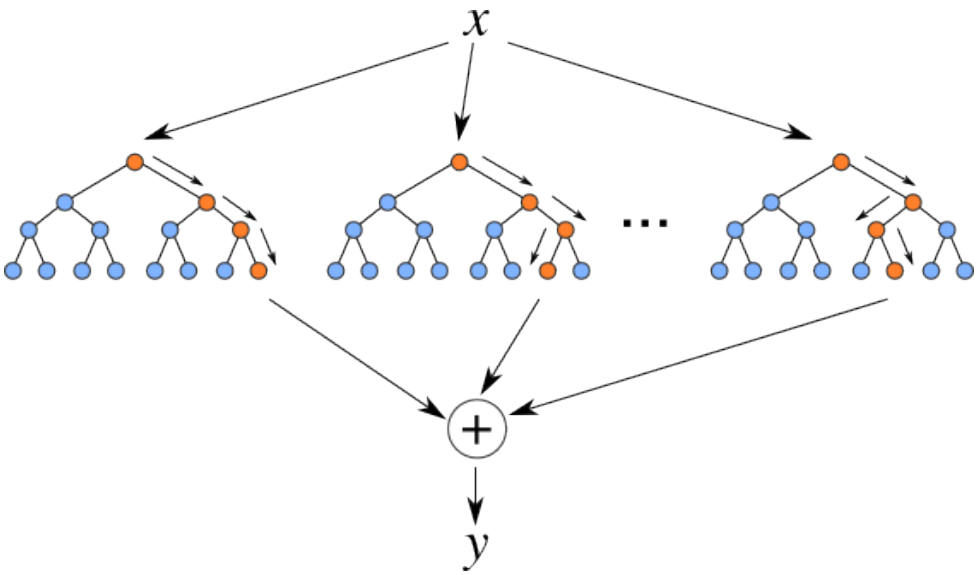

Figura 2.2 Random Forest

2.2.3.2 Pasting Small Votes

Bases de dados grandes são mais comuns atualmente do que no início do desenvolvimento dos primeiros algoritmos de combinação de classificadores. E a disponibilidade de aumento de capacidade computacional permitiu com que a manipulação dessa massa maior de dados disponível fosse possível. Breiam [215] então sugeriu o que ficou conhecido como *small votes*, na qual classificadores individuais são treinados com pequenos subconjuntos do conjunto de treinamento. Os conjuntos de treinamento são obtidos de forma aleatória do conjunto original. Caso esse conjunto seja obtido de forma similar ao *bagging* então este é denominado *RVotes*, e caso seja baseado em importância, é denominado de *IVotes*.*Pasting Small Votes* é adequado para aprendizado *on-line*, de forma que o conjunto de classificadores pode ser atualizado adicionando um novo classificador cada vez que uma nova quantidade de dados do conjunto de treinamento é acumulada. *Pasting RVotes* possui o como ponto positivo o fato de ser mais simples, no entendo perde em acurácia se comparado com *Pasting IVotes* [215, 216].

Algorithm 3 IVotes

1. **ENTRADA:** Conjunto de Treinamento: S, Número de Iterações: T, Tamanho do Bootstrap: n, Classificador fraco: I.
2. **SAÍDA:** O Classificador Final: $H(x) = sign\left(\sum_{t=1}^{T} h_t(x)\right)$ onde $h_t(x))$ onde $h_t(x) \in \{-1,+1\}$.
2. $e_n \leftarrow 0.5$
2. **Repita**
2. $e_o \leftarrow e_n$
2. $S_t \leftarrow \emptyset$
3. **while** *size* $S_t < n$ **do**
4. $x \leftarrow InstanciaAleatoria(S)$
5. **if** x for classificado de forma incorreta por um classificador fora do conjunto **then**
6. $S_t \leftarrow S_t \cup \{x\}$
7. **else**
8. $S_t \leftarrow S_t \cup \{x\}$ com probabilidade $(\frac{e_o}{1-e_o})$
9. **end if**
10. **end while**
10. $h_t \leftarrow I(S_t)$
10. $e_n \leftarrow$ erro do classificador fora do conjunto
10. **Até** $e_n < e_o$

2.2.3.3 Random Subspace

O método *Random Subspace* **??** consiste em utilizar vários classificadores cada um operando em um subespaço do espaço de características. De forma mais específica, o classificador consiste em se construir sistematicamente múltiplas árvores selecionando pseudorandomicamente subconjuntos dos componentes do vetor de características. Dessa formas, as árvores são construídas por esses subespaços escolhidos randomicamente. Este método, como já detalhado anteriormente, utiliza um subconjunto de características selecionadas aleatoriamente e atribui a um determinado algoritmo de aprendizagem. Dessa forma, é obtido um subespaço aleatório do espaço de características original, construindo assim os classificadores dentro desse espaço reduzido. A forma mais comum utilizada para se obter o resultado é por meio de um voto de cada classificador utilizando-se pesos. Isso se justifica por ter sido mostrado que este método é eficiente para classificadores que possuem um curva de aprendizado decrescente construída sobre um conjunto pequeno de treinamento. O conjunto de classificadores é construído como pode ser visto no algoritmo abaixo:

2.3 Boosting

2.3.1 Origem do Algoritmo Boosting

Podemos dizer que *Boosting* é um método geral para melhorar a precisão de uma classificação[219]. Este algoritmo foi desenvolvido por Schapire em 1990 [40 review] e o objetivo

Algorithm 4 Random Subspace

1. Considere um número de objetos de treinamento N e o número de características do conjunto de dados como sendo D.
2. Atribua L como sendo o número de classificadores do conjunto de classificadores.
3. Para cada classificador individual I, escolha d_i ($d_i < D$) como sendo o número de variáveis de entrada para I.
4. Para cada classificador individual I, crie um conjunto de treinamento escolhendo d_i características de D sem reposição e treine o classificador.
5. Para classificar um novo objeto, combine as saídas de cada classificador individual L utilizando a maioria dos votos com pesos.

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 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.

2.3.2 A Razão do Boosting Funcionar

Um ponto bastante positivo desta família de algoritmos é sua convergência rápida da taxa de erro do conjunto de treinamento para valores ditos como bons, praticamente nas primeiras iterações. O meio usado para criar as bases de dados consiste em usar uma distribuição balanceada de instâncias fáceis e instâncias difíceis [42 review]. As instâncias difíceis são detectadas pelos classificadores fora do conjunto. Uma instância é considerada difícil quando é classificada de forma incorreta pelo conjunto de classificadores formado por aqueles classificadores que não usaram esta instância no seu treinamento.

2.3.3 Variantes do Algoritmo Boosting

As seções abaixo mostra algumas variantes do algoritmo têxtilboosting, bem como o seu funcionamento e definição.

2.3.3.1 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.

2.3.3.2 AdaBoost.M1

2.3.3.3 AdaBoost.M2

2.4 Qual a melhor abordagem: *Bagging* ou *Boosting*

Em princípio este tipo de pergunta se apresenta de forma mal colocada partindo do pressuposto que não existe o melhor método para todos os casos. No entanto, vários autores têm comparado as duas abordagens e gerando vários resultados [106, 116, 236, 238, 239 livro]. O consenso geral é de que os métodos *boosting* alcançam uma menor taxa de erro. No entanto, algoritmos baseados em *boosting* são sensíveis a ruídos e *outliers*, especialmente quando a base de dados é pequena [116,236,239].

2.5 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 5 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: $\mathscr{L}$ que automaticamente aprende um classificador binário $h(x) : \mathbf{R}^d \mapsto \{-1, +1\}$ 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. **Algoritmo**
4. Inicialize a distribuição $D_0(i) = 1/N, i = 1, \ldots, N$
5. **for** $t = 1$ até T **do**
6. Escolha m exemplos com reposição de $\mathscr{D}$ 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 $I(z)$ dá como saída 1 quando z é verdadeiro e zero caso contrário.
9. Saia do laço 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

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

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

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

Algorithm 6 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 **??**, 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 **??**,

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

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.

Um esquema do fluxo de funcionamento do algoritmo ICS-Bagging pode ser visto na figura **??**.

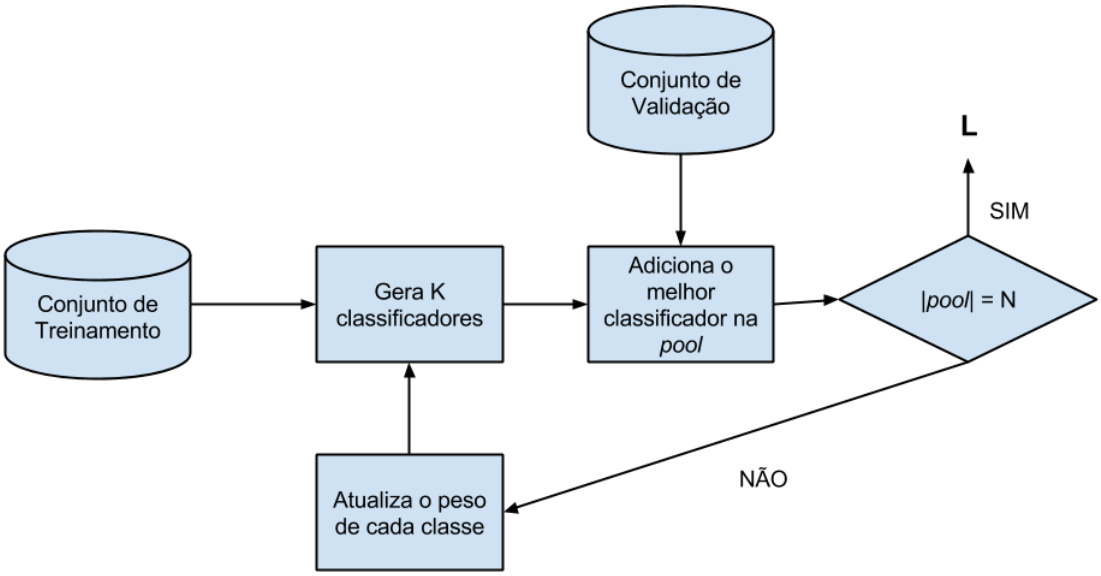

Figura 2.3 Arquitetura do ICS-Bagging. Onde L é o conjunto final de classificadores.

CAPÍTULO 3

Análise dos Experimentos

Esta seção apresenta a metodologia e os resultados dos experimentos feitos para a variação dos parâmetros da função de *fitness* do algoritmo ICS-Bagging.

3.1 Metodologia do Experimento Base

Os parâmetros do algoritmo ICS-Bagging foram avaliados utilizando 15 bases obtidas do repositório KEEL [31]. As bases de dados são binárias e possuem taxa de balanceamento crescente, taxa esta que é dada pelo número de exemplos da classe majoritária, divido pelo número de exemplos da classe minoritária. As bases utilizadas neste experimento estão descritas na Tabela **??**, que mostra o número de exemplos, o número de atributos, distribuição das classes e a taxa de balanceamento (IR - *Imbalance Ratio*).

Tabela 3.1 Características das Bases de Dados

Rótulo	Nome	Padrões	Atributos	% (min., maj.)	IR
1	Glass1	214	9	(35.51, 64.49)	1.82
2	Pima	768	8	(34.84, 66.16)	1.90
3	Iris0	150	4	(33.33, 66.67)	2.00
4	Yeast1	1484	8	(28.91, 71.09)	2.46
5	Vehicle2	846	18	(28.37, 71.63)	2.52
6	Vehicle3	846	18	(28.37, 71.63)	2.52
7	Ecoli1	336	7	(22.92, 77.08)	3.36
8	Ecoli2	336	7	(15.48, 84.52)	5.46
9	Glass6	214	9	(13.55, 86.45)	6.38
10	Yeast3	1484	8	(10.98, 89.02)	8.11
11	Ecoli3	336	7	(10.88, 89.12)	8.19
12	Vowel0	998	13	(9.10, 90.9)	9.98
13	Glass4	214	9	(6.07, 93.93)	15.47
14	Ecoli4	336	7	(5.95, 94.05)	15.8
15	Page-blocks13vs4	472	10	(5.93, 94.07)	15.85

As bases de dados foram particionadas utilizando o procedimento *5-fold cross validation*, que significa que cada conjunto de dados foi dividido em 5 subconjuntos (cada um com

20"%"das amostras) e os experimentos foram executados cinco vezes, cada iteração com um dos subconjuntos para teste e os demais para treinamento. A partição foi feta respeitando a proporção entre as classes.

As métricas de avaliação foram acurácia e diversidade. Para acurácia na classificação foi utilizado a área sobre a curva ROC (AUC). Para diversidade, nestes primeiros experimentos, foi utilizada a medida de entropia. Os parâmetros utilizados nos algoritmos podem ser conferidos na Tabela **??**.

Tabela 3.2 Algoritimos e parametros

Algoritmo	Parâmetro
ICS-Bagging	$N = 40$, *repetição = Sim*
	$K = 5$, $\alpha = 0.75$
Bagging	$N = 40$, *repetição = Sim*
RandomSubspace	$N = 40$, $features_{max} = 0.3$

O classificador base utilizado nos experimentos foi uma árvore de decisão de profundidade máxima 9, e o número mínimo de amostras necessárias para ser um nó folha como sendo 1. A regra de combinação foi a maioria dos votos simples.

3.1.1 Análise e Resultados do Experimento Base

A Tabela **??** mostra a média e o desvio padrão da AUC dos três algoritmos que são objetivo destes experimentos nas configurações propostas.

A Tabela **??** mostra a média e o desvio padrão e da medida de diversidade Entropia dos três algoritmos que são objetivo destes experimentos nas configurações propostas.

3.1.1.1 Conclusão

Como pode ser notado na Tabela **??**, o algoritmo ICS-Bagging apresentou a melhor média de AUC quando comparado com os algoritmos *Bagging* e *Random Subspace*. Já quando se compara os resultados obtidos da medida de entropia E, o algoritmo *Random Subspace* apresenta melhor média de diversidade. Isto se deve por conta que o algoritmo *Random Subspace* somente seleciona parte das características de cada classificador, e isto se mostra como sendo mais efetivo do que selecionar parte dos dados para cada classificador. Esta última informação é de suma importância por fazer um comparativo com a base do algoritmo *bagging*.

3.2 Metodologia dos demais Experimentos

Para os demais experimentos foram mantidas algumas considerações do experimento base, tais como: as mesmas 15 bases, com os dados binária e possuem taxa de balanceamento crescente. A configuração das bases pode ser conferida na Tabela **??**. No entanto, para o primeiro experimento foi utilizado LHS, que é um método estatístico para geração de coleções de parâmetros

Tabela 3.3 Média e Desvio Padrão da AUC

Conjunto de Dados	ICS-Bagging-40		Bagging-40		RandomSubspace-40	
glass1	**0.7923**	**0.0500**	0.7521	0.0529	0.7576	0.0816
pima	**0.7390**	**0.0194**	0.7215	0.0227	0.6328	0.0295
iris0	**1.0000**	**0.0000**	**1.0000**	**0.0000**	**1.0000**	**0.0000**
yeast1	**0.7309**	**0.0343**	0.6810	0.0187	0.5325	0.0148
vehicle2	0.9592	0.0168	0.9594	0.0225	**0.9633**	**0.0129**
vehicle3	**0.7368**	**0.0188**	0.6600	0.0226	0.6327	0.0278
ecoli1	**0.8722**	**0.0497**	0.8480	0.0454	0.7672	0.0887
ecoli2	**0.8855**	**0.0495**	0.8714	0.0464	0.7283	0.0357
glass6	**0.8917**	**0.0190**	0.8865	0.0624	0.8640	0.0836
yeast3	**0.9089**	**0.0273**	0.8434	0.0337	0.5030	0.0061
ecoli3	**0.8047**	**0.0967**	0.7724	0.0677	0.6279	0.0708
vowel0	0.9483	0.0573	**0.9589**	**0.0540**	0.9278	0.0572
glass4	**0.8750**	**0.1877**	0.7467	0.1654	0.6475	0.1362
ecoli4	0.8405	0.1143	**0.8671**	**0.1326**	0.7000	0.1000
page-blocks-1-3_vs_4	**0.9978**	**0.0045**	0.9766	0.0414	0.8733	0.0712
Average	**0.8655**	**0.0497**	0.8363	0.0526	0.7439	0.0544

Tabela 3.4 Média e Desvio Padrão da Medida Entropia *E*

Conjunto de Dados	ICS-Bagging-40		Bagging-40		RandomSubspace-40	
glass1	0.4445	0.0560	0.3786	0.0531	**0.4604**	**0.0347**
pima	0.4284	0.0133	0.4131	0.0196	**0.4966**	**0.0216**
iris0	0.0000	0.0000	0.0000	0.0000	**0.0073**	**0.0060**
yeast1	**0.4342**	**0.0230**	0.3410	0.0029	0.2727	0.0222
vehicle2	0.1006	0.0148	0.0886	0.0126	**0.1699**	**0.0204**
vehicle3	**0.3900**	**0.0219**	0.3661	0.0177	0.3349	0.0382
ecoli1	0.1542	0.0361	0.1456	0.0337	**0.2965**	**0.0475**
ecoli2	0.1556	0.0440	0.1216	0.0286	**0.2174**	**0.0281**
glass6	0.1108	0.0550	0.0613	0.0148	**0.1265**	**0.0313**
yeast3	0.0767	0.0076	0.0825	0.0086	**0.1158**	**0.0119**
ecoli3	0.1187	0.0364	0.1246	0.0290	**0.1710**	**0.0476**
vowel0	0.0346	0.0176	0.0299	0.0069	**0.0994**	**0.0093**
glass4	0.0684	0.0299	0.0944	0.0312	**0.1026**	**0.0363**
ecoli4	0.0460	0.0228	0.0333	0.0151	**0.0984**	**0.0284**
page-blocks-1-3_vs_4	0.0052	0.0065	0.0269	0.0090	**0.0524**	**0.0152**
Média	0.1712	0.0257	0.1538	0.0189	**0.2015**	**0.0266**

de uma distribuição multinomial [colocar ref da wiki]. Este método foi utilizado por ser bastante comum na geração de amostras de parâmetros com fins de experimentos computacionais. Este primeiro experimento teve como objetivo encontrar alguma relação entre os parâmetros da função de *fitness* do algoritmo ICS-Bagging.

3.3 Análise e Resultados do Primeiro Experimento - Geração de amostras utilizando LHS

Este primeiro experimento tem como objetivo gerar diversas configurações possíveis para K e $alfa$ com a finalidade de levantar alguma hipótese. Em conjunto com possíveis hipóteses a serem levantadas, será escolhida a melhor configuração e a mesma será analisada posteriormente.

3.3.1 Análise e Resultados

Como foi mencionado anteriormente, o primeiro experimento gerou uma série de configurações possíveis para 100 amostras utilizando o método LHS de forma que para cada amostra gerada é obtido um valor para o número de classificadores da *pool* K e o valor para o parâmetro alfa da função de *fitness*. Para cada configuração gerada, ou seja, cada combinação de valor de alfa e de K foi calculada a média e o desvio padrão das 15 bases que são objetivo deste trabalho. O apêndice **??** detalha todas as combinações utilizadas neste primeiro experimento bem como os resultados da média e desvio padrão. E a Figura **??** mostra os 10 melhores resultados dos 100 gerados.

Tabela 3.5 Dez melhores configurações utilizando LHS

Experimento	Alfa	K	Média	Desvio Padrão
Experimento 5	0.0505791514132	9	0.8804600000000	0.0848489937084
Experimento 26	0.262644897321	12	0.8799800000000	0.0834441066423
Experimento 2	0.0202494447442	13	0.8799133333330	0.0854376253311
Experimento 21	0.212149809235	7	0.8794800000000	0.0852588231993
Experimento 11	0.111150784393	14	0.8788533333330	0.0851454588076
Experimento 44	0.444499921896	17	0.8788533333330	0.0848942625990
Experimento 43	0.434370633338	18	0.8784800000000	0.0859109399320
Experimento 15	0.151536054488	17	0.8783933333330	0.0858358469535
Experimento 52	0.525291411015	11	0.8779200000000	0.0838991871236
Experimento 23	0.232315494974	12	0.8773866666670	0.0832561738785

Para um melhor entendimento destes resultados, a Figura **??** mostra os resultados da AUC relacionado com o parâmetro alfa. E pode ser notado que a AUC obedece uma tendência de baixa quando o alfa cresce, independente do valor do K.

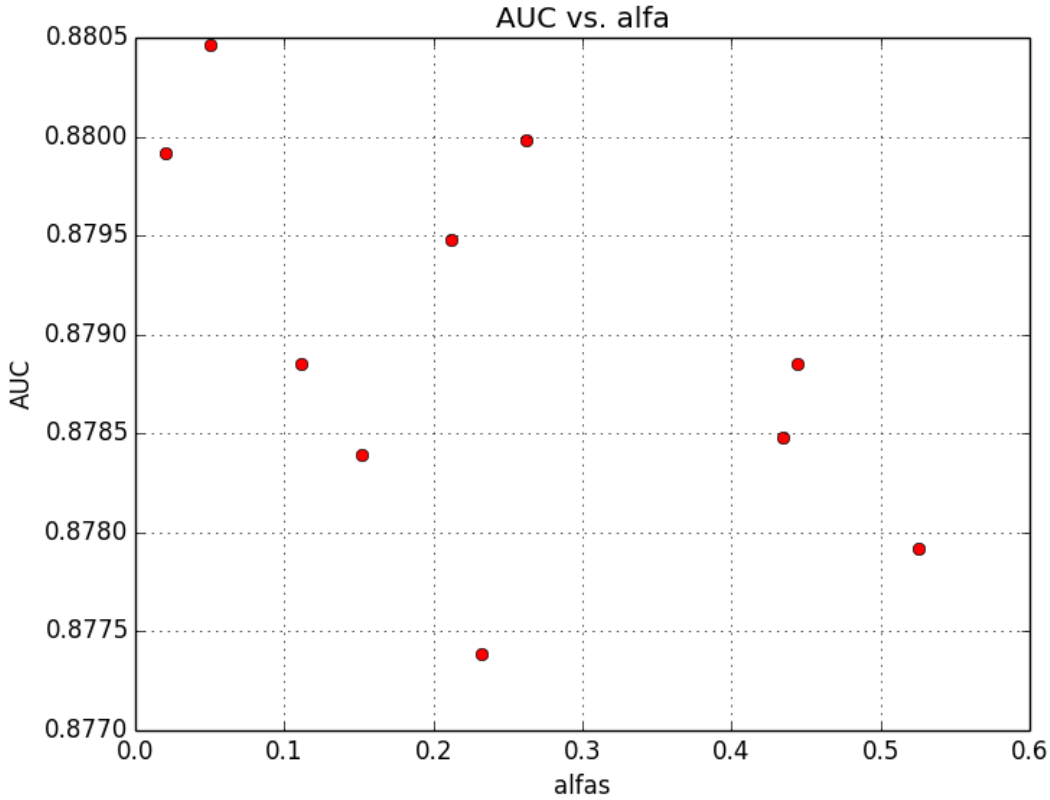

Figura 3.1 AUC x Alfa

Por outro lado, a Figura **??** mostra os resultados da AUC relacionado com o valor de K. E nota-se a ausência de um padrão bem definido com relação aos parâmetros analisados.

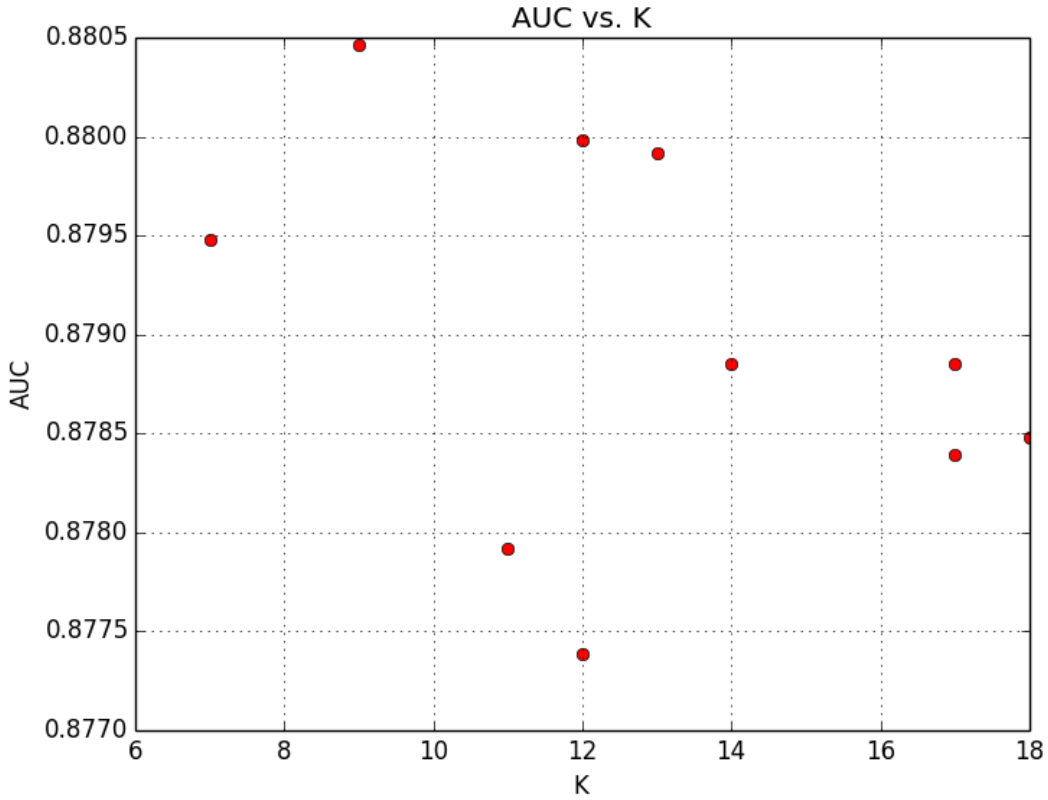

Figura 3.2 AUC x K

3.3.1.1 Conclusão - Geração de amostras utilizando LHS

Há de se ter cuidado ao inferir resultados a partir dos dez melhores resultados e generalizar para toda a amostra e até mesmo é casos mais gerais e amostras até maiores. No entanto, ao desenvolver a mesma tarefa dos dez melhores resultados a base inteira de cem resultados as hipóteses ainda se confirmam, mesmo que de forma um pouco menos clara. Algum ruído pode ser observado e um padrão exato em todos os pontos não pode ser observado para o caso do decrescimento da AUC ao se diminuir o parâmetro alfa, porém a tendência de baixa de mantêm. A Figura **??** mostra toda as cem experimentações relacionando a AUC e o parâmetro alfa. Para o caso do valor de K nenhum padrão pode ter sido estabelecido, no entanto, valores de K muito próximos a 20 possuem um decaimento abrupto no valor da AUC que precisa ser investigado de forma mais precisa. A Figura **??** mostra os resultados da relação entre a AUC e o valor de K.

Importante salientar que a Figura **??** e a Figura **??** foram construídas utilizando curvas *splines* para que houvesse interpolação entre os dados e algum tipo de inferência quanto aos resultados pudesse ser tomado.

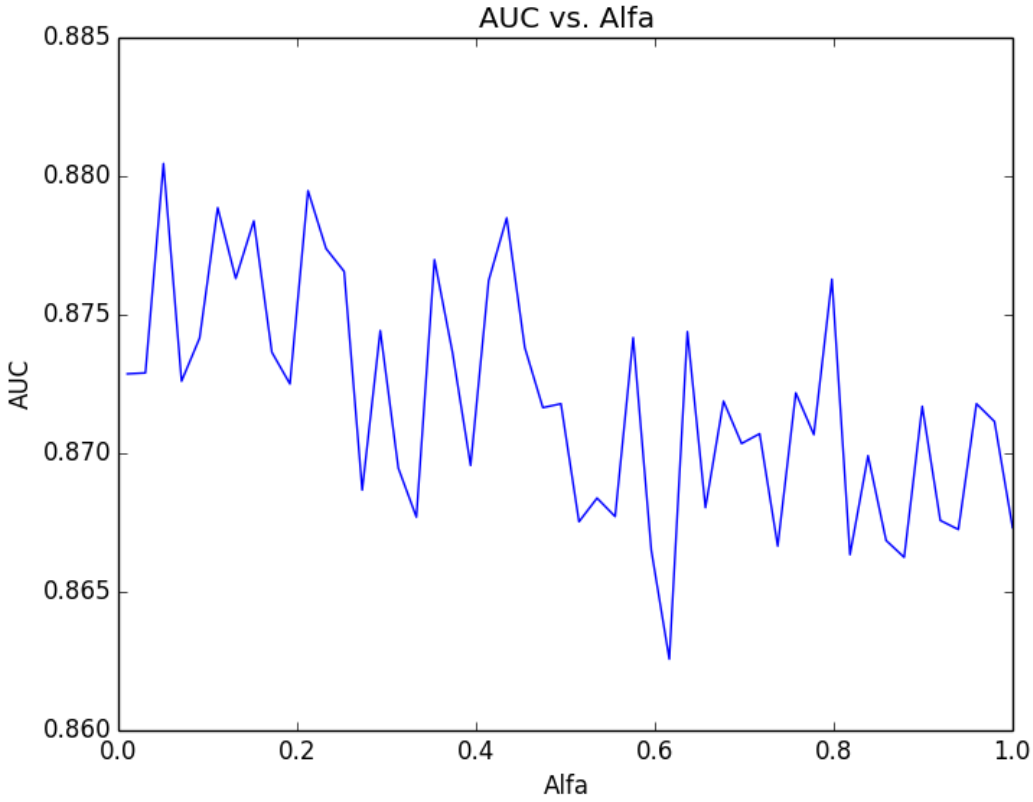

Figura 3.3 AUC x Alfa

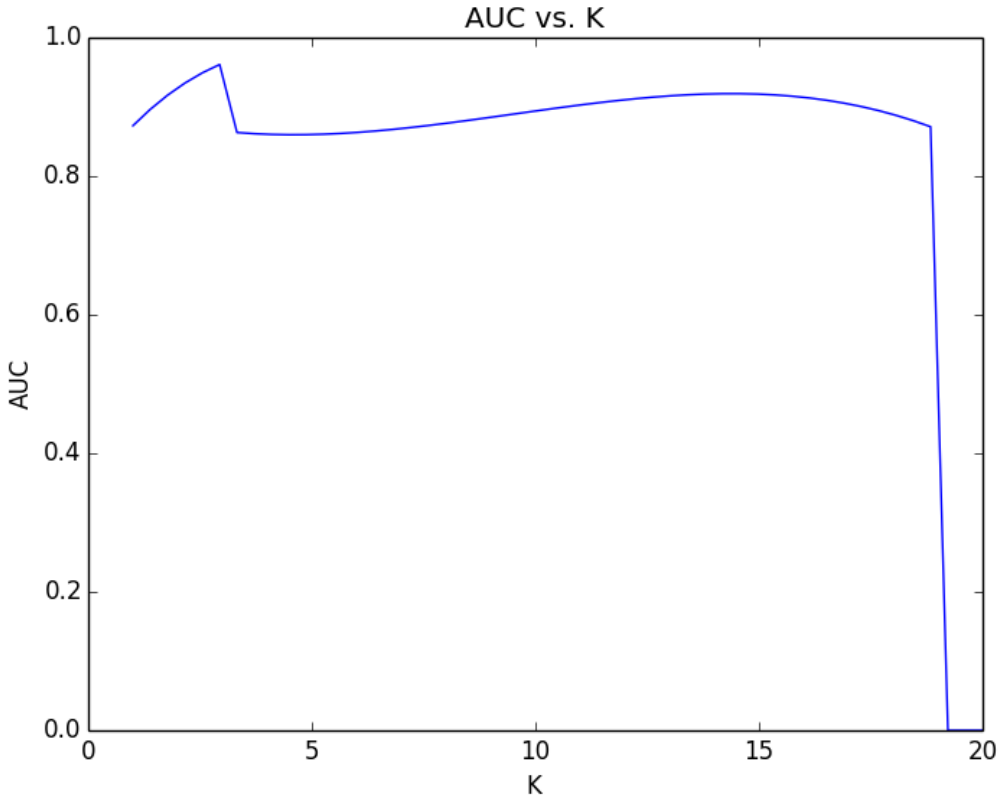

Figura 3.4 AUC x K

3.4 Análise e Resultados do Segundo Experimento - *K* Fixo em 5

O objetivo deste experimento é analisar as implicações da variação do parâmetro $alfa$ quando temos a variável *K* fixada.

3.4.1 Análise e Resultados

Neste segundo experimento, o valor do parâmetro *K* foi fixado em 5. Desta forma, o valor de *K*, que é o número de classificadores a serem adicionados a cada iteração é fixado para que possa ser analisado o comportamento do valor de alfa. Por isso, o valor de alfa será analisado variando-o de 0.0 até 1.0. O resultado deste experimento pode ser visto na Tabela **??**.

Como pode ser visto na Figura **??**, a AUC decresce com o aumento do valor de alfa. Esse decréscimdo se dá de forma suave ao se olhar de uma perspectiva mais ampla. Um consequência direta deste fato é que como o alfa é um parâmetro da função de *fitness* e define a proporção entre diversidade e acurácia da *pool* de classificadores, então valores para alfa menor dão uma ênfase maior ao componente diversidade da função de *fitness*. Lembrando que a função de *fitness* é definida como abaixo:

Tabela 3.6 Resultado dos Experimentos para K fixo em 5 e Alfa variando de 0 a 1

Alfa	Média da AUC	Desvio Padrão
0.0	0.87388	0.0852539907179
0.1	0.873333333333	0,0856087897875
0.2	0.87086	0.0904865500134
0.3	**0.87504**	**0.0869947569301**
0.4	0.8695	0.0890983501531
0.5	0.864113333333	0.0872420530606
0.6	0.86806	0.0891251838708
0.7	0.870633333333	0.0883060712648
0.8	0.866206666667	0.0907677950719
0.9	0.867326666667	0.0884994347753
1.0	0.865386666667	0.0898583022814

$$fitness = \alpha \times ACC + (1 - \alpha) \times DIV \tag{3.1}$$

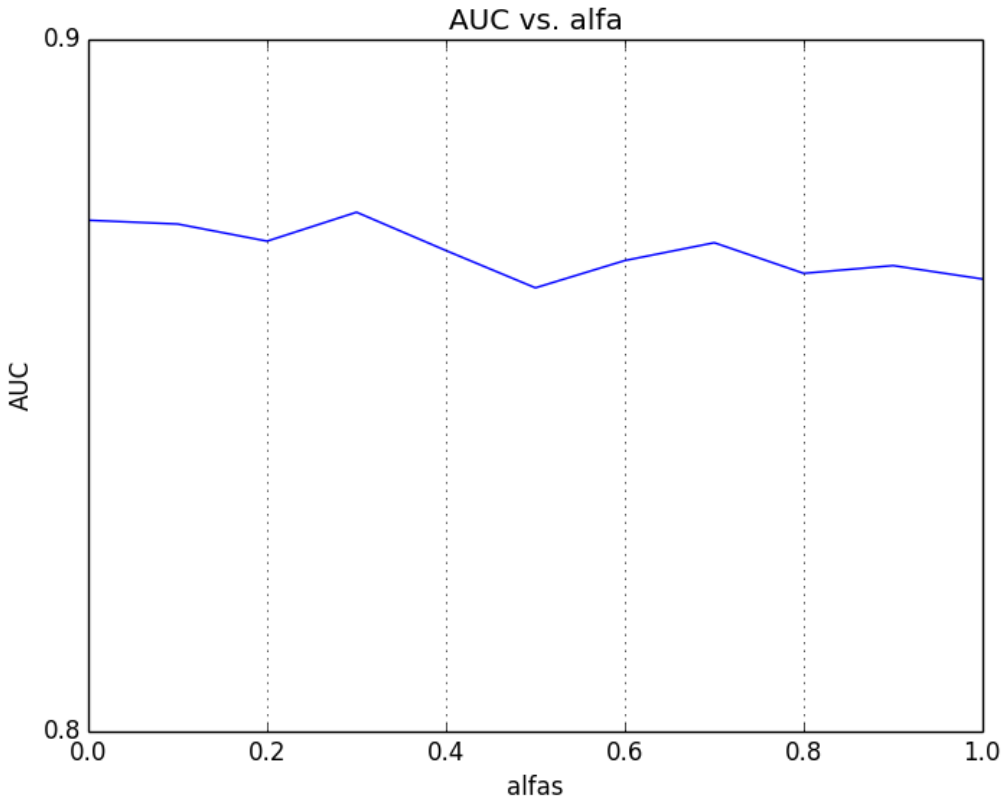

Figura 3.5 AUC x Alfa

Por outro lado, este enfoque da Figura **??** não ressalta informações mais sensíveis. De forma que alguns comportamentos dos dados não são percebidos com uma visão distante da informação. A Figura **??** mostra a distribuição deste experimento com um nível maior de detalhamento. Nela pode ser notado que obedece aos comentários já tecidos anteriormente que mencionam a tendência de decréscimo do valor da AUC quando o valor de $alfa$ diminui. No entanto, três pontos em particular do gráfico chamam a atenção. O ponto onde a AUC é maior é para o valor de alfa 0.3, e o ponto onde a AUC volta a aumentar é para o valor de alfa igual a 0.7. Levantando uma hipótese sobre a distribuição da diversidade e da acurácia da função de *fitness* ter um comportamento proporcional entre 30% e 70% de distribuição.

Mais importante ainda, é a informação obtida de que quando o alfa se torna igual a 0.5 a AUC tem o pior resultado. Evidenciando que ao colocar a mesma proporção na função de *fitness* para a diversidade e para a acurácia o resultado foi o pior dentre os demais.

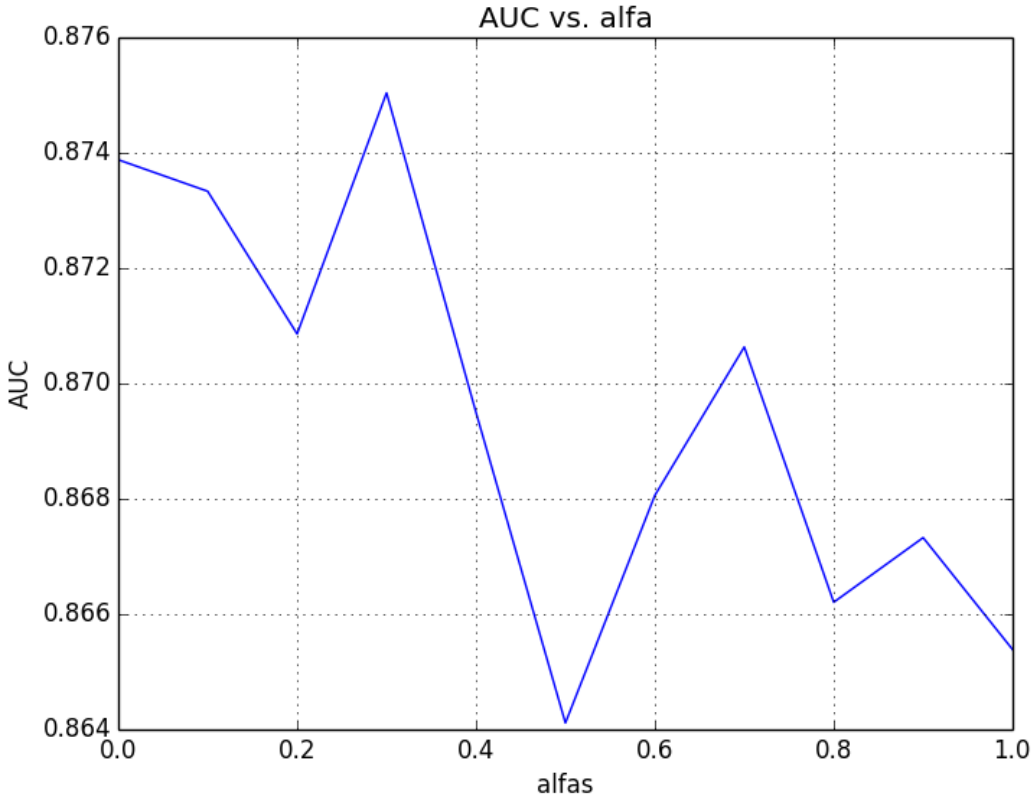

Figura 3.6 AUC x Alfa

3.4.2 Conclusão - K Fixo em 5

Partindo deste segundo experimento algumas informações relevantes puderam ser extraídas. Quando a proporção do alfa torna os pesos iguais para diversidade e acurácia na função de *fitness*, ou seja, o alfa é igual a 0.5, então o valor da AUC é o pior possível. Desta forma,

manter proporções iguais para as duas medidas não oferece vantagem.

Outra informação importante foi obtida notando a proporção de 30% e 70% para alfa obteve resultados bons. Melhor quando a proporção foi de 30%, evidenciando a importância da diversidade neste algoritmo.

3.5 Análise e Resultados do Terceiro Experimento - $alfa$ Fixo em 0.05

O objetivo deste experimento é analisar as implicações da variação do parâmetro K quando temos a variável $alfa$ fixada. Desta forma é possível observar a importância do parâmtro K no algoritmo ICS-Bagging. O valor de $alfa$ foi fixado em 0.05 por ter sido o melhor resultado obtido no primeiro experimento.

3.5.1 Análise e Resultados

Neste terceiro experimento o valor de K foi analisado entre os valores de 1 a 10. A Tabela **??** mostra os resultados deste experimento, onde pode ser notado que o melhor resultado obtido foi para o valor de $K = 10$. Mas não somente isso, o valor de K se relaciona de maneira crescente com a AUC, ou seja, quanto maior o valor de K maior a AUC.

Tabela 3.7 Resultado dos Experimentos para alfa fixo em 0.05 e Alfa variando de 1 a 10

K	Média da AUC	Desvio Padrão
1	0.868446666667	0.0883055971549
2	0.87052	0.0864033811067
3	0.86972	0.0865383013469
4	0.86882	0.0888397148427
5	0.873033333333	0.0871429298464
6	0.873146666667	0.0862880437192
7	0.877193333333	0.0856610332778
8	0.8763	0.0850978730639
9	0.87224	0.0874269889679
10	**0.877506666667**	**0.0845944715819**

A relação entre a AUC e o valor de K para o alfa fixo em 0.05 pode ser melhor percebido na Figura **??**, onde pode ser visto que quando maior o valor do K maior a AUC.

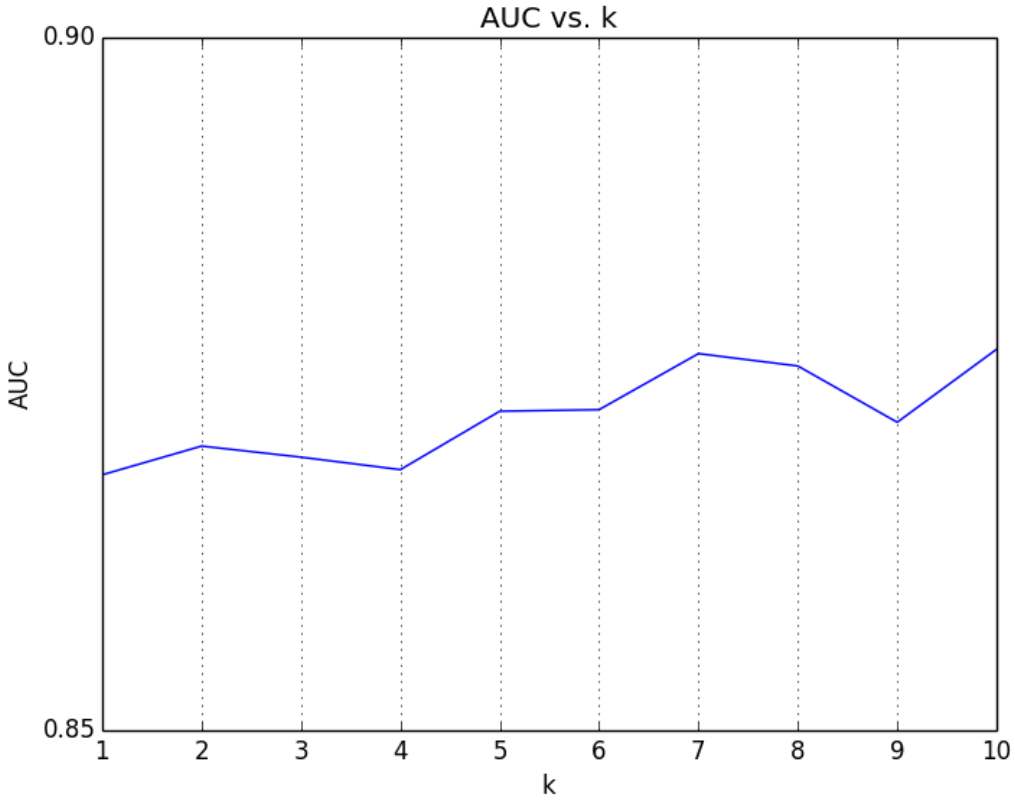

Figura 3.7 AUC x Alfa

3.5.2 Conclusão - alfa Fixo em 0.05

Após realizar os experimentos com o $alfa$ fixado em no valor de 0.05 e a variável K variando entre 1 e 10, pode ser concluído que quanto maior o valor do K, maior o valor da AUC. Ou seja, o número de classificadores a serem adicionados a cada iteração possui um fator relevante nos resultados, de forma que quanto maior o número de classificadores adicionados a cada iteração, melhores são os resultados obtidos.

3.6 Análise e Resultados do Quarto Experimento - Número de Classificadores reduzido e *K* fixo

Nesse quarto experimento será usado o número de classificadores igual a 10, o valor de k será fixado em 5 e o alfa será incrementado de 0.0 a 1.0. O objetivo deste e do próximo experimento é tentar avaliar de forma mais perceptiva as hipóteses e conclusões que surgiram nas seções anteriores. Ao se reduzir o número de classificadores espera-se que seja enfatizado a correlação de alfa com AUC e de K com AUC.

3.6.1 Análise e Resultados

3.6.2 Conclusão - Número de Classificadores reduzido e alfa variando

3.7 Análise e Resultados do Quinto Experimento - Número de Classificadores reduzido e $alfa$ fixo

Nesse quinto experimento será usado o número de classificadores igual a 10, o valor de k será variado de 1 a 5 e o alfa será fixo em (melhor resultado do quarto experimento)

3.7.1 Análise e Resultados

3.7.2 Conclusão - Número de Classificadores reduzido e alfa variando

CAPÍTULO 4

Análise das Métricas de Diversidade

4.1 This is a section

CAPÍTULO 5

Conclusão

5.1 Considerações Finais

5.2 Trabalhos Futuros

Tabela 1: Geração de configurações utilizando LHS

Experimento	Alfa	K	Média	Desvio Padrão
Experimento 1	0.0101879674065	1	0.8728666666670	0.0830612036727
Experimento 2	0.0202494447442	13	0.8799133333330	0.0854376253311
Experimento 3	0.0303879540906	3	0.8729000000000	0.0837933887607
Experimento 4	0.0404295120423	16	0.8764933333330	0.0893906965082
Experimento 5	0.0505791514132	9	0.8804600000000	0.0848489937084
Experimento 6	0.0606302362919	10	0.8714666666670	0.0877320440635
Experimento 7	0.0707364401375	2	0.8725800000000	0.0827474809284
Experimento 8	0.0808359392659	5	0.8758400000000	0.0883625546635
Experimento 9	0.0909098347286	15	0.8742200000000	0.0904861698456
Experimento 10	0.101084947576	1	0.8676000000000	0.0858640242865
Experimento 11	0.111150784393	14	0.8788533333330	0.0851454588076
Experimento 12	0.121203928708	16	0.8713333333330	0.0844698578718
Experimento 13	0.131391246699	6	0.8763066666670	0.0849548544163
Experimento 14	0.14142815941	10	0.8695800000000	0.0891007160465
Experimento 15	0.151536054488	17	0.8783933333330	0.0858358469535
Experimento 16	0.161683749899	1	0.8670400000000	0.0883582540947
Experimento 17	0.171794983089	10	0.8736466666670	0.0840568606493
Experimento 18	0.18182314747	12	0.8666933333330	0.0867517566905
Experimento 19	0.191953071203	13	0.8724933333330	0.0868531209700
Experimento 20	0.20202032439	14	0.8714266666670	0.0894172501379
Experimento 21	0.212149809235	7	0.8794800000000	0.0852588231993
Experimento 22	0.222250297053	6	0.8714733333330	0.0882491055038
Experimento 23	0.232315494974	12	0.8773866666670	0.0832561738785
Experimento 24	0.242489677696	14	0.8697533333330	0.0862585714903
Experimento 25	0.252526175921	8	0.8765000000000	0.0859753840740
Experimento 26	0.262644897321	12	0.8799800000000	0.0834441066423
Experimento 27	0.272779646992	5	0.8686733333330	0.0860332917474
Experimento 28	0.282893240833	18	0.8744000000000	0.0841846541835
Experimento 29	0.292927160975	13	0.8744266666670	0.0871476101540
Experimento 30	0.303097684837	10	0.8750933333330	0.0844166845015
Experimento 31	0.313154243419	5	0.8694600000000	0.0889654449772
Experimento 32	0.323244969072	14	0.8747133333330	0.0839726770378
Experimento 33	0.333354999903	2	0.8677066666670	0.0858639502676
Experimento 34	0.343426120358	12	0.8735133333330	0.0876904773748
Experimento 35	0.353546380037	11	0.8770200000000	0.0869266602756
Experimento 36	0.363689059168	9	0.8648600000000	0.0927051656238
Experimento 37	0.373739655984	19	0.8736066666670	0.0849283499323
Experimento 38	0.383881024187	4	0.8708666666670	0.0880550181547
Experimento 39	0.393926963115	2	0.8695733333330	0.0857322534924
Experimento 40	0.404072099479	9	0.8691266666670	0.0884403562986
Experimento 41	0.414163830726	14	0.8762400000000	0.0879169441386
Experimento 42	0.424260565828	10	0.8677866666670	0.0881191380399
Experimento 43	0.434370633338	18	0.8784800000000	0.0859109399320
Experimento 44	0.444499921896	17	0.8788533333330	0.0848942625990
Experimento 45	0.454543074708	7	0.8738466666670	0.0858046841508
Experimento 46	0.464642078098	2	0.8668533333330	0.0853567092982
Experimento 47	0.474769460863	17	0.8716400000000	0.0870681633358
Experimento 48	0.484821543591	11	0.8730933333330	0.0888088469067
Experimento 49	0.494929170176	6	0.8718066666670	0.0892997459247
Experimento 50	0.505057082284	9	0.8666800000000	0.0924101019009