ALAN GLEZER

Nota final
8,8 (oito e oito)

CONSTRUÇÃO DE CLASSIFICADORES A PARTIR DE BASES INCOMPLETAS UTILIZANDO O ALGORITMO CO-TRAINING

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do título de Engenheiro

São Paulo
2005

ALAN GLEZER

CONSTRUÇÃO DE CLASSIFICADORES A PARTIR DE BASES INCOMPLETAS UTILIZANDO O ALGORITMO CO-TRAINING

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do título de Engenheiro

Área de concentração:
Engenharia Mecatrônica

Orientador: Prof. Dr. Fábio G. Cozman

São Paulo
2005

ALAN GLEZER

CONSTRUÇÃO DE CLASSIFICADORES A PARTIR DE BASES INCOMPLETAS UTILIZANDO O ALGORITMO CO-TRAINING

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do título de Engenheiro

São Paulo,
Dezembro de 2005

ABSTRACT

Due to the cost of obtaining labeled data for training classifiers, we begin to see the use of algorithms that are able to learn from unlabeled data instead of learning only from previous labeled data. Created in 1998, an algorithm known as Co-Training is the main scope of study in this project. Using an implementation already developed for generating classifiers Naïve Bayes, a modification was implemented to deal with two parallel classifiers. Dealing with two views of the data, a combined classifier is created providing a better result that each one individually. Not only the Co-Training algorithm has been created, but also an interface for testing was developed. Finally, a wide range of datasets were used to validate the effectiveness of the hybrid classifier compared to other known methods of classification.

RESUMO

À medida que observamos o cresimento estrondoso da quantidade de informação produzida pela humanidade, cresce a necessidade de organização destas informações para que possam ser de alguma valia. Entretanto, a classificação tornou-se uma tarefa complexa com o aumento da complexidade dos dados gerados e suas relações.

Médicos precisam estabelecer relações complexas entre diversos exames para chegar a uma conclusão fundamentada. Robos precisam tomar decisões baseadas em leituras de diversos sensores, cuja relação entre si não é clara. Softwares precisam decidir se um email é desejado ou indesejado (spam). Empresas precisam classificar clientes de modo a otimizar seus serviços e se manter a frente de seus conconrentes. Todos estes exemplos citados representam áreas onde se poderia aplicar classificadores para o auxilio na tomada de decisões. Observa-se deste modo a vastidão de possiveis areas de aplicação.

Dentre as dificuldades de uso de classificadores gerados por computadores, destaca-se o problema de obtenção de dados já classificados para o treinamento dos classificadores. Devido a este necessidade, surgiram alguns algoritmos que possuem a capacidade de aprender com dados não rotulados, melhorando desta forma a eficácia da classificação.

Neste trabalho estudar-se-á o desempenho de um algoritmo para geração de classificadores conhecido por Co-Training. Para tal criar-se-á um sistema de automatização de testes e utilizar-se-á bases disponiveis na internet. Consi

Índice

1 Introdução

A quantidade de informação gerada pela humanidade cresce de forma estrondosa ao longo dos anos, de tal modo que classificar tais informações tornou-se uma tarefa trivial para uma possível analise das informações. Complexa e dispendiosa, a análise de dados apresenta-se como escopo de diversos estudos acadêmicos. Pesquisadores, empresas, famílias, todos possuem a necessidade de classificar a informação que lhes é provida.

A classificação de dados não pode ser considerada um advento recente, porém podemos observar uma grande diferença entre os grandes armários de arquivo encontrados nos corredores das empresas durante as últimas décadas e os complexos sistemas de informação disponíveis atualmente. Apesar de mais organizados pelos sistemas de informação, os dados são mais volumosos e suas relações mais complexas.

Estendendo-se por todas as áreas do conhecimento, a classificação eficaz de dados demonstra cada vez mais seu incalculável valor para a humanidade. Na medicina observamos classificações de tumores baseados na analises de exames, que apresentam uma complexa relação entre si. Leituras complexas de diversos sensores podem ser auxiliadas por classificadores para que uma inteligente tomada de decisão possa ser tomada.

O custo da classificação de dados está diretamente relacionado com a mão-de-obra humana necessária para realizar a classificação. Tal fato deve-se à necessidade da realização de um treinamento de sistemas automatizados de classificação utilizando dados previamente rotulados. Métodos que demandam uma quantidade menor de dados rotulados para classificar com um erro considerável uma grande quantidade de registros são um dos principais tópicos estudados por pesquisadores da área.

Os primeiros algoritmos utilizados para classificação de bases de dados eram estáticos, haja vista que o classificador era mantido intacto conforme realizadas grandes quantidades de rotulações. Dentre tais

algoritmos, destacamos o conhecido Naive Bayes, a ser utilizado para como base de comparação.

Dentre diversos fatores, consideramos relevante o alto custo para a obtenção de dados rotulados como um dos motivadores do desenvolvimento de uma nova classe de algoritmos que possui a capacidade de aprender conforme classifica novos dados. Deste modo é possível realizar uma classificação eficaz, dados somente poucos registros rotulados.

O objetivo deste trabalho será realizar testes em diversas bases de dados, de forma a verificar a eficácia do algoritmo conhecido por Co-Trainning. Desenvolvido em 1998 por Blum e Mitchell, tal método para geração de classificadores tornou-se cada vez mais popular.

Visando realizar uma verificação condizente do algoritmo, utilizar-se-á bases deveras variadas em conteúdos e áreas de aplicação.

O escopo deste trabalho será também o desenvolvimento de uma interface de testes mais amigável e prática para a implementação e uso do algoritmo de classificação desenvolvido. Encontra-se também dentro do objetivo deste projeto a realização de testes para bases que se enquadram e não se enquadram nos requisitos do Co-Training para que seja feita uma analise condizente.

2 O problema da classificação

É de grande interesse prático e recente objeto de extensivos estudos a construção de classificadores eficientes de bases de dados. Essa necessidade surge pelo fato de ser muito custoso em termos de tempo despendido classificar (rotular) manualmente as amostras de dados uma a uma.

A importância relacionada com a classificação de dados cresce de forma assombrosa com o passar dos anos. Empresas desejam cada vez mais segmentar seus clientes. Maquinas tornam-se cada vez mais independentes e, portanto necessitam da habilidade de classificar o que se encontra ao seu redor. Fenômenos da natureza começam a ser compreendidos e classificados de maneira lógica. A habilidade de classificar é essencialmente humana, porém é possível implementar algoritmos que simulam o modo de pensar de uma pessoa.

Dado um conjunto de amostras $\boldsymbol{X}$, e um conjunto de rótulos, daqui por diante chamados de classes, $\boldsymbol{Y}$, construir um classificador significa encontrar uma função $g(X_i)$ que retorna uma classe Y_i pertencente a $\boldsymbol{Y}$, que é a classe atribuída pelo classificador àquela amostra X_i.

Se supusermos conhecidas as classes verdadeiras das amostras, o melhor classificador g^* é aquele que minimiza a probabilidade de erro de classificação, ou seja,

$$g^* = argmin\ P(\ g(X)\ !=\ Y),$$

Com erro:

$$e^* = min\ P(\ g(X)\ !=\ Y).$$

Dado que existe uma distribuição de probabilidades $p(X, Y)$ sobre as amostras e classes, os casos são:

1ª) Estimar $p(X,Y)$ e construir o classificador usando $p\hat{}(X,Y)$;

2ª) Construir um classificador que "funcione bem" para qualquer (ou quase todas) $p(X,Y)$.

3ª) Obter $g(X)$ com alguma representação que não exija estimação "completa" de $p(X,Y)$.

Nesse trabalho, estaremos interessados no primeiro caso.

Para a estimação da distribuição de probabilidades, precisamos, em princípio, de uma base de treinamento. Essa base de treinamento possui amostras já classificadas, e estas serão usadas para "ensinar" o classificador o padrão de comportamento das amostras de cada classe.

A estimação dessas probabilidades utilizando amostras classificadas é o que chamamos de *aprendizagem supervisionada*. O método que apresentaremos para este caso é o Naive Bayes.

No entanto, o foco principal desse trabalho está em explorar como bases de dados não classificadas podem nos ajudar na construção desses classificadores. Isso é a chamada *aprendizagem não-supervisionada*. O método mais comum para este tipo de aprendizagem é o chamado *EM (Expectation-Maximization)*, que será detalhado mais à frente.

O que propomos neste trabalho é a utilização de outro método de aprendizagem chamado *Co-Training*. Esse método foi proposto em 1998 por Avrim Blum e Tom Mitchell, no artigo *Combining Labeled and Unlabeled Data with Co-Training*. Ele se baseia no fato que para algumas aplicações, as amostras podem ser vistas de duas maneiras distintas, independentes entre si, sendo que cada visualização por si só já é suficiente para a correta classificação. Esse método também será mais detalhado à frente.

3 Naive Bayes

Esta seção apresenta o Naive Bayes, um classificador probabilístico bem conhecido. Naive Bayes é a base sobre a qual iremos mais tarde elaborar para incorporar dados incompletos. A tarefa de aprendizagem aqui é estimar os parâmetros de um modelo generativo de dados usando dados de treinamento completos apenas. O algoritmo usa estes parâmetros estimados para classificar novas amostras calculando qual classe é a mais provável.

A premissa básica deste método consiste na suposição que todos os atributos das amostras são condicionalmente independentes entre si, dada a classe. Por conseqüência, cada amostra é independente das outras, dada a sua classe, ou seja:

$$P(\mathbf{x}|C_i) = \prod P(x_j|C_i)$$

Aplicando o teorema de Bayes,

$$P(C_i|\mathbf{x}) = [(P(C_i)) / (P(\mathbf{x}))] \prod P(x_j|C_i)$$

O termo $P(x)$ pode ser ignorado já que não depende da classe,

$$P(C_i|\mathbf{x}) \ \alpha \ P(C_i) \prod P(x_j|C_i)$$

Para cada classe é calculado um valor proporcional a $P(C_i|x)$. Uma previsão determinística pode ser obtida escolhendo-se a classe mais provável.

As probabilidades $P(C_i)$ e $P(x_j|C_i)$ são estimadas simplesmente fazendo-se a contagem de amostras na base de treinamento que, respectivamente, são da classe C_i, e daquelas que pertencem a essa classe, possuem o valor do atributo j como x_j.

A vantagem desse método está na sua simplicidade, mas ele tem um problema fundamental, para que se construa um bom classificador, faz-se necessário um elevado número de amostras classificadas. Isto demanda

tempo e, por conseqüência, custo. Esse problema, inclusive, é o que motiva o aparecimento de métodos que utilizam as próprias amostras não classificadas, que são muito menos custosas de se obter, para melhorar a eficiência dos classificadores.

4 EM (Expectation-Maximization)

Naive Bayes se mostra muito eficiente na maioria das aplicações a qual é destinado, mas somente se dispuser de vasta quantidade de dados classificados para com eles aprender. Mas sabe-se que o custo de obtenção de uma vasta quantidade de amostras classificadas (*labeled*), é extremamente elevado, daí observamos o grande problema da dificuldade de se obter dados classificados. No caso de uma empresa, o custo para a realização de pesquisas para classificar clientes demanda tempo e dinheiro. No caso de fenômenos da natureza, muitas vezes a classificação é inviável devido a fatores geográficos. O mesmo é válido para robôs, que podem estar muito distantes.

. Diante de tal dificuldade, foram desenvolvidos algoritmos que utilizam muitos dados não classificados (*unlabeled*) juntamente com poucos dados classificados para construir-se o classificador. Um método muito popular para a geração de classificadores a partir dessa mistura é o EM.

O EM é uma classe de algoritmos iterativos para estimativas de máxima verossimilhança ou máximo *a posteriori* em problemas com dados incompletos, que neste caso são os dados sem classificação. Ele é uma extensão do Naive Bayes no sentido que o utiliza diversas vezes sobre os mesmo dados até que os parâmetros de classificação não apresentação variação significativa.

Vejamos a aplicação do EM ao Naive Bayes.

Deve-se primeiro estimar os parâmetros geradores da distribuição $\hat{\theta}$ utilizando Naive Bayes sobre os dados completos. Depois podemos utilizar este preditor inicial para determinar as probabilidades das amostras incompletas pertencerem a cada classe. Diante desta classificação prévia das amostras incompletas, podemos refazer o classificador utilizando agora tanto os dados completos originais como esses incompletos que tiveram atribuídos probabilidades de classe. Prossegue-se utilizando este novo classificador para novamente atribuir probabilidades às amostras

incompletas, e esse ciclo continua até que $\hat{\theta}$ pare de variar. A convergência destas iterações foi demonstrada por Dempster.

O uso do EM com o Naive Bayes é descrito em partes a seguir:

1. Devem ser obtidos dados Labeled $\mathcal{D}^l$ e Unlabeled $\mathcal{D}^u$

2. Construir um classificador Naive Bayes inicial, $\hat{\theta}$, a partir dos dados *labeled*. Usar a estimativa máximo *a posteriori* para o cálculo dos parâmetros $\hat{\theta} = \arg\max_\theta \mathrm{P}(\mathcal{D}|\theta)\mathrm{P}(\theta)$. Como exemplo, para o caso de classificação de documentos, as equações a seguir devem ser usadas para tal calculo.

$$\hat{\theta}_{w_t|c_j} \equiv \mathrm{P}(w_t|c_j;\hat{\theta}) = \frac{1 + \sum_{i=1}^{|\mathcal{D}|} N(w_t, d_i)\mathrm{P}(y_i = c_j|d_i)}{|V| + \sum_{s=1}^{|V|}\sum_{i=1}^{|\mathcal{D}|} N(w_s, d_i)\mathrm{P}(y_i = c_j|d_i)}$$

$$\hat{\theta}_{c_j} \equiv \mathrm{P}(c_j|\hat{\theta}) = \frac{1 + \sum_{i=1}^{|\mathcal{D}|} \mathrm{P}(y_i = c_j|d_i)}{|C| + |\mathcal{D}|}$$

3. Iterar enquanto houver melhora no classificador. Mede-se a melhora no classificador por $l_c(\theta|\mathcal{D};\mathbf{z})$ (a log-probabilidade completa dos dados classificados e não-classificados e a *à priori*):

$$l_c(\theta|\mathcal{D};\mathbf{z}) = \log(\mathrm{P}(\theta)) + \sum_{d_i \in \mathcal{D}}\sum_{j=1}^{|\mathbf{c}|} z_{ij}\log\left(\mathrm{P}(c_j|\theta)\mathrm{P}(d_i|c_j;\theta)\right)$$

3.1 E-Step

Usar o classificador atual $\hat{\theta}$ para estimar a probabilidade de cada uma das amostras pertencer à classe em questão $\mathrm{P}(c_j|d_i;\hat{\theta})$.

$$\mathrm{P}(y_i = c_j|d_i;\hat{\theta}) = \frac{\mathrm{P}(c_j|\hat{\theta})\prod_{k=1}^{|d_i|}\mathrm{P}(w_{d_i,k}|c_j;\hat{\theta})}{\sum_{r=1}^{|\mathcal{C}|}\mathrm{P}(c_r|\hat{\theta})\prod_{k=1}^{|d_i|}\mathrm{P}(w_{d_i,k}|c_r;\theta)}$$

3.2 M–Step

Reestimar o classificador ($\hat{\theta}$) utilizando as probabilidades calculadas para as amostras incompletas. Usar a estimativa maximum a posteriori para achar o novo $\hat{\theta} = \arg\max_\theta \mathrm{P}(\mathcal{D}|\theta)\mathrm{P}(\theta)$.

Voltar para o passo 3.1.

Existem técnicas para melhorar o desempenho do algoritmo EM, sendo que dois exemplos são : usar peso menor que 1 para os dados *unlabeled* e considerar múltiplos componentes de mistura para cada classe. Tais métodos demonstram-se eficazes em casos onde a magnitude da quantidade de registros *labeled* é diversas vezes menor que a de registros *unlabeled*.

Em resumo, EM encontra um $\hat{\theta}$ que maximiza localmente a verossimilhança dos seus parâmetros dados todos os registros – tanto os completos quanto os incompletos. Ele fornece um método através do qual dados não classificados podem contribuir para a estimação de parâmetros.

5 Co-Training

O algoritmo Co-Training explicitamente usa uma divisão de características quando aprendendo de dados rotulados e não-rotulados. Sua abordagem é incrementalmente construir dois classificadores sobre cada um dos grupos de características.

Cada classificador é inicializado usando apenas os poucos exemplos rotulados a disposição. Em cada rodada do Co-Training, cada classificador escolhe um registro não-rotulado por classe para adicionar ao conjunto de dados completos. Os registros selecionados são aqueles com maior confiança na classificação dada pelo classificador base. Então, cada classificador reconstroi-se do novo conjunto rotulado aumentado e o processo repete-se. Neste trabalho, utilizamos Naive Bayes como algoritmo de construção do classificador base. As probabilidades de classe são as estimativas de confiança usadas por Co-Training. Na hora da classificação, as previsões dos dois classificadores são combinadas multiplicando-se suas probabilidades a posteriori e renormalizando-as. A seqüência a seguir esquematiza tal processo:

Entradas : Uma coleção inicial de exemplos rotulados e uma de não-rotulados.

Processo: Itera-se enquanto existirem dados não rotulados

1. Construa classificador A usando a porção A de cada exemplo
2. Construa classificador B usando a porção B de cada exemplo
3. Para cada classe C, escolha o exemplo não-rotulado para o qual o classificador A está mais confiante em relação à classe C, retire-o da coleção de não-rotulados e junte-o à coleção de exemplos rotulados
4. Para cada classe C, escolha o exemplo não-rotulado para o qual o classificador B está mais confiante em relação à classe C, retire-o da coleção de não-rotulados e junte-o à coleção de exemplos rotulados

Saídas: Dois classificadores, A e B, que predizem rótulos de classe para novos exemplos. Estas predições são combinadas multiplicando-as e renormalizando os valores de probabilidade de classes.

A idéia é que, se a base de dados satisfizer duas condições a serem apresentadas a seguir, esse algoritmo produzirá resultados ótimos, possivelmente melhores que Naive Bayes com grande quantidade de rotulados à disposição.

A seleção de uma base de dados consistente e que atenda aos requisitos do Co-Training influi diretamente no desempenho do algoritmo. As bases de dados devem apresentar uma divisão natural de suas características, ou seja, os classificadores treinados por cada um destes grupos de características devem ser igualmente capazes de classificar novos registros. Desta forma, o registro que está sendo rotulado receberá a mesma previsão de classe, independente do classificador utilizado.

Deve-se observar que existe um segundo pré-requisito para a escolha da base, relacionado com os conjuntos de dados que gerarão os dois classificadores do algoritmo. As características dos dois conjuntos devem ser independentes condicionadas à classe, indicando deste modo a inexistência de uma relação entre os dois grupos de informação.

Enfim, a essência deste algoritmo está em que duas visões diferentes de um mesmo exemplo podem ser usadas para produzir classificação, e este método explora o fato de que um classificador construído por uma visão aprende com o outro de forma incremental para produzir no final dois classificadores mais precisos do que se tivessem sido gerados independentemente.

6 Macro Processo para classificação de dados com o algoritmo Cotraining

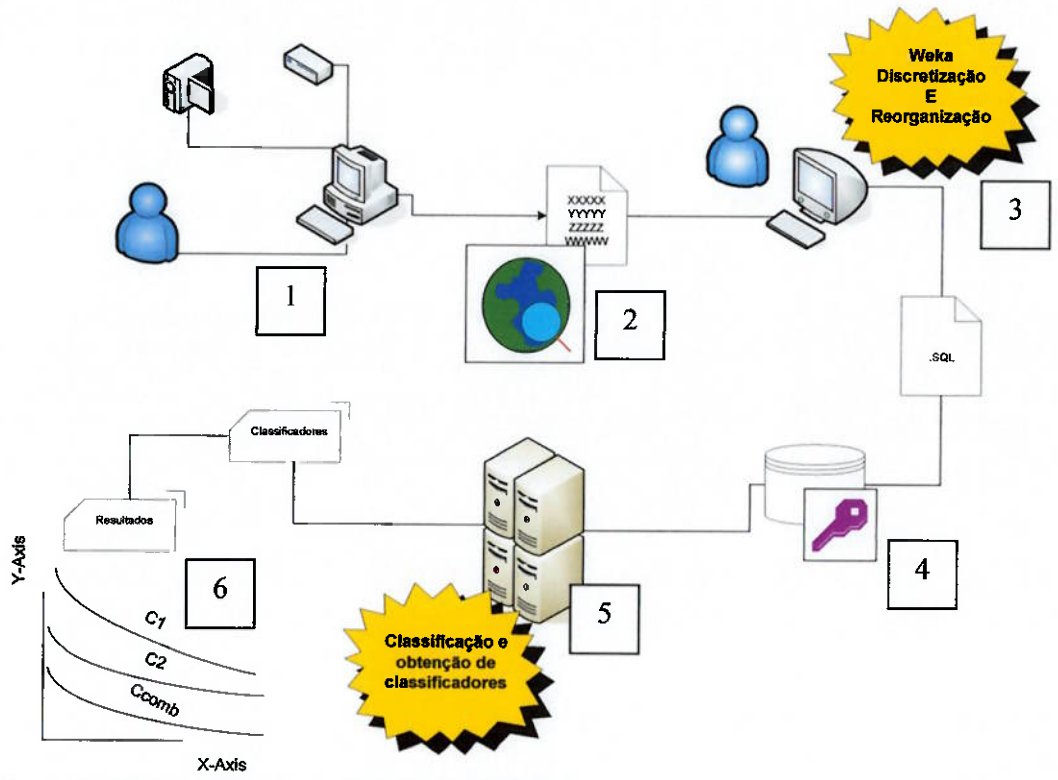


Figura 1- Macro processo de geração de classificadores

A obtenção de bases de dados que se adequem às limitações dos classificadores de dados atuais apresenta-se como um dos principais empecilhos para a utilização mais ampla de classificadores para o algoritmo implementado neste projeto. Oriundas de sensores, exames clínicos, dados da população entre outras fontes, as informações dificilmente encontram-se nos padrões aceitos pelo sistema desenvolvido. Primeiramente, explica-se a seguir o processo através do qual encontramos bases na Internet, para que depois sejam desenvolvidos os processos que terão como output os classificadores e seus resultados.

As fontes de dados utilizadas neste projeto de conclusão de curso foram obtidas na internet, dado que esta fora do escopo do mesmo a geração destes inputs. Dentre os websites que fornecem bases de dados

doadas por terceiros, utilizou-se neste vastamente o website http://www.ics.uci.edu/~mlearn/MLRepository.html. Tal site é mantido pela University of California, Irvine, Dept. of Information and Computer Sciences.

Encontra-se na Internet arquivos que apresentam como problema chave a continuidade dos parâmetros de classificação e uma quantidade de classes maior que duas. Deste modo torna-se necessário um tratamento prévio das bases de dados encontradas. A discretização e agrupamento de classes foram os tratamentos aplicados a estas fontes, além da reorganização do formato.

39,State-gov,77516,Bachelors,13,Never-married,Adm-clerical,Not-in-family,White,Male,2174,0,40,United-States,<=50K
50,Self-emp-not-inc,83311,Bachelors,13,Married-civ-spouse,Exec-managerial,Husband,White,Male,0,0,13,United-States,<=50K
38,Private,215646,HS-grad,9,Divorced,Handlers-cleaners,Not-in-family,White,Male,0,0,40,United-States,<=50K
53,Private,234721,11th,7,Married-civ-spouse,Handlers-cleaners,Husband,Black,Male,0,0,40,United-States,<=50K
28,Private,338409,Bachelors,13,Married-civ-spouse,Prof-specialty,Wife,Black,Female,0,0,40,Cuba,<=50K
37,Private,284582,Masters,14,Married-civ-spouse,Exec-managerial,Wife,White,Female,0,0,40,United-States,<=50K
49,Private,160187,9th,5,Married-spouse-absent,Other-service,Not-in-family,Black,Female,0,0,16,Jamaica,<=50K
52,Self-emp-not-inc,209642,HS-grad,9,Married-civ-spouse,Exec-managerial,Husband,White,Male,0,0,45,United-States,>50K

Tabela 1 - Exemplo de base de dados crua

Explica-se no capitulo 8 o processo de discretização e agrupamento de classe feito no software Weka.

Após o pré-tratamento das bases de dados, a importação para o banco de dados é necessária. A implementação de uma interface no formato de banco de dados foi feita de forma a facilitar o futuro input de dados, haja vista que atualmente esta é a forma mais utilizada para armazenar informação. Detalhar-se-á o tratamento feito nesta interface com o banco de dados também neste capitulo.

Por fim, a geração dos classificadores pode ser iniciada com base no banco de dados carregado. A explicação sobre a geração dos classificadores e o resultado dos testes para as diversas combinações de dados rotulados e não rotulados será fornecida no próximo capitulo.

6.1 Java – Preparação de bases

Devido à maneira através da qual implementou-se o algoritmo Co-Training, a preparação das entradas para o método demonstrou-se deverás trabalhosa. Os arquivos texto utilizados para definir instancias Labeled,

Unlabeled e Testes, eram previamente preparados de forma manual. Desta forma a realização dos testes demandava tempo, tornando-se também mais vulnerável a erros na preparação. Para a correta validação do algoritmo Co-Training como gerador de classificadores é necessária a realização de combinações de proporções entre instancias Labeled e Unlabeled.

Tendo em vista tais dificuldades para a realização de testes, será implementada uma classe cujo objetivo será a preparação dos arquivos de entrada e a automatização da variação de Labeled/Unlabeled.

Para que seja viabilizada esta implementação, a utilização de meios mais eficazes para manipulação de dados demonstra-se necessária. Assim sendo, utilizar-se-á o banco de dados Microsoft Access juntamente com componentes em Java para lidar com a preparação das entradas do algoritmo.

6.2 Java – Co-training

A implementação do algoritmo Co-Training deu-se através de codificação em linguagem Java. Aproveitamos, para isso, trechos do código do software EMBayes, escrito pelo Prof. Dr. Fábio Cozman (EPUSP), que contém métodos que manipulam variáveis (atributos) das bases de dados e redes bayesianas.

Este código foi adaptado para a utilização conforme o algoritmo Co-Training, ou seja, ele passou a aplicar Naive Bayes para duas bases distintas, que apesar de representarem os mesmos registros, continham atributos diferentes. Por conseqüência, o código passou a lidar com dois classificadores (duas redes bayesianas com parâmetros e variáveis distintos), ao invés de um só.

Foram implementadas funções de manipulação de arquivos, para adicionar registros à base de classificados e retirar da base de não-classificados.

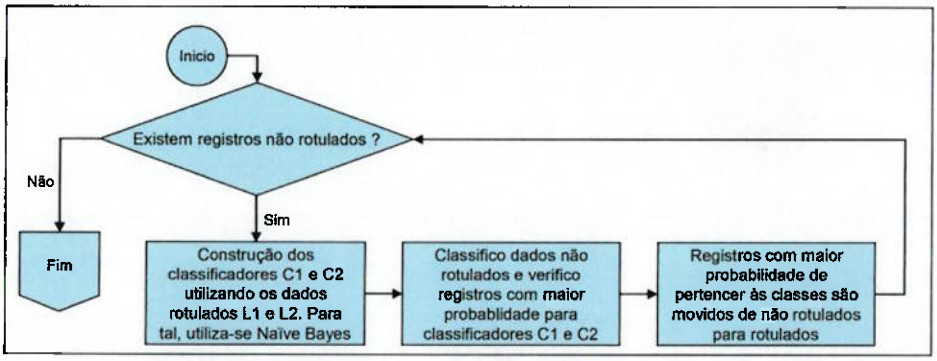
Foi preciso aumentar o número de parâmetros de entrada tratá-los de acordo. Alguns novos parâmetros são : especificação dos 4 arquivos com

dados que serão manipulados, 2 arquivos com dados para teste, 2 arquivos onde seriam gravadas as saídas do programa.

As limitações desta implementação são:

- As classes possíveis dos registros devem ser no máximos DUAS; Seria possível adaptar o código para mais de duas classes, porém o impacto seria intenso requerendo diversas alterações no código. No entanto, tal implementação na alteraria a análise e entendimento do método de geração dos classificadores
- Os atributos das base podem conter apenas valores discretos: isso se deve ao fato de que as probabilidades são calculadas baseadas na contagem das ocorrências, e isso seria impossível com um atributo de valores contínuos.
- Os valores possíveis não podem ultrapassar 5 (valor variável depende do número de registros): quanto menos registros existirem na base, menor é o número máximo de valores que os atributos podem assumir, pois se os registros escolhidos para serem utilizados como treinamento poderiam ser muito singulares, e o classificador não teria uma boa estimativa da distribuição de probabilidades real.

A figura a seguir ilustra o fluxograma simplificado do funcionamento do Co-Training.



A codificação da classe principal deste programa, que executa o algoritmo Co-Training, está listada na tabela 5.1, anexo A.

7 Estrutura desenvolvida em Java

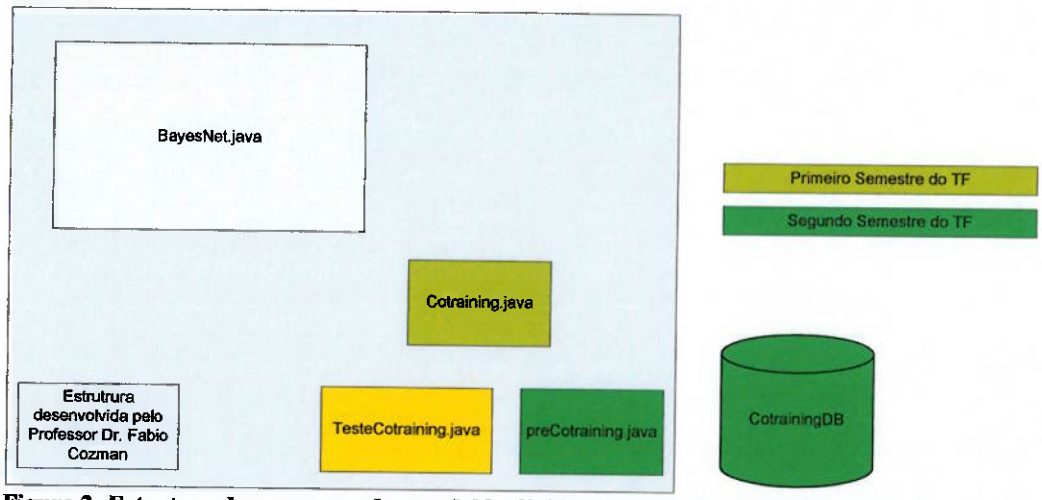


Figura 2- Estrutura do programa desenvolvido divido por semestres do Trabalho de Conclusão de curso

7.1 Descrição da camada Java

A implementação do algoritmo Co-training e da estrutura para realização de testes de performance foi dividida em dois semestres.

O desenvolvimento das classes Cotraining.java e da classe TesteCotraining.java foi feito no primeiro semestre, de modo que sua extensão (classe preCotraining.java) foi implementada no segundo semestre.

As funcionalidades de cada classe serão explicadas nos próximos subcapítulos.

7.1.1 CoTraining.java

A classe Cotraining.java é a principal classe desenvolvida, sendo esta a classe que contém o main.

Nesta classe define-se a quantidade de testes que será feita para a base contida no banco de dados, assim como suas proporções de dados Rotulados e não Rotulados.

Após a definição da quantidade de testes e do percentual Labeled, esta classe chamará a preCotraining.java para que seja iniciada a preparação dos arquivos utilizados como entrada para o inicio da classificação. Explicar-se-á o funcionamento desta preparação de arquivos a seguir.

O processo de geração dos classificadores é iniciado após o termino da preparação dos arquivos que serão utilizados para este processo. Conforme prega o algoritmo co-training, é iniciada a geração de dois classificadores (C1 e C2) que visualizam cada uma das duas divisões dos registros da base de dados. À medida que os classificadores são gerados, é feita uma classificação dos dados não rotulados, que será utilizada para a escolha dos registros que serão movidos para o arquivo de dados rotulados. Este processo ocorrerá até que não existam mais dados não rotulados, desta forma serão gerados classificadores que tendem a apresentar uma melhor performance. Ao fim da classificação de todos os dados rotulados, será chamado um método que gerará um arquivo de saída com o desempenho de cada um dos classificadores e do classificador combinado para os dados separados previamente para os testes.

7.1.2 TesteCotraining.java

A classe TesteCotraining.java foi criado com o objetivo de achar o registro que possui a maior probabilidade de pertencer às classes previamente definidas. Tal função será utilizada para a reconstrução dos arquivos Labeled e Unlabeled, de forma a retirar os registros unlabeled que possuem a maior probabilidade de pertencer às classes mencionadas e enviá-los para o arquivo Labeled.

Desta forma a função será chamada até que o arquivo que contém os registros Unlabeled não possua nenhum arquivo, de maneira que se possa gerar o ultimo classificador. Espera-se que o ultimo classificador gerado, utilizando os dados previamente denominados unlabeled como labeled apresenta uma menor incerteza na classificação.

7.1.3 PreCotraining.java

A classe PreCotraining.java foi desenvolvida diante da necessidade de facilitar a geração dos arquivos de entrada para a geração e testes do Cotraining. Desta maneira, ela será responsável por todo o processo de geração dos arquivos Labeled, Unlabeled e Teste.

Primeiramente, a classe entende como uma premissa a existência de dados na tabela COTRA_MAIN. Dado este fato, um método responsável irá carregar uma tabela auxiliar com todos os registros contidos na COTRA_MAIN.

Após a carga da tabela auxiliar, o método irá separar 20% dos registros para teste. Separa-se estes registros que futuramente serão utilizados para testes de forma a selecionar 50% da classe 1 e 50% da classe 2. A tabela COTRA_TEST será carregada com estes registros. Posteriormente será gerado um arquivo texto contendo os registros aqui aleatoriamente selecionados, a ser utilizados pelas classes descritas anteriormente.

Após a separação dos registros para teste, são separados os registros para a geração do classificador. A proporção de registros Labeled e registros Unlabeled será feita de acordo com o parâmetro de entrada, que poderá variar conforme é feito pela classe Cotraining, aumentando desta maneira a velocidade gasta para a geração de arquivos com diversas combinações de dados Labeled e Unlabeled.

Ao final do processo os seguintes arquivos são gerados:

Labeled1.in

Labeled2.in

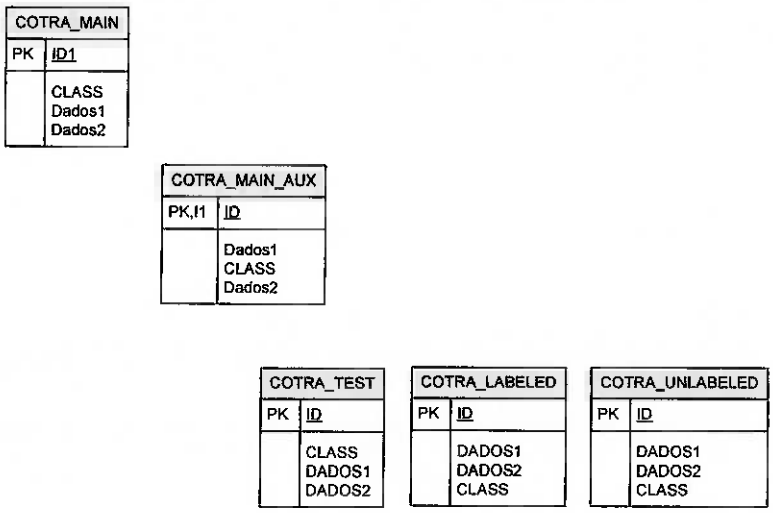
Unlabeled1.in

Unlabeled2.in

TesteUnlabeled1.in

TesteUnlabeled2.in

7.2 Modelo de dados da camada em Microsoft Access



8 Weka, Discretizar e arrumar bases de dados

A vasta quantidade de dados disponíveis na internet apresenta uma limitação em relação ao seu uso no sistema de classificação utilizado pelo algoritmo do Co-Training implementado neste trabalho. Haja vista que o cunho deste projeto é realizar testes com bases que se aproximem de forma coerente da realidade, é de trivial importância a utilização de bases reais. Tais bases são geradas a partir de resultados de exames médicos, analises da população, informações de sensores, dentre outros geradores. A conseqüência imediata deste fato é a continuidade das informações, causando um conflito com o algoritmo co-training que apresenta somente um tratamento para bases de dados com discretização.

Neste projeto será utilizado um software desenvolvido pela University of Waikato (Nova Zelândia), que possui diversas funcionalidades para o tratamento de bases de dados. Dentre as diversos tratamentos de dados disponíveis, será usada a implementação do método Fayyad & Irani's MDL para a realização da discretização de componentes contínuos na bases de dados.

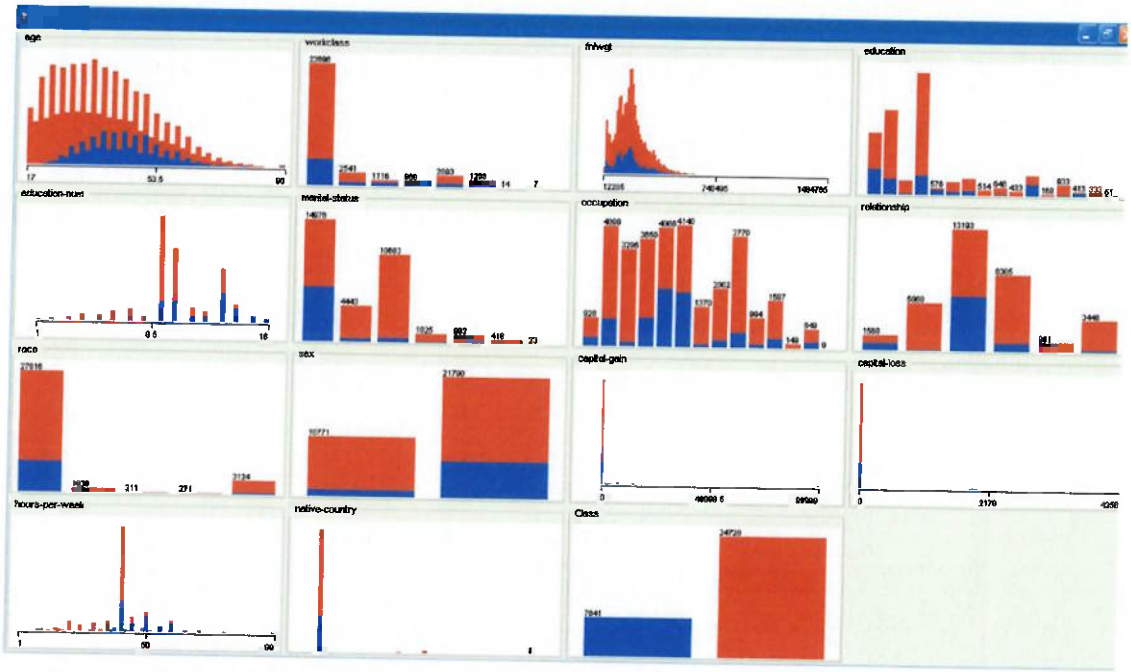


Figura 3- Dados contínuos no Weka

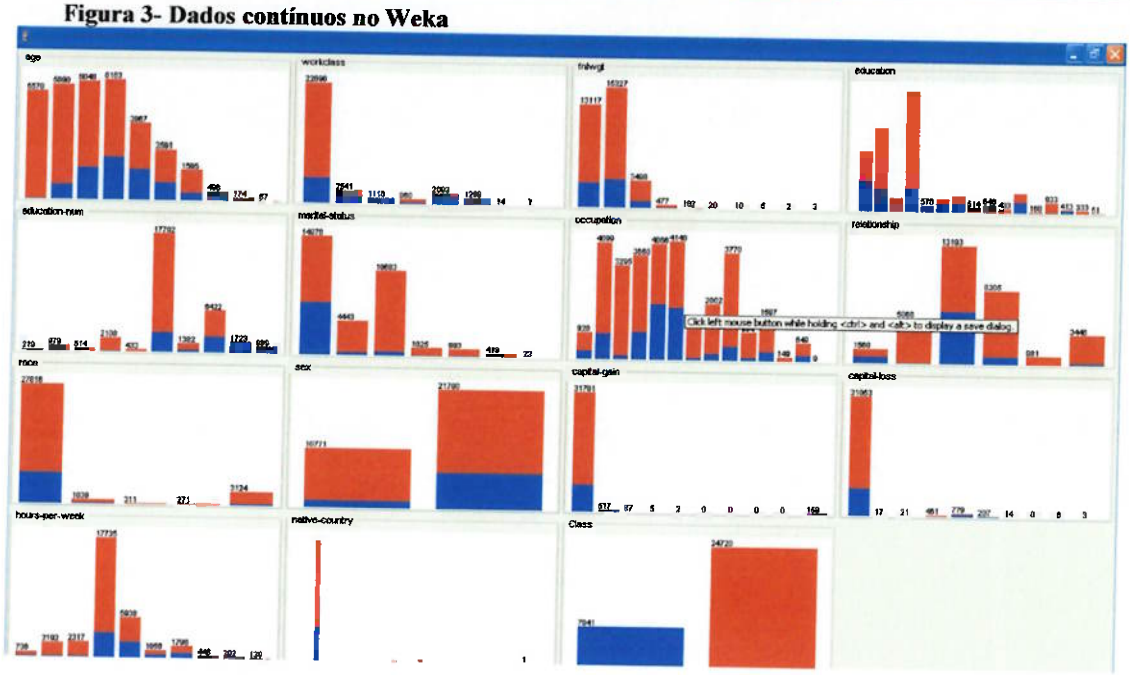
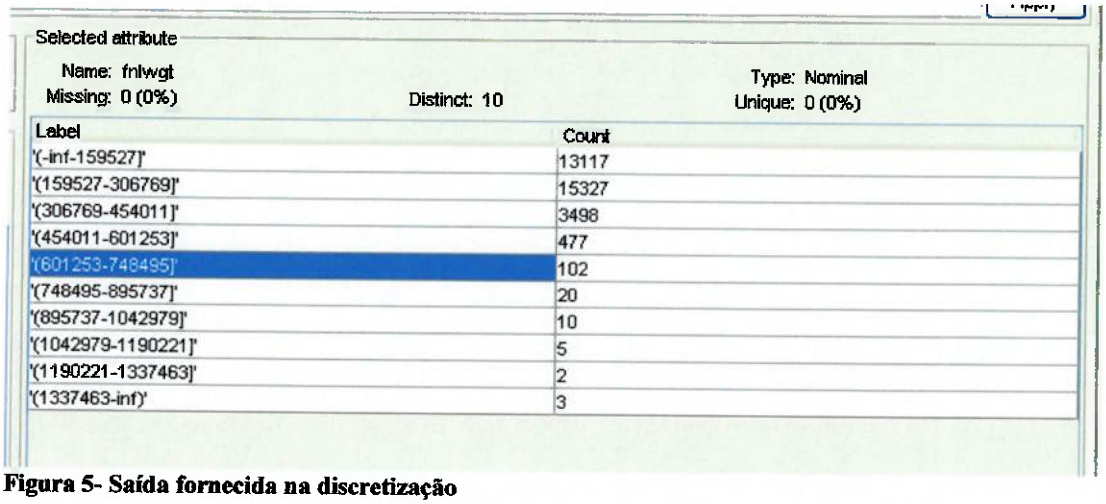


Figura 4- Dados discretizados no Weka

Há, no entanto, um problema em relação à discretização de dados pelo Weka devido às particularidades da implementação em Java. Espera-se que as bases fornecidas para a criação dos classificadores apresentem-se na forma de inteiros, ou seja, diferente da forma de saída do Weka. Desta

maneira, é necessário tratar dos arquivos texto de saída das bases de dados de forma a substituir os intervalos por valores inteiros distintos.



Selected attribute

Name: fnlwgt
Missing: 0 (0%)
Distinct: 10
Type: Nominal
Unique: 0 (0%)

Label	Count
'(-inf-159527]'	13117
'(159527-306769]'	15327
'(306769-454011]'	3498
'(454011-601253]'	477
'(601253-748495]'	102
'(748495-895737]'	20
'(895737-1042979]'	10
'(1042979-1190221]'	5
'(1190221-1337463]'	2
'(1337463-inf)'	3

Figura 5- Saída fornecida na discretização

Por fim, para que a base demonstre-se em perfeitas para a entrada no banco de dados, deve-se observar que o numero de classes não pode ser maior que duas. Dentre as bases utilizados nos testes durante este trabalho, diversas apresentavam mais que a quantidade permitida de classes. A maneira mais eficaz para solucionar este empecilho é o agrupamento de classes, de forma a obter somente duas classes resultantes. O agrupamento foi feito de forma lógica, observando que a quantidade de registros para as classes e sua compatibilidade lógica.

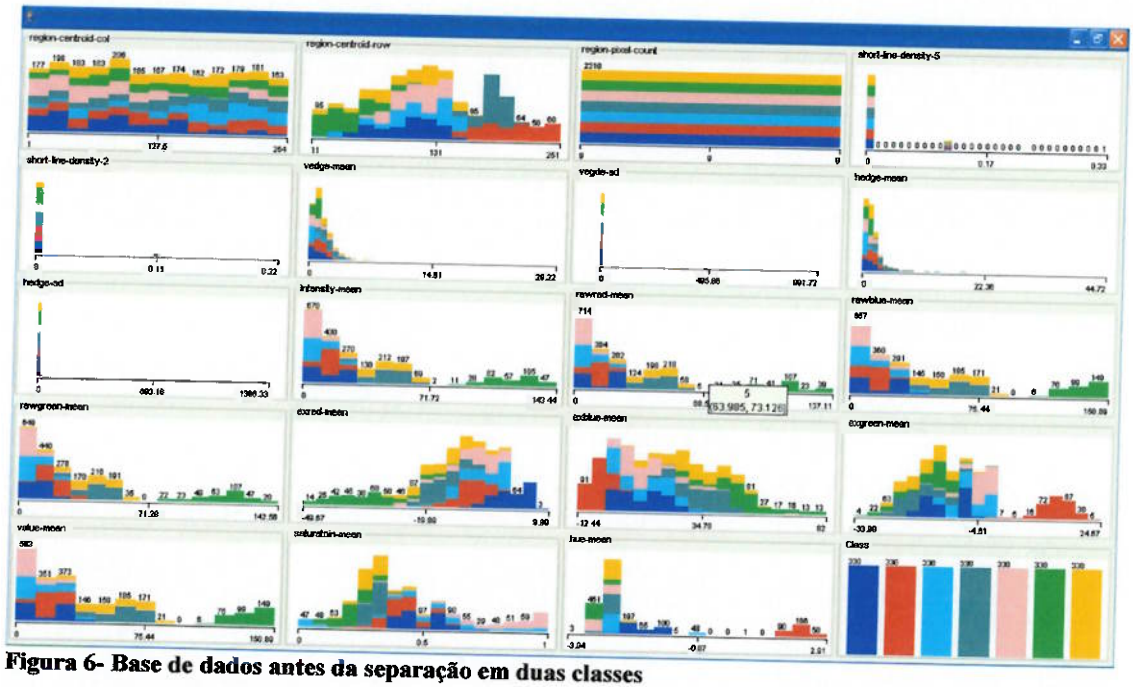


Figura 6- Base de dados antes da separação em duas classes

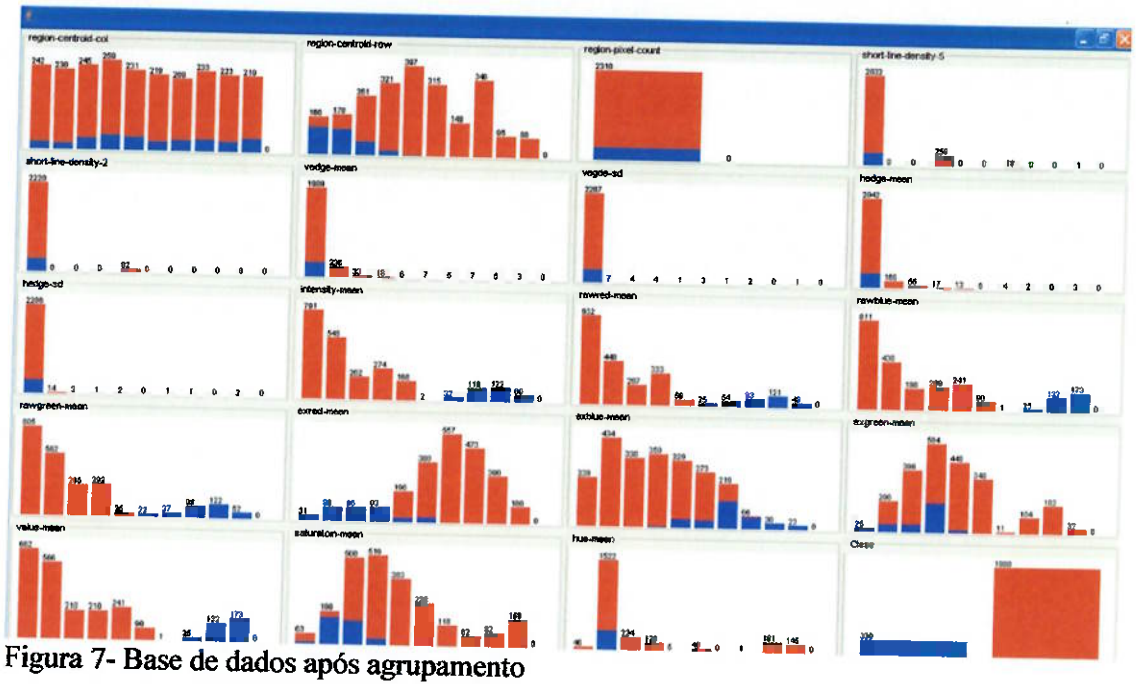


Figura 7- Base de dados após agrupamento

9 Descrição das bases de dados para testes

Dado que a eficácia do algoritmo gerador de classificadores só pode ser mensurada através da realização de testes, submeter-se-á o sistema desenvolvido à uma bateria de testes sobre diversas bases de dados. Entende-se que uma base de dados deve atender às seguintes premissas para que apresente uma classificação condizente através do Co-Training:

1ª premissa: *Cada conjunto de características é suficiente para classificação.* Se dividirmos essa base em duas, com oito atributos para cada uma, além da classe, é uma suposição razoável afirmar que cada base sozinha é suficiente, se pensarmos que deputados de um mesmo partido tenderão a votar de um mesmo modo, independentemente de que projeto está sendo votado.

2ª premissa: *Os dois conjuntos de características de cada instância são condicionalmente independentes entre si, dada a classe.* Novamente, é razoável supormos que, dado o partido ao qual o deputado faz parte, o seu voto para o projeto X independe do voto para o projeto Y, pois ele tenderá a votar de acordo com a recomendação do seu partido.

Entretanto, utilizar-se-á bases que atendem às premissas e também bases que não atendem, visando a compreensão do desempenho nos ambientes mais variados possíveis.

Devido à implementação feita para a automatização dos testes, o processo será tal que o tempo despendido para esta tarefa será razoável. Para cada uma das bases serão realizados testes, onde será variada a relação de instancias Labeled/Unlabeled. Esta variação de dados poderá nos dar uma visão mais clara da quantidade de dados necessários para que o algoritmo obtenha um resultado satisfatório.

As bases a serem utilizadas estão disponíveis na Internet em vasta quantidade, podendo ser usadas livremente para os testes. Encontram-se no site da University of California (Irvine) diversas bases utilizadas por outros

pesquisadores para o aprendizado de classificadores (http://www.ics.uci.edu/~mlearn/MLRepository.html). No entanto, há uma necessidade de adaptação de algumas destas bases, para que desta forma atendam às limitações de implementação do algoritmo. Neste momento é válido lembrar as três limitações triviais do algoritmo implementado :

- Bases devem conter apenas DUAS classes
- Atributos devem ser discretos
- Atributos discretos devem ser limitados à uma quantidade determinada

Tendo em vista esta limitações e observando as bases disponíveis na Internet, conclui-se que será necessário realizar algumas adaptações nestas. Tais adaptações visam adequar as bases às limitações apresentadas previamente. Algumas bases deverão ser discretizadas de forma racional, para que informações não sejam perdidas. Outras deverão ter seu número de classes reduzido para que possam ser classificadas pelo algoritmo desenvolvido.

Segue uma breve descrição das bases que serão inicialmente utilizadas no testes.

9.1 Congressional Voting Records

A primeira base escolhida para teste é a chamada "1984 United States Congressional Voting Records". Esta base inclui os votos de cada um dos congressistas americanos de 1984 nos 16 votos-chave identificados pelo Congressional Quarterly Almanac. O CQA lista nove tipos diferentes de votos: voted for, paired for, e announced for (estes simplificados para Y para os testes), voted against, paired against, e announced against (estes três simplificados para N), voted present, voted present to avoid conflict of interest, e did not vote or otherwise make a position known (estes três simplificados para uma posição desconhecida).

Há 435 registros nessa base, sendo 267 da classe Democrata e 168 da classe Republicano. É composta pelos seguintes atributos (exceto pelo partido, todos são projetos de lei, emendas constitucionais, ou decisões de caráter social ou religioso votados na U.S. House of Representatives no ano de 1984):

1. Nome do partido: 2 (democrat, republican)
2. Handicapped-infants: 2 (y,n)
3. Water-project-cost-sharing: 2 (y,n)
4. Adoption-of-the-budget-resolution: 2 (y,n)
5. Physician-fee-freeze: 2 (y,n)
6. El-salvador-aid: 2 (y,n)
7. Religious-groups-in-schools: 2 (y,n)
8. Anti-satellite-test-ban: 2 (y,n)
9. Aid-to-nicaraguan-contras: 2 (y,n)
10. Mx-missile: 2 (y,n)
11. Immigration: 2 (y,n)
12. Synfuels-corporation-cutback: 2 (y,n)
13. Education-spending: 2 (y,n)
14. Superfund-right-to-sue: 2 (y,n)
15. Crime: 2 (y,n)
16. Duty-free-exports: 2 (y,n)
17. Export-administration-act-south-africa: 2 (y,n)

É claro que a classificação neste caso se dá pelo partido a que cada deputado pertence.

É claro que pode-se pensar que questões como grupos religiosos nas escolas, ajuda a crianças inválidas, imigrantes e outras podem depender muito mais da opinião pessoal do deputado do que propriamente do que o seu partido recomenda. Neste caso, a base não mais satisfaz aos critérios procurados, mas em última análise, ainda é uma base válida pois pelo menos no plano filosófico aproxima-se das condições ideais.

9.2 Breast Cancer Wisconsin

A base é composta de 699 registros de amostras de células cancerosas, tal que os atributos são características destas células. As informações encontram-se discretizadas, variando dentre 5 possíveis

valores. Os atributos variavam originalmente entre 1 e 10, porém devido às implementações do algoritmo, agrupou-se os atributos dois a dois.

Segue uma descrição dos atributos encontrados nesta base:

1. Tipo de tumor: 2 (maligno ou benigno)
2. Densidade da amostra: 5,4,3,2,1
3. Uniformidade no tamanho das células: 5,4,3,2,1
4. Uniformidade na forma das células: 5,4,3,2,1
5. Aderência marginal: 5,4,3,2,1
6. Tamanho da célula epitelial simples: 5,4,3,2,1
7. Núcleos expostos: 5,4,3,2,1
8. Cromatina branda: 5,4,3,2,1
9. Nucléolos normais: 5,4,3,2,1
10. Mitoses: 5,4,3,2,1

9.3 Análise de imagens

A base representa a composição de pixeis de imagens feitas pelo Vision Group da University of Massachusetts. As instancias são compostas de diversos atributos dos pixeis que compõe a imagem e servem para realizar a classificação de acordo com o tipo de região da qual este faz parte. Dentre as possíveis opções de classificação estão:

- Tijolo
- Céu
- Folhagem
- Cimento
- Janela
- Caminho
- Grama

Os valores dos atributos desta base são disponibilizados de maneira continua, sendo deste modo necessária a realização de um tratamento para a realização da discretização dos dados. Os seguintes atributos compõe a base:

1.region-centroid-col: Coluna do pixel central da região

2.region-centroid-row: Linha do pixel central da região

3.region-pixel-count: Número de pixeis na região = 9

4.short-line-density-5: Resultados de um algoritmo para extração de uma linha que conta quantas linhas de tamanho 5 (qualquer orientação) com pouco contraste, com cinco ou menos, passam pela região

5.short-line-density-2: Mesmo que short-line-density-5, porém conta as linhas de alto contraste

6.vedge-mean: Mede o contraste de pixeis horizontais adjacentes numa região. Existem 6, a média e o desvio padrão são dados. Este atributo é usado como um detector de beira.

7.vegde-sd: Desvio padrão do atributo citado em 6

8.hedge-mean: Mede o contraste vertical de pixeis adjacentes. Usado para detecção de linhas horizontais.

9.hedge-sd: Desvio padrão do dado citado em 8

10.intensity-mean: Média sobre a região de (R + G+ B)/3

11.rawred-mean: Média de Vermelho na região.

12.rawblue-mean: Média de Azul na região

13.rawgreen-mean: Média de Verde na região

14.exred-mean: (2R - (G + B))

15.exblue-mean: (2B - (G + R))

16.exgreen-mean: (2G - (R + B))

17.value-mean: Transformação 3D não linear de RGB

18.saturatoin-mean: Resultado da transformação feita em 17

19.hue-mean: Resultado da transformação feita em 17

Devido às limitações de implementação do algoritmo, será necessário realizar o agrupamento de algumas classes, para que a classificação seja feita em relação a somente dois possíveis valores. Assim sendo, através de uma analise da quantidade de registros por classe e também de um possível agrupamento lógico dos mesmos, optou-se por agrupar os dados em duas classes: céu e outros.

9.4 Renda da população

A base foi extraída do Censo Americano, contendo dados gerais sobre a população. O objetivo desta classificação será distinguir pessoas que possuem renda maior que 50k e pessoas com renda inferior a 50k, sendo estas as duas classes a serem trabalhadas. A base é composta de 45222 instancias, e seus atributos são:

- Idade
- Classe de trabalho
- Peso da amostra final

- Educação
- Anos de estudo
- Estado civil
- Ocupação
- Relacionamento
- Raça
- Sexo
- Ganho de capital
- Perda de capital
- Horas de trabalho semanais
- Nacionalidade

9.5 *Leitura de sonares*

A base é composta de leituras de sonares cujos sinais refletiram em um cilindro de metal ou em um cilindro de rocha. As instancias são compostas de 60 números entre 0.0 e 1.0, de modo que tais valores representam a energia numa freqüência particular integrada ao longo do tempo. Os valores foram obtidos através da variação dos ângulos de recepção.

9.6 *Imagens de satélite*

Nesta base encontram-se registros de valores multi-espectrais de pixeis numa região 3x3 de uma foto de satélite, e a classificação associada ao pixel central. Dadas estas características dos pixeis, o objetivo é prever o tipo de solo que se encontra na região equivalente àquele pixel central. As possíveis classes são:

- Solo vermelho
- Plantação de algodão
- Solo cinzento
- Solo cinzento úmido
- Solo com vegetação
- Mistura de solos
- Solo cinzento muito úmido

9.7 Doenças Epiteliais

Esta base de dados contém 34 atributos, dos quais 33 são lineares e um deles é nominal.

Os diferentes diagnósticos de doenças erythemato-squamous é um grande problemas encontrado na dermatologia. Todas estas doenças compartilham sintomas e tem poucas diferenças entre si. Normalmente uma biopsia é necessária para diagnosticar estas doenças, porém infelizmente muitas vezes os resultados destes testes são muito semelhantes. Outro ponto de dificuldade para este tipo de diagnostico é o fato de algumas destas doenças apresentarem os sintomas das outras nos estágios iniciais e podem continuar apresentando este tipo de problema nos próximos estágios.

- ✓ Psoriasis 112 registros
- ✓ seboreic dermatitis 61 registros
- ✓ lichen planus 72 registros
- ✓ pityriasis rosea 49 registros
- ✓ cronic dermatitis 52 registros
- ✓ pityriasis rubra pilaris 20 registros

Nesta base os pacientes foram primeiramente avaliados clinicamente com 12 features, e depois realizou-se a biopsia com a verificação de 22 features histológicos.

Os dados desta base possuem as seguintes características, de acordo com sua categoria. Para os parâmetros que indicam histórico familiar, tem valor 1 se alguma destas doenças já foi observada na família e 0 no caso contrario. Na idade utilizou-se um parâmetro continuo. Para os outros features, temos um range de 0 a 3, de tal maneira que 0 indica a completa ausência, e 1-3 indica o nível de presença.

1. erythema: 0,1,2,3.
2. scaling: 0,1,2,3.
3. definite_borders: 0,1,2,3.
4. itching: 0,1,2,3.
5. koebner_phenomenon: 0,1,2,3.
6. polygonal_papules: 0,1,2,3.
7. follicular_papules: 0,1,2,3.
8. ral_mucosal_involvement: 0,1,2,3.
9. knee_and_elbow_involvement: 0,1,2,3.
10. scalp_involvement: 0,1,2,3.
11. family_history: 0,1.

12. melanin_incontinence: 0,1,2,3.
13. eosinophils_in_the_infiltrate: 0,1,2,3.
14. PNL_infiltrate: 0,1,2,3.
15. fibrosis_of_the_papillary_dermis: 0,1,2,3.
16. exocytosis: 0,1,2,3.
17. acanthosis: 0,1,2,3.
18. hyperkeratosis: 0,1,2,3.
19. parakeratosis: 0,1,2,3.
20. clubbing_of_the_rete_ridges: 0,1,2,3.
21. elongation_of_the_rete_ridges: 0,1,2,3.
22. thinning_of_the_suprapapillary_epidermis: 0,1,2,3.
23. spongiform_pustule: 0,1,2,3.
24. munro_microabcess: 0,1,2,3.
25. focal_hypergranulosis: 0,1,2,3.
26. disappearance_of_the_granular_layer: 0,1,2,3.
27. vacuolisation_and_damage_of_basal_layer: 0,1,2,3.
28. spongiosis: 0,1,2,3.
29. saw-tooth_appearance_of_retes: 0,1,2,3.
30. follicular_horn_plug: 0,1,2,3.
31. perifollicular_parakeratosis: 0,1,2,3.
32. inflammatory_monoluclear_inflitrate: 0,1,2,3.
33. band-like_infiltrate: 0,1,2,3.
34. Age:continuous.

9.8 Tabuleiro de Xadrez – kr vs kp a7

Nesta base de dados encontram-se os dados de um tabuleiro de xadrez, onde se tem como objetivo classificar o tabuleiro quanto ao fim do jogo. Temos neste jogo uma situação definida por um rei e um peão contra um rei e uma torre. A posição destas peças é tal que o peão está a uma posição para se tornar uma rainha e a vez é do lado rei e torre (branco).

A base possui 36 atributos representando cada um uma posição do tabuleiro de xadrez. A classificação será quanto à vitória do branco ou sua impossibilidade de ganhar.

9.9 Avaliação de Carros

Dentre as diversas aplicações praticas do uso de classificadores, pode-se destacar a possibilidade da utilização de classificadores como tomadores de decisão. Visando realizar um estudo sobre a efetividade do co-training como tomador de decisão, optou-se pelo uso da base na qual se trata da decisão de compra de um automóvel.

Originalmente o grupo de possíveis classes era formado por quatro componentes, porém para atender às limitações do algoritmo foi necessário a redução para duas classes. Dentre os diversos agrupamentos possíveis,

admitiu-se que o mais logicamente correto seria agrupar os registros em carros aceitáveis e carros não aceitáveis. Deste modo as qualificações ac, good e vgood enquadraram-se como aceitáveis e os registros unacc ficaram com a classe não aceitáveis.

Os parâmetros para esta base de dados são:

- ✓ buying: vhigh, high, med, low.
- ✓ maint: vhigh, high, med, low.
- ✓ doors: 2, 3, 4, 5more.
- ✓ persons: 2, 4, more.
- ✓ lug_boot: small, med, big.
- ✓ safety: low, med, high.

10 Testes e Resultados

Os testes devem ser realizados de forma a comprovar a eficácia da implementação do Co-Training desenvolvida ao longo deste trabalho. Desta forma submeter-se-á as diversas bases de dados apresentadas à uma bateria de testes, onde serão variados os parâmetros de entrada do algoritmo. Como parâmetros de entrada do algoritmo deve-se entender a relação entre o número de instancias Labeled e Unlabeled. Tal variação será feita pela classe implementada, responsável pela preparação das bases de dados. No entanto os percentuais de dados Labeled em relação aos Unlabeled deverá ser fornecido ao programa, que por sua vez gerará os resultados.

Para que se possa analisar o coeficiente de acerto do classificador gerado pelo Co-Training, serão gerados classificadores auxiliares que serão usados como padrões de comparação. Observa-se no entanto que as bases devem atender às premissas do algoritmo Co-Training, já apresentadas no inicio deste documento.

Analisar-se-á os resultados de acordo com as premissas, verificando-se desta maneira se estes condizem com o esperado. Visando uma melhor compreensão de cada uma das bases, será feita uma analise individual dos resultados obtidos.

Visando uma melhor compreensão dos resultados, estes serão apresentados na forma de uma tabela. Desta forma, será facilitada a comparação entre diversas bases.

A tabela de comparação é composta por oito campos, sendo:

- ✓ Classificador 1 converge
  - Analisa a convergência do classificador 1, ou seja, verifica-se se o erro tenderá a ser menor com o aumento da quantidade de arquivos Labeled
- ✓ Classificador 2 converge
  - Idem ao feito para o classificador 1
- ✓ Classificador Combinado converge
  - Idem ao feito para o classificador 1
- ✓ Classificador Combinado apresenta erro menor que classificadores C1 e C2
  - Verifica-se a precisão do classificador combinado, tendo em vista que teoricamente ele deverá apresentar um erro menor que cada um dos classificadores C1 e C2
- ✓ Desempenho do classificador combinado é melhor que EM para %Labeled < 5%
  - O desempenho do classificador combinado é comparado com o desempenho do EM para uma quantidade de registros Labeled < 5%
- ✓ Desempenho do classificador combinado é melhor que EM e NB
  - Compara-se o classificador final do co-training (combinado) com os demais classificadores de referencia
- ✓ Base atende à premissa da auto-suficiência
  - Analise da base de dados quanto à adequação ao principio de que cada uma das divisões da base deve ser suficiente por si só para a realização de uma classificação eficiente
- ✓ Base atende à premissa da independência
  - Analise da base de dados quanto à adequação ao principio de que cada uma das divisões da base é independente da outra.

10.1 Imagens de satélite

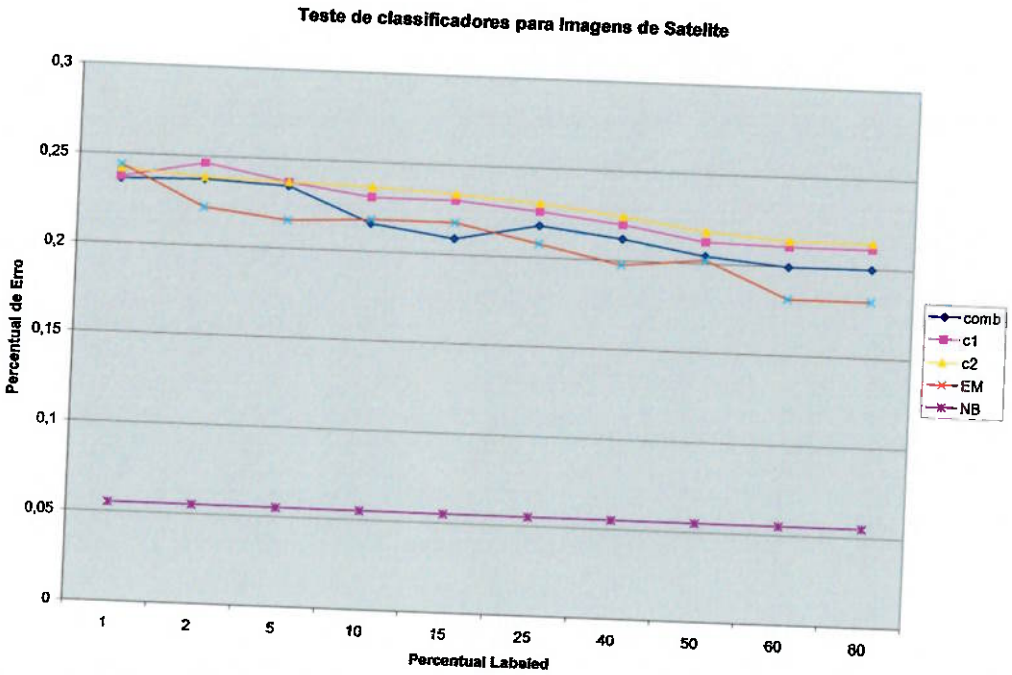


Figura 8- Teste de classificadores para Imagens de Satélite

Imagens de satélite	
Classificador 1 converge	Sim
Classificador 2 converge	Sim
Classificador Combinado converge	Sim
Classificador Combinado apresenta erro menor que classificadores C1 e C2	Sim
Desempenho do classificador combinado é melhor que EM para %Labeled < 5%	Sim
Desempenho do classificador combinado é melhor que EM e NB	Não
Base atende à premissa da auto-suficiência	Sim
Base atende à premissa da independência	Não

10.2 Leituras de Sonar

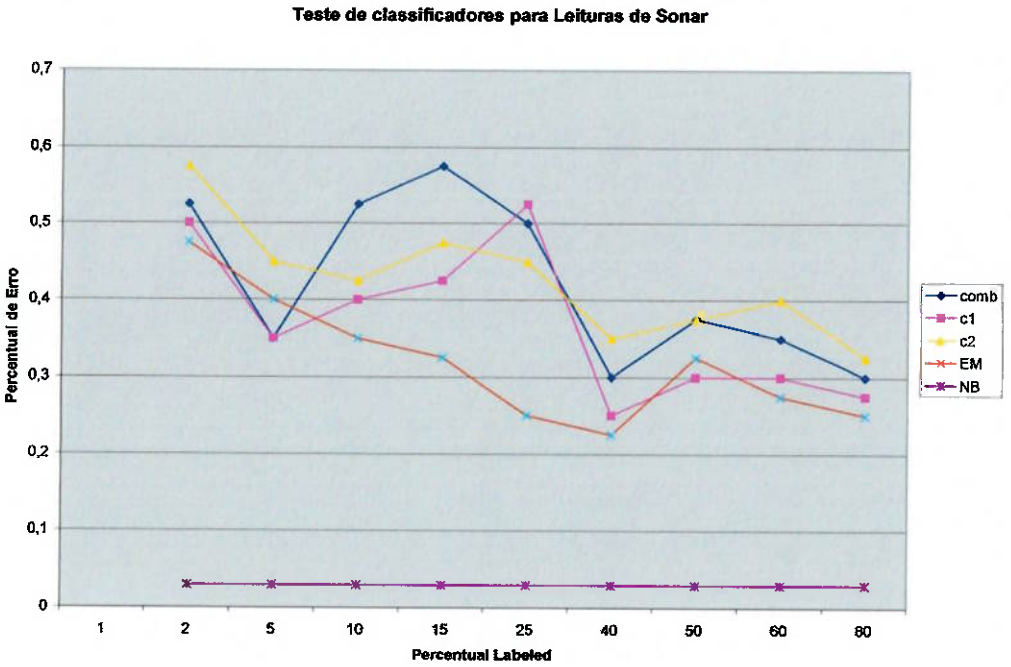


Figura 9- Teste de classificadores para Leituras de Sonar

Leituras de Sonar	
Classificador 1 converge	Não
Classificador 2 converge	Não
Classificador Combinado converge	Não
Classificador Combinado apresenta erro menor que classificadores C1 e C2	Não
Desempenho do classificador combinado é melhor que EM para %Labeled < 5%	Não
Desempenho do classificador combinado é melhor que EM e NB	Não
Base atende à premissa da auto-suficiência	Sim
Base atende à premissa da independencia	Não

10.3 Renda de adultos americanos

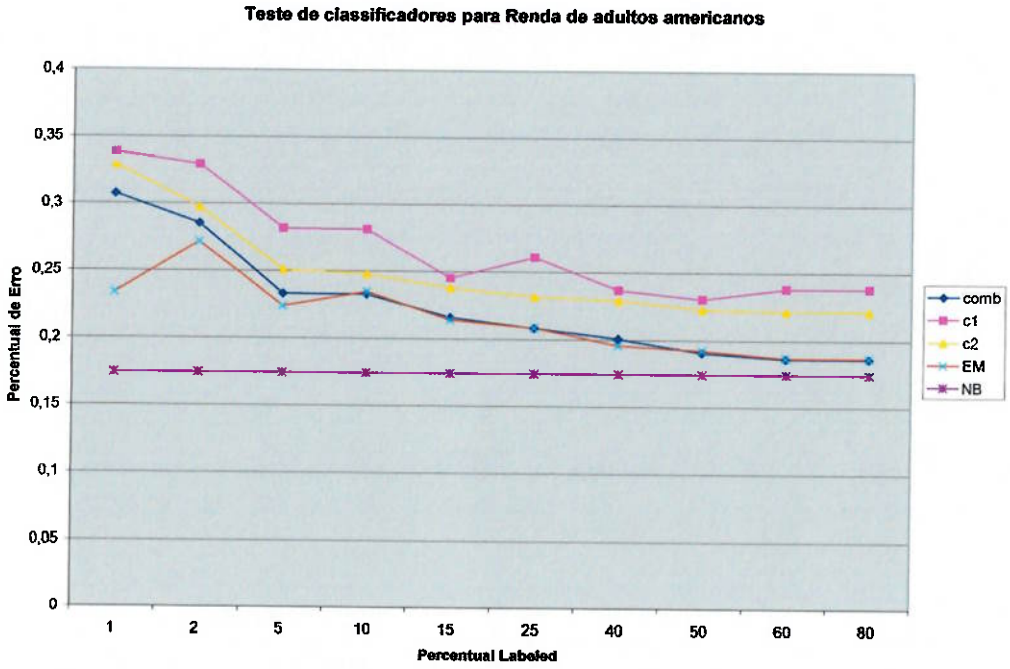


Figura 10- Teste de classificadores para Renda de adultos americanos

Renda de adultos	
Classificador 1 converge	Sim
Classificador 2 converge	Sim
Classificador Combinado converge	Sim
Classificador Combinado apresenta erro menor que classificadores C1 e C2	Sim
Desempenho do classificador combinado é melhor que EM para %Labeled < 5%	Não
Desempenho do classificador combinado é melhor que EM e NB	Não
Base atende à premissa da auto-suficiência	Sim
Base atende à premissa da independencia	Sim

10.4 Votos de congressistas americanos

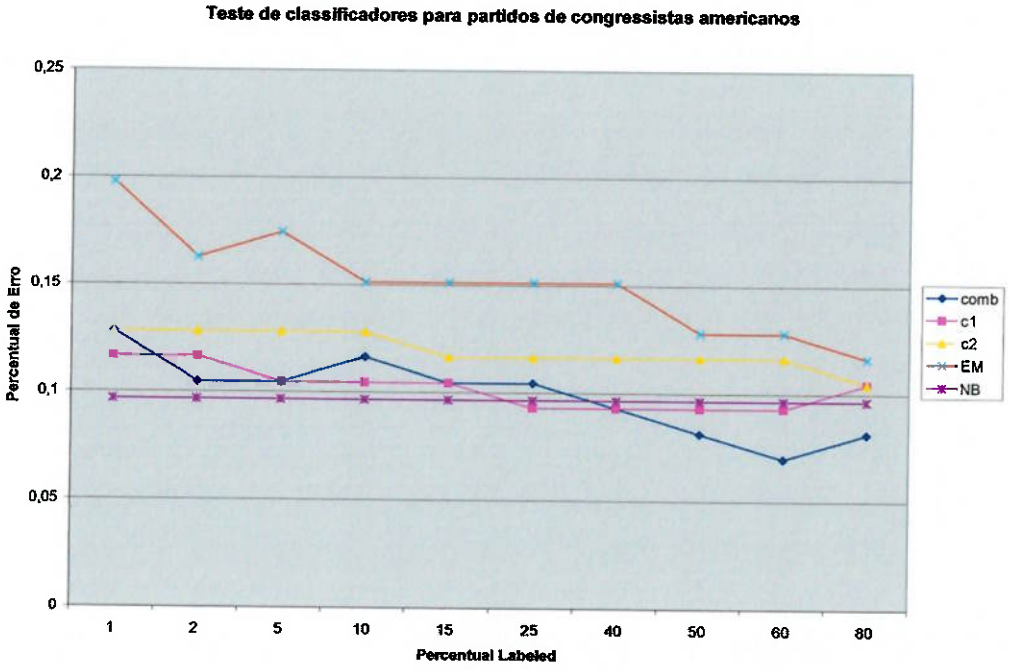


Figura 11- Teste de classificadores para partidos de congressistas americanos

Votos dos congressistas americanos	
Classificador 1 converge	Sim
Classificador 2 converge	Não
Classificador Combinado converge	Sim
Classificador Combinado apresenta erro menor que classificadores C1 e C2	Sim
Desempenho do classificador combinado é melhor que EM para %Labeled < 5%	Sim
Desempenho do classificador combinado é melhor que EM e NB	Sim
Base atende à premissa da auto-suficiência	Sim
Base atende à premissa da independencia	Não

10.5 Resultados de exames de câncer de mama

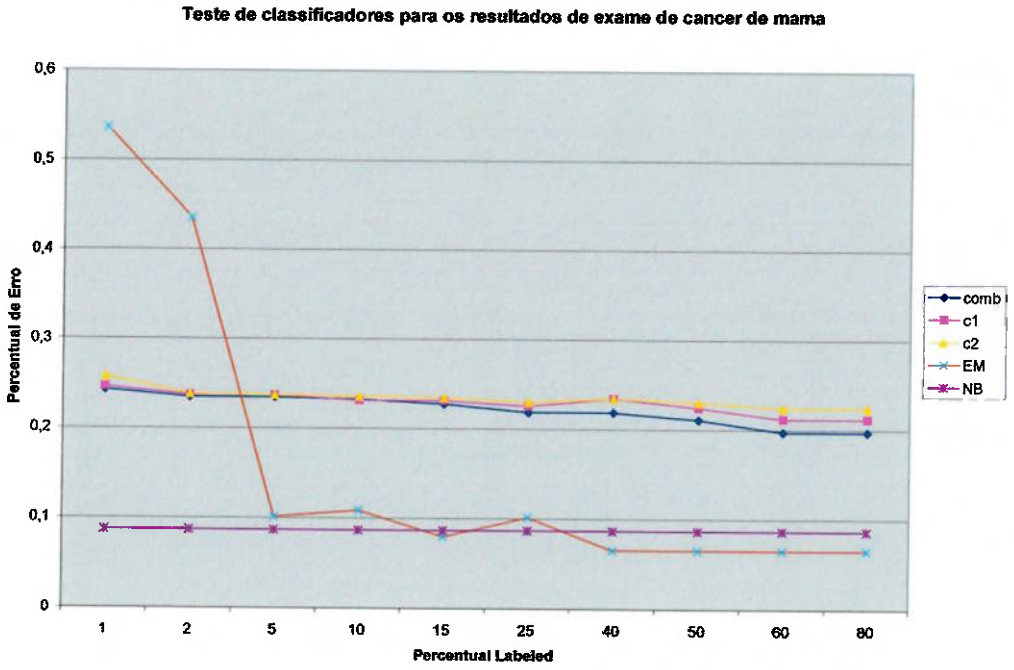


Figura 12- Teste de classificadores para os resultados de exame de câncer de mama

Resultados de exame de cancer de mama	
Classificador 1 converge	Sim
Classificador 2 converge	Sim
Classificador Combinado converge	Sim
Classificador Combinado apresenta erro menor que classificadores C1 e C2	Sim
Desempenho do classificador combinado é melhor que EM para %Labeled < 5%	Sim
Desempenho do classificador combinado é melhor que EM e NB	Não
Base atende à premissa de auto-suficiência	Sim
Base atende à premissa da independencia	Sim

10.6 Resultados de exames de doenças epiteliais

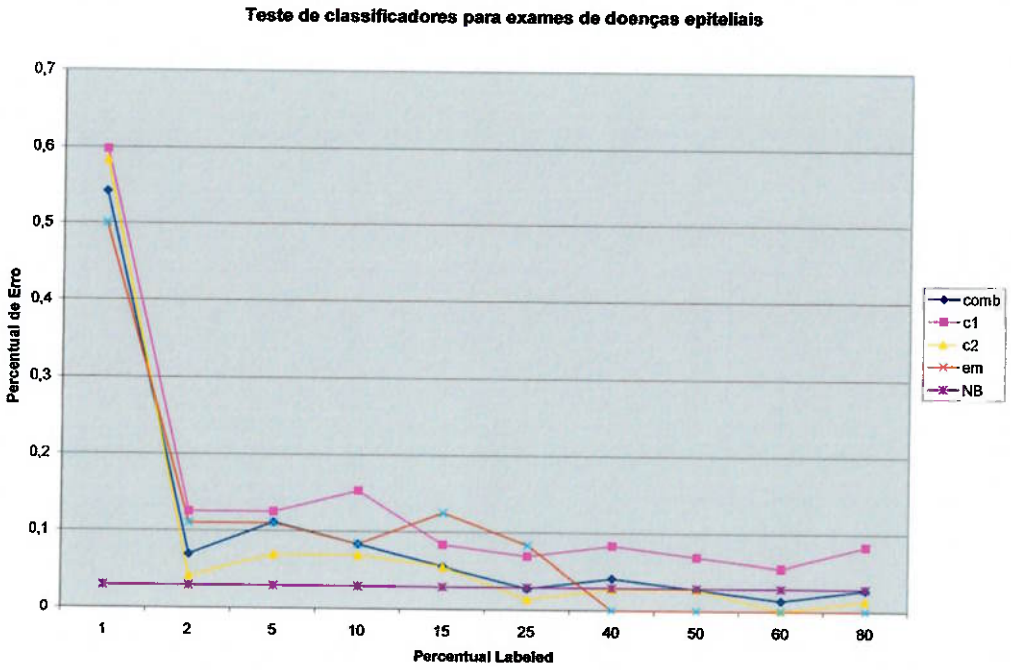


Figura 13- Teste de classificadores para exames de doenças epiteliais

Exames de doenças epiteliais	
Classificador 1 converge	Sim
Classificador 2 converge	Sim
Classificador Combinado converge	Sim
Classificador Combinado apresenta erro menor que classificadores C1 e C2	Não
Desempenho do classificador combinado é melhor que EM para %Labeled < 5%	Não
Desempenho do classificador combinado é melhor que EM e NB	Sim
Base atende à premissa da auto-suficiência	Sim
Base atende à premissa da independencia	Sim

10.7 Tomada de decisão para compra de um carro

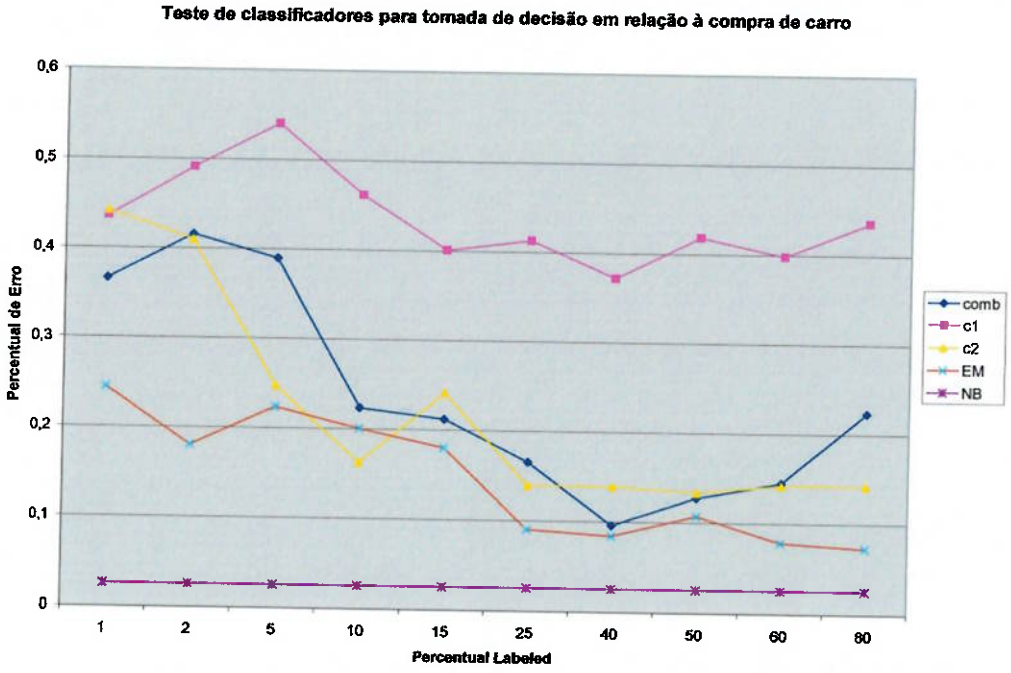


Figura 14- Teste de classificadores para tomada de decisão em relação à compra de carro

Tomada de decisão para compra de carro	
Classificador 1 converge	Não
Classificador 2 converge	Sim
Classificador Combinado converge	Sim
Classificador Combinado apresenta erro menor que classificadores C1 e C2	Não
Desempenho do classificador combinado é melhor que EM para %Labeled < 5%	Não
Desempenho do classificador combinado é melhor que EM e NB	Não
Base atende à premissa da auto-suficiência	Não
Base atende à premissa da independencia	Sim

10.8 Imagens

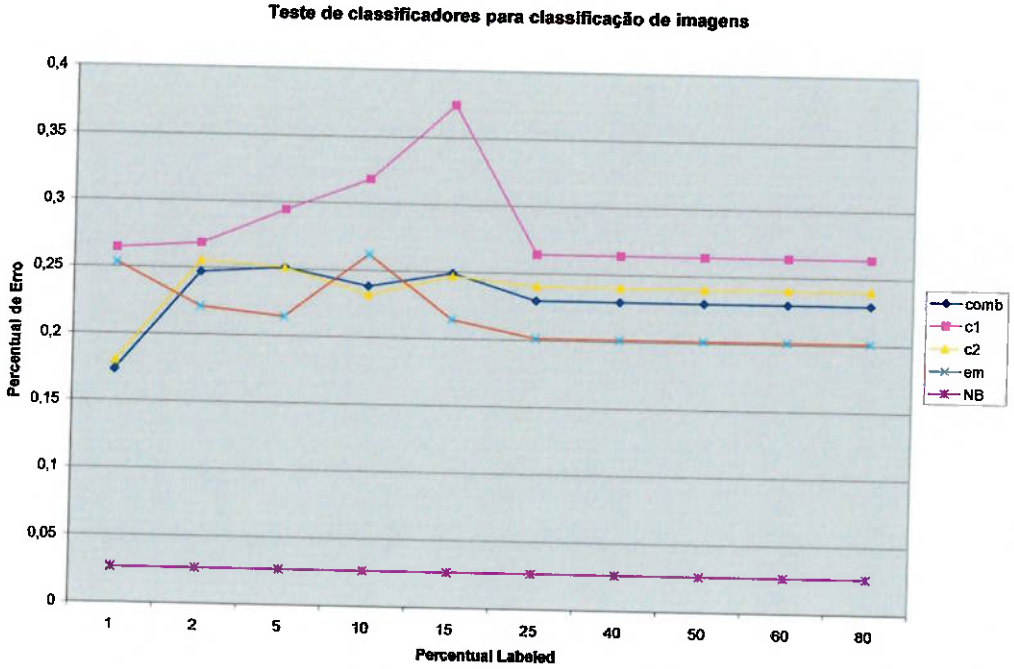


Figura 15- Teste de classificadores para classificação de imagens

Imagens	
Classificador 1 converge	Não
Classificador 2 converge	Não
Classificador Combinado converge	Não
Classificador Combinado apresenta erro menor que classificadores C1 e C2	Não
Desempenho do classificador combinado é melhor que EM para %Labeled < 5%	Sim
Desempenho do classificador combinado é melhor que EM e NB	Não
Base atende à premissa da auto-suficiência	Não
Base atende à premissa da independencia	Não

10.9 Partida de Xadrez kr x kp (a7)

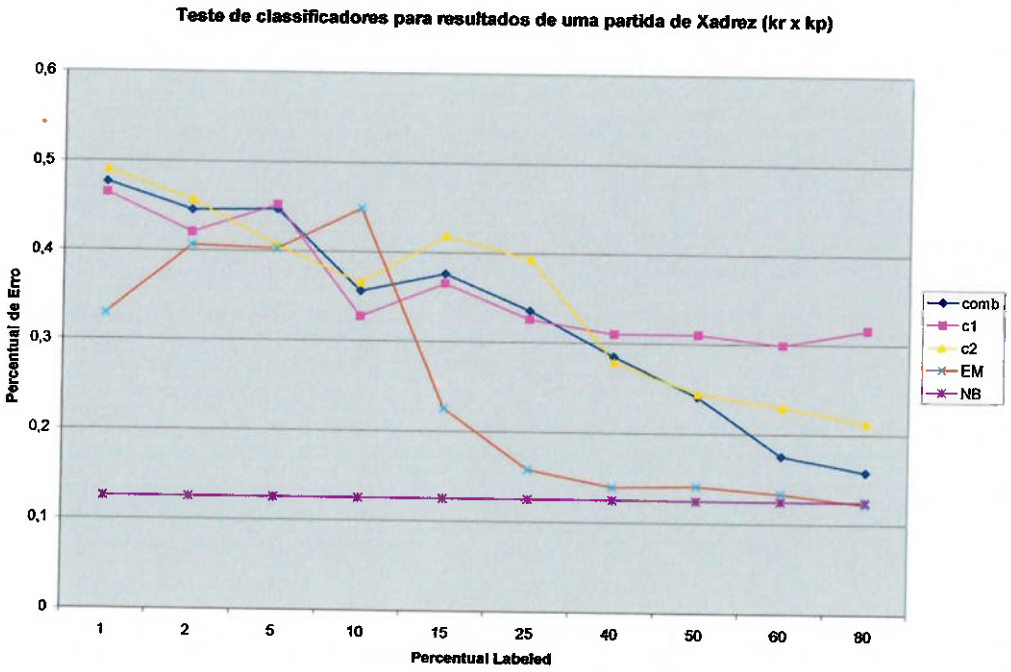


Figura 16- Teste de classificadores para resultados de uma partida de Xadrez (kr x kp)

Xadrez	
Classificador 1 converge	Sim
Classificador 2 converge	Sim
Classificador Combinado converge	Sim
Classificador Combinado apresenta erro menor que classificadores C1 e C2	Sim
Desempenho do classificador combinado é melhor que EM para %Labeled < 5%	Sim
Desempenho do classificador combinado é melhor que EM e NB	Não
Base atende à premissa da auto-suficiência	Não
Base atende à premissa da independencia	Sim

11 Casos práticos de utilização de classificadores automáticos

O algoritmo de classificação conhecido como Co-Training pode ser eficaz para alguns tipos de classificações. Dentre tais tipos, destaca-se por exemplo a classificação de Web Pages através de seu conteúdo e seus Hiperlinks (que apontam para ele), de modo que seus dois classificadores serão baseados nestas informações respectivamente. O problema da classificação de Web Pages é um problema atual e de demasiada importância para a grande maioria das pessoas que utiliza a Web como fonte de informação para pesquisas. Alguns experimentos demonstram um resultado satisfatório para este tipo de classificação de Web Pages.

Outra interessante aplicação seria o uso em robôs que possuem as seguintes fontes de dados para realizar uma classificação que auxiliará na tomada de decisão: visão , sonar e laser range.Considere o problema de distinguir se uma passagem está livre ou não para a passagem do robô, a quantidade de dados não classificados (Unlabeled) é infinitamente grande, enquanto a quantidade de passagens livres é bastante limitada. Poder-se-ia facilmente treinar um classificador para classificar passagens como livres ou não, utilizando para tal o Co-Training.

Atualmente, temos um grande foco na aplicações que utilizam o algoritmo co-training para a distinção entre e-mails desejados e e-mails indesejados (Spam). Entende-se que o algoritmo apresente uma performance excelente para este tipo de dado, pois ele atende rigorosamente às premissas do co-training. Para tal, analisamos que a divisão Assunto e corpo do e-mail é aquele mais lógico, de forma que ambas as informações são por si só suficientes para a classificação do Spam e também podem ser consideradas razoavelmente independentes.

12 Conclusões Finais

Neste trabalho apresentou-se o contexto atual da Classificação, alguns algoritmos utilizados como Naive Bayes e EM e o algoritmo de aprendizado semi-supervisionado Co-Training, sua implementação e resultados obtidos após testes.

O algoritmo Co-Training necessita de um pequeno conjunto de dados rotulados e um conjunto maior de dados não rotulados para a sua execução. Os dois conjuntos de dados devem estar representados por duas descrições diferentes desse dados. Com as duas descrições iniciais do pequeno conjunto de exemplos rotulados, Co-Training induz dois classificadores que são utilizados para rotular uns poucos exemplos de sua respectiva descrição. Após, os exemplos correspondentes que foram rotulados com maior certeza por cada um desses classificadores são adicionados ao conjunto de exemplos rotulados e o processo se repete até não ser mais possível rotular mais exemplos ou outro de critério de parada ser atingido.

O algoritmo foi avaliado considerando a precisão dos classificadores induzidos e/ou o classificador combinado. Ponderou-se também a precisão dos classificadores parciais e combinado do co-training com os classificadores NB e EM.

Nesta etapa final da análise dos resultados, consolidou-se todas as bases analisadas em um gráfico consolidado. Para tal, foi feita a média aritmética dos erros para cada uma das combinações base de dados e porcentagem Labeled. Diante dos resultados obtidos, optou-se por apresentá-los de uma maneira gráfica que facilitasse a interpretação. Assim sendo, foram traçadas linhas de tendência para cada um dos classificadores. Os resultados são apresentados a seguir.

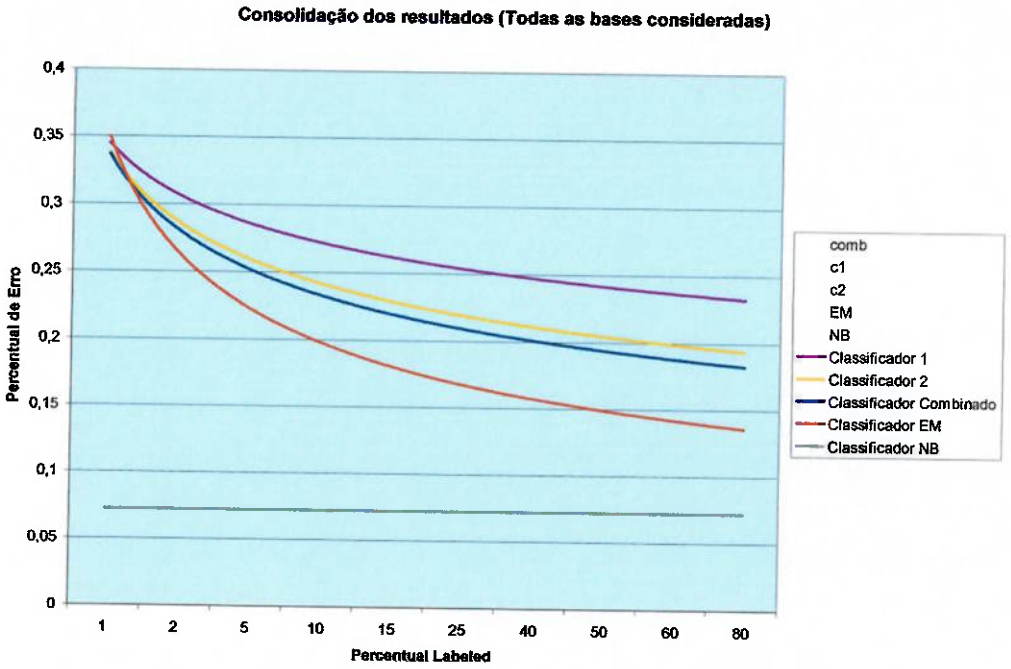


Figura 17- Consolidação dos resultados (Todas as bases testadas)

Observando o gráfico resultando, pode-se traçar algumas conclusões iniciais sobre o desempenho do algoritmo co-training de maneira geral. Primeiramente, observamos que o desempenho do classificador combinado é superior ao desempenho de cada um dos classificadores parciais. Entretanto, o Co-Training não supera o desempenho obtido pelo algoritmo EM para nenhuma combinação de dados Labeled e Unlabeled. O desempenho é insatisfatório quando comparado ao obtido pelo classificador NB, porém sabemos que o custo para a obtenção da quantidade de registros Labeled requerida pelo NB é consideravelmente alto.

Contemplando o resultado anterior, pode-se pensar em uma análise de dados mais detalhada. Sabemos que algumas bases utilizadas para a geração do gráfico anterior não estão completamente adequadas às premissas do algoritmo Co-Training. Desta forma, tais bases poderiam afetar o resultado obtido. Assim sendo, após observar a analise feita individualmente para cada uma das bases, concluiu-se que seria interessante remover algumas delas do resultado final. Haja vista que estas claramente não atendem às premissas do co-training e seus resultados

individuais não estiveram de acordo com o esperado, remover-se-á as mesmas da versão final do gráfico consolidado. Espera-se que desta maneira um resultado mais fidedigno possa ser observado.

Analisando-se criteriosamente as bases testadas, chegou à conclusão que as seguintes bases poderiam induzir o resultado e portanto deveria ser excluídas do resultado final:

- ✓ Leitura de Sonar

  Sabe-se que a base de leituras de sonar apresenta os dados de energia para determinadas freqüências. Diante de tal fato, podemos concluir que esta base não atende ao critério de auto-suficiência, pois é possível que somente as freqüências medidas em um dos agrupamentos de dados responda pela correta classificação quanto à conclusão sobre ser uma mina ou uma rocha.

- ✓ Carros

  Observa-se que claramente a opção por decidir sobre a compra ou não do carro depende dos parâmetros utilizados pelo classificador 2. Desta forma, conclui-se que a incapacidade de classificar a base apresentada pelo classificador 1 poderia induzir à uma classificação errônea.

- ✓ Imagens

  Observando a divisão dos dados, temos que todas as informações referentes às cores encontram-se no segundo agrupamento de dados. Desta maneira, concluímos que é impossível realizar a classificação utilizando somente o primeiro agrupamento de dados, haja vista que para a correta entre uma imagem do céu e outras imagens é devéras trivial a informação sobre a cor.

- ✓ Xadrez

  Entende-se que para que seja tomada uma decisão correta em relação à vitória da equipe branca é necessário analisar o tabuleiro de xadrez como um todo. Se dividirmos o

> tabuleiro de xadrez em duas partes e analisarmos cada uma destas individualmente, teremos para o jogo definido nesta base dados insuficientes dados para a classificação a ser feita por um dos classificadores individuais. Tal indefinição se dará pelo fato da existência de somente quatro peças no jogo, sendo que para que a realização de uma correta classificação temos que ter pleno conhecimento da posição de todas elas, para que assim se possa induzir com uma confiança razoável o resultado da partida.

Dados os fatos acima mencionados, reconstruíu-se a tabela com a média dos erros x percentuais Labeled, de forma a obter um resultado mais fidedigno. Após a filtragem das bases, obteve-se um segundo resultado conclusivo. Tal resultado apresentou uma importante diferença em relação ao primeiro, que será explicada a seguir.

Primeiramente observamos que o classificador combinado apresenta um desempenho melhor que cada um dos classificadores do co-training individualmente, estando desta maneira de acordo com o esperado. Diante da observação dos resultados para percentuais de Labeled inferiores a 5%, temos que o desempenho do Co-Training é melhor que o apresentado pelo EM. Conclui-se desta maneira que diante de bases que atendem rigorosamente os pré-requisitos do co-training, este apresentará um desempenho melhor que o algoritmo EM para pequenos percentuais de dados Labeled.

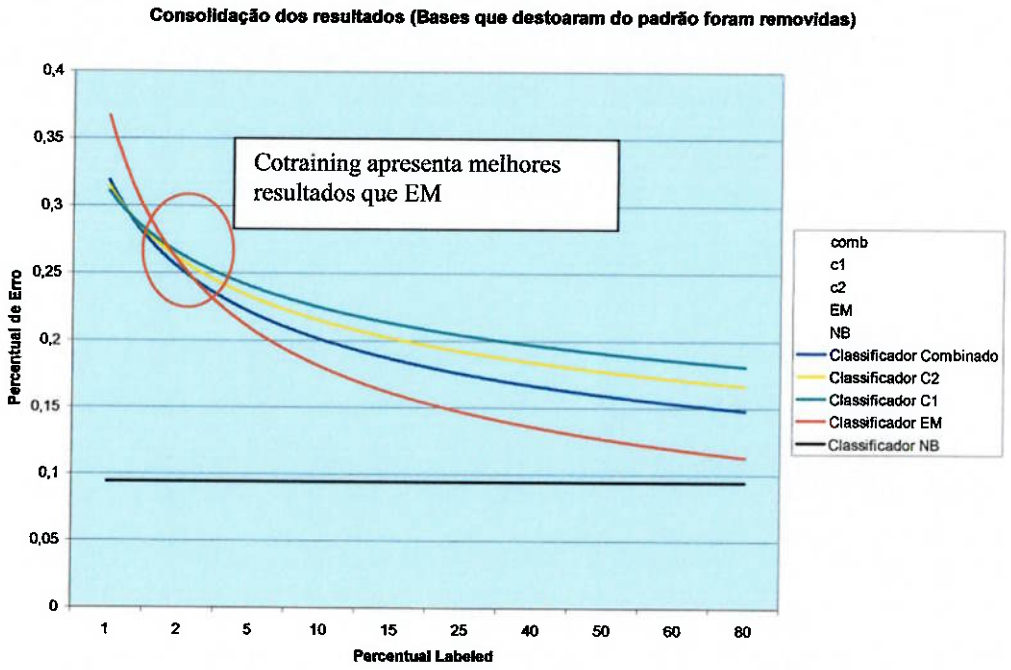


Figura 18- Consolidação dos resultados (Somente bases que previamente atenderam aos pré-requisitos

Entretanto, também considera-se que o uso de Co-Training, ou qualquer outro algoritmo de aprendizado semi-supervisionado, que têm como principal objetivo rotular quando há poucos exemplos rotulados disponíveis, deve ser usado com precaução. O motivo principal é que esses algoritmos podem rotular erroneamente alguns exemplos; como esse exemplos são incorporados ao conjunto de exemplos rotulados e o processo é repetido, esses erros serão propagados nas próximas iterações do algoritmo o qual rotulará com alta probabilidade, mais exemplos errados. Dessa maneira, o classificador resultante será induzido observando exemplos que descrevem erroneamente instâncias do conceito a ser aprendido.

13 Bibliografia

[1] BLUM, A.; MITCHELL T. **Combining Labeled and Unlabeled Data with Co-Training**. 1998. 9p. Proceedings of the 1998 Conference on Computational Learning Theory.

[2] MATSUBARA, E. A. **O algoritmo de aprendizado semi-supervisionado co-training e sua aplicação na rotulação de documentos**. 2004. 105 p., Dissertação (Mestrado), ICMC, Universidade de São Paulo. São Paulo, 2004.

[3] NIGAM, K. et al. **Text Classification from Labeled and Unlabeled Documents using EM**. 1999. 34p.

[4] NIGAM, K.; GHANI R. **Analyzing the Effectiveness and Applicability of Co-training**. 2000. Ninth International Conference on Information and Knowledge Management, pp. 1-8.

[5] STUART, R.; NORVIG, P. **Inteligência Artificial**. Rio de Janeiro: Editora Campus, 2004. Cap. 13, 14, 20. pp. 449-501, 690-709.

[6] UCI MACHINE LEARNING REPOSITORY. **Repositório de bases, teorias de domínio e geradores de base.** Disponível em: http://www.ics.uci.edu/~mlearn/MLRepository.html . Acesso em: dez/2004.

[7] WEKA TUTORIAL. *www.cs.waikato.ac.nz/ml/weka*

[8] PORTAL JAVA (JDBC)
http://www.portaljava.com/home/modules.php?name=Content&pa=showpage&pid=5

14 Anexo I – Fontes

14.1 Cotraining

```
/**
 * Main class for learning classifiers
 * for the labeled-unlabeled data problem.
 * Author: Alan Glezer
 */

import lutil.*;
import nan.*;
import preCotraining.*;
//import tan.*;

import BayesianNetworks.*;

import java.io.*;
import java.util.*;

public class CoTraining {

      static File unlabeledDataFile1;
      static BufferedReader unlabeledReader1;
      static File unlabeledDataFile2;
      static BufferedReader unlabeledReader2;

  public static void main(String args[]) {

                  System.out.println("\n/**** CoTraining ****/");
                  System.out.println("\t// Program to learn a CoTraining classifier");
                  //System.out.println("\t// Accepts only categorical data");

                  if (args.length < 1) {
                     System.out.print("\t Usage: java NB <parameter filename>");
                     System.exit(0);
                  }

                  Vector pars = null;

                  try {
                     pars = readParameters(args[0]);
                  }
                  catch (IOException e) {
                     System.out.println("Problem in reading parameters!");
                     System.exit(-1);
                  }

                  String trainingFilename = getPar("Training:", pars);

                  if (trainingFilename == null) {
                     System.out.println("\t Insert filename with training data");
                     System.exit(-1);
                  }

                  String CoTrainingDL1 = getPar("CoTrainingLabeledData1:", pars);
                  String CoTrainingDL2 = getPar("CoTrainingLabeledData2:", pars);
                  String CoTrainingDU1 = getPar("CoTrainingUnlabeledData1:", pars);
                  String CoTrainingDU2 = getPar("CoTrainingUnlabeledData2:", pars);

                  String typeClassifier = getPar("Type:" , pars);
                  int numberIterations = getInt("MaxIterations:", pars, 1);
                  double likelihoodChange = getDouble("MinLikelihood:", pars, 0.00001);
                  String testFilename1 = getPar("Test1:", pars);
                  String testFilename2 = getPar("Test2:", pars);
                  String startingPointFilename = getPar("StartingPoint:", pars);
                  boolean useOnlyStartingStructure = getBoolean("OnlyStartingStructure:", pars, false);
                  boolean discardUnlabeledData = getBoolean("DiscardUnlabeledData:", pars, false);
```

```
boolean removeZerosFromStartingPoint = getBoolean("RemoveZerosFromStartingPoint:", pars, false);

String outFilename1 = getPar("Output1:", pars);
String outFilename2 = getPar("Output2:", pars);
String outFilenameComb = getPar("OutputComb:", pars);

int iContQtdLabeled = 10;


while (iContQtdLabeled > 0){

        String strNomedaBase = ".satimages1211nb";

        int iPercLabeled = 0;

        if (iContQtdLabeled == 10) iPercLabeled = 1 ;
        if (iContQtdLabeled == 9) iPercLabeled = 2 ;
        if (iContQtdLabeled == 8) iPercLabeled = 5 ;
        if (iContQtdLabeled == 7) iPercLabeled = 10 ;
        if (iContQtdLabeled == 6) iPercLabeled = 15 ;
        if (iContQtdLabeled == 5) iPercLabeled = 25 ;
        if (iContQtdLabeled == 4) iPercLabeled = 40 ;
        if (iContQtdLabeled == 3) iPercLabeled = 50 ;
        if (iContQtdLabeled == 2) iPercLabeled = 60 ;
        if (iContQtdLabeled == 1) iPercLabeled = 80 ;

        iContQtdLabeled--;

        preCotraining preCotra = new preCotraining();
        preCotra.preparaArquivos(iPercLabeled);

        PrintStream out1 = null;
        try {
          if (outFilename1 != null) {
                        out1 = new PrintStream(new FileOutputStream(new File(  outFilename1 + iPercLabeled +
strNomedaBase )));
          }
          else
                        out1 = new PrintStream(System.out);
        }
        catch (IOException e) {
          System.out.println("\t Problem opening output file 1.");
        }

        PrintStream out2 = null;
        try {
                if (outFilename2 != null) {
                        out2 = new PrintStream(new FileOutputStream(new File( outFilename2 + iPercLabeled  +
strNomedaBase )));
                }
                else
                        out2 = new PrintStream(System.out);
        }
        catch (IOException e) {
                System.out.println("\t Problem opening output file 2.");
        }

        PrintStream outComb = null;
        try {
                if (outFilenameComb != null) {
                        outComb = new PrintStream(new FileOutputStream(new File( outFilenameComb + iPercLabeled  +
strNomedaBase )));
                }
                else
                        outComb = new PrintStream(System.out);
        }
        catch (IOException e) {
                System.out.println("\t Problem opening Combined output file.");
        }

        InputStream startingPointStream = null;
        try {
```

```
    if (startingPointFilename != null) {
                startingPointStream = new FileInputStream(new File(startingPointFilename));
    }
  }
  catch (IOException e) {
    System.out.println("\t Problem opening starting point network.");
  }

  BayesNet startingPointNetwork = null;

  try {
    if (startingPointStream != null) {
                startingPointNetwork = new BayesNet(startingPointStream);
                out1.println("\n//Starting point:");
                startingPointNetwork.print(out1);
    }
  }
  catch (Exception e) {
    System.out.println("\t Problem reading starting point network.");
  }

  System.out.println("\n//Parameters: \n//Training filename: " +
                trainingFilename + "\n//CoTrainingLabeledData1: " +
                CoTrainingDL1 + "\n//CoTrainingLabeledData1: " +
                CoTrainingDL2 + "\n//CoTrainingUnlabeledData2: " +
                CoTrainingDU1 + "\n//CoTrainingUnlabeledData1: " +
                CoTrainingDU2 + "\n//Maximum number of iterations: " +
                numberIterations + "\n//Minimum likelihood change: " +
                likelihoodChange + "\n//Output filename: " +
                outFilename1 + "\n//Test filename: " +
                testFilename1 + "\n//Starting point filename: " +
                startingPointFilename +
                "\n//Use only starting point structure? " +
                useOnlyStartingStructure + "\n\n");
/*
  if (trainingFilename.startsWith("#")) {
    String trainingInfo = trainingFilename.substring(1);
    int labeledPoints[] = new int[] { 30, 300, 3000 };
    int unlabeledPoints[] = new int[] { 0, 30, 300, 3000, 30000 };
    int i, j, k;
    for (i=0; i<labeledPoints.length; i++) {
                for (j=0; j<unlabeledPoints.length; j++) {
                for (k=1; k<=10; k++) {
                        trainingFilename =
                        trainingInfo + "L" + labeledPoints[i] +
                        "U" + unlabeledPoints[j] + "Trial" + k + ".in";
                        out.println("// ********************************** ");
                        out.println("// Processing " + trainingFilename);
                        System.out.println("// Processing " + trainingFilename);
                        runClassifier(typeClassifier,
                                startingPointNetwork, useOnlyStartingStructure,
                                trainingFilename, discardUnlabeledData,
                                numberIterations, likelihoodChange,
                                removeZerosFromStartingPoint,
                                testFilename, out);
                }
                }
    }
  }
  else {
    out.println("// ********************************** ");
    out.println("// Processing " + trainingFilename);
    System.out.println("// Processing " + trainingFilename);
    runClassifier(typeClassifier,
                startingPointNetwork, useOnlyStartingStructure,
                trainingFilename, discardUnlabeledData,
                numberIterations, likelihoodChange,
                removeZerosFromStartingPoint,
                testFilename, out);
  }

  out.close();
*/
```

```
BayesNet classificador1 = new BayesNet();
BayesNet classificador2 = new BayesNet();

trainingFilename = CoTrainingDL1;

out1.println("// ********************************** ");
out1.println("// Processing " + trainingFilename);
System.out.println("// Processing " + trainingFilename);
classificador1 = runClassifier(typeClassifier,
        startingPointNetwork, useOnlyStartingStructure,
        trainingFilename, discardUnlabeledData,
        numberIterations, likelihoodChange,
        removeZerosFromStartingPoint,
        null, null);

trainingFilename = CoTrainingDL2;

out2.println("// ********************************** ");
out2.println("// Processing " + trainingFilename);
System.out.println("// Processing " + trainingFilename);
classificador2 = runClassifier(typeClassifier,
        startingPointNetwork, useOnlyStartingStructure,
        trainingFilename, discardUnlabeledData,
        numberIterations, likelihoodChange,
        removeZerosFromStartingPoint,
        null, null);

ReadIn unlabeledDataFile1 = new ReadIn(CoTrainingDU1);
ReadIn unlabeledDataFile2 = new ReadIn(CoTrainingDU2);

int registroMaisProvavel1[];
int registroMaisProvavel2[];
int contador = 0;

while(unlabeledDataFile1.hasNextRecord() == true && unlabeledDataFile2.hasNextRecord() == true)
{
        contador++;
        System.out.println(contador);

        if(contador == 107)
                System.out.println("contador = " + contador);

        unlabeledDataFile1.close();
        unlabeledDataFile2.close();

        unlabeledDataFile1 = new ReadIn(CoTrainingDU1);
        registroMaisProvavel1 = TesteCoTraining.test(unlabeledDataFile1, classificador1);
        unlabeledDataFile1.close();

        if(registroMaisProvavel1 != null)
                EditaArquivos(registroMaisProvavel1, CoTrainingDL1, CoTrainingDL2, CoTrainingDU1,
CoTrainingDU2);

        unlabeledDataFile2 = new ReadIn(CoTrainingDU2);
        registroMaisProvavel2 = TesteCoTraining.test(unlabeledDataFile2, classificador2);
        unlabeledDataFile2.close();

        if(registroMaisProvavel2 != null)
                EditaArquivos(registroMaisProvavel2, CoTrainingDL1, CoTrainingDL2, CoTrainingDU1,
CoTrainingDU2);

        unlabeledDataFile1 = new ReadIn(CoTrainingDU1);
        unlabeledDataFile2 = new ReadIn(CoTrainingDU2);

        trainingFilename = CoTrainingDL1;
        classificador1 = runClassifier(typeClassifier,
                                                        startingPointNetwork,
useOnlyStartingStructure,
                                                        trainingFilename,
discardUnlabeledData,
                                                        numberIterations, likelihoodChange,
                                                        removeZerosFromStartingPoint,
```

```
                                                                                    null, null);

                        trainingFilename = CoTrainingDL2;
                        classificador2 = runClassifier(typeClassifier,
                                                                                    startingPointNetwork,
useOnlyStartingStructure,
                                                                                    trainingFilename,
discardUnlabeledData,
                                                                                    numberIterations, likelihoodChange,
                                                                                    removeZerosFromStartingPoint,
                                                                                    null, null);

                }

                unlabeledDataFile1.close();
                unlabeledDataFile2.close();

                //Redes bayesianas finais, que sï¿½o gravadas nos arquivos de saï¿½da
                trainingFilename = CoTrainingDL1;
                classificador1 = runClassifier(typeClassifier,
                                                                    startingPointNetwork, useOnlyStartingStructure,
                                                                    trainingFilename, discardUnlabeledData,
                                                                    numberIterations, likelihoodChange,
                                                                    removeZerosFromStartingPoint,
                                                                    testFilename1, out1);

                trainingFilename = CoTrainingDL2;
                classificador2 = runClassifier(typeClassifier,
                                                                    startingPointNetwork, useOnlyStartingStructure,
                                                                    trainingFilename, discardUnlabeledData,
                                                                    numberIterations, likelihoodChange,
                                                                    removeZerosFromStartingPoint,
                                                                    testFilename2, out2);

                NAN.combinedTest(testFilename1,
                                                    testFilename2,
                                                    classificador1,
                                                    classificador2,
                                                    outComb);

                System.out.println("Acabou para " + iPercLabeled + "% de labeled");

        }
}

private static boolean EditaArquivos(   int[] posicoes,
                                                                    String CoTrainingDL1,
                                                                    String CoTrainingDL2,
                                                                    String CoTrainingDU1,
                                                                    String CoTrainingDU2){
        boolean flag = true;
        ReadIn readDU1;
        ReadIn readDU2;


        //Adiciona linhas nos arquivos labeled
        try {
                File dataFile1 = new File(CoTrainingDL1);
                PrintWriter prtWriter1 = new PrintWriter(new BufferedWriter(new FileWriter(dataFile1, true)));
                File dataFile2 = new File(CoTrainingDL2);
                PrintWriter prtWriter2 = new PrintWriter(new BufferedWriter(new FileWriter(dataFile2, true)));

                int classe, aux;

                for (classe = 0; classe < posicoes.length; classe++) {

                        int[] registro1 = {}, registro2 = {};

                        readDU1 = new ReadIn(CoTrainingDU1);
                        readDU2 = new ReadIn(CoTrainingDU2);

                        for (aux = 0; aux <= posicoes[classe]; aux++){
                                registro1 = readDU1.readNextRecord();
```

```
                        registro2 = readDU2.readNextRecord();

                }

                String strRegistro1 = intArraytoString(registro1);
                String strRegistro2 = intArraytoString(registro2);

                strRegistro1 = atribuiClassePrevista(strRegistro1, classe);
                strRegistro2 = atribuiClassePrevista(strRegistro2, classe);

                prtWriter1.write("\n" + strRegistro1);
                prtWriter2.write("\n" + strRegistro2);

                readDU1.close();
                readDU2.close();

        }

        prtWriter1.close();
        prtWriter2.close();

}

catch (IOException e) {
        System.out.println("Exception when opening write file" + e);
        System.exit(-1);
}

//Remove linhas dos arquivos unlabeled
readDU1 = new ReadIn(CoTrainingDU1);
readDU2 = new ReadIn(CoTrainingDU2);

try {
        /*
        File dataFile1 = new File(CoTrainingDU1);
        File dataFile2 = new File(CoTrainingDU2);
        boolean data1 = dataFile1.delete();
        boolean data2 = dataFile2.delete();

        dataFile1 = new File(CoTrainingDU1);
        PrintWriter prtWriter1 = new PrintWriter(new BufferedWriter(new FileWriter(dataFile1, true)));
        dataFile2 = new File(CoTrainingDU2);
        PrintWriter prtWriter2 = new PrintWriter(new BufferedWriter(new FileWriter(dataFile2, true)));
        */
        File writeFile1 = File.createTempFile("temp1", "in", new File("./testing/"));
        File writeFile2 = File.createTempFile("temp2", "in", new File("./testing/"));
        PrintWriter prtWriter1 = new PrintWriter(new BufferedWriter(new FileWriter(writeFile1, true)));
        PrintWriter prtWriter2 = new PrintWriter(new BufferedWriter(new FileWriter(writeFile2, true)));

//      escreve primeira linha do cabeï¿½calho no arquivo U1
        for(int i = 0; i < readDU1.getNames().length; i++){
                if(i == 0)
                        prtWriter1.write(readDU1.getNames()[i]);
                else
                        prtWriter1.write("," + readDU1.getNames()[i]);
        }
        prtWriter1.write("\n");

//      escreve primeira linha do cabeï¿½calho no arquivo U2
        for(int i = 0; i < readDU2.getNames().length; i++){
                if(i == 0)
                        prtWriter2.write(readDU2.getNames()[i]);
                else
                        prtWriter2.write("," + readDU2.getNames()[i]);
        }
        prtWriter2.write("\n");

//      escreve segunda linha do cabeï¿½calho no arquivo U1
        for(int i = 0; i < readDU1.getNumberOfValues().length; i++){
                if(i == 0)
                        prtWriter1.write(new Integer(readDU1.getNumberOfValues()[i]).toString());
                else
```

```
                                        prtWriter1.write("," + new Integer(readDU1.getNumberOfValues()[i]).toString());

                        }
                        prtWriter1.write("\n\n");

//                      escreve segunda linha do cabeï¿½calho no arquivo U2
                        for(int i = 0; i < readDU2.getNumberOfValues().length; i++){
                                if(i == 0)
                                        prtWriter2.write(new Integer(readDU2.getNumberOfValues()[i]).toString());
                                else
                                        prtWriter2.write("," + new Integer(readDU2.getNumberOfValues()[i]).toString());

                        }
                        prtWriter2.write("\n\n");


                        for(int pos = 0; readDU1.hasNextRecord() && readDU2.hasNextRecord(); pos++){

                                boolean isDeleted = false;
                                int[] record1, record2;

                                for(int i = 0; i < posicoes.length; i++){
                                        if(posicoes[i] == pos)
                                                isDeleted = true;
                                }

                                record1 = readDU1.readNextRecord();
                                record2 = readDU2.readNextRecord();

                                if(!isDeleted){
                                        prtWriter1.write(intArraytoString(record1) + "\n");
                                        prtWriter2.write(intArraytoString(record2) + "\n");

                                }
                        }

                        prtWriter1.close();
                        prtWriter2.close();

                        readDU1.close();
                        readDU2.close();

                        File unlabeledFile1 = new File(CoTrainingDU1);
                        File unlabeledFile2 = new File(CoTrainingDU2);
                        boolean data1 = unlabeledFile1.delete();
                        boolean data2 = unlabeledFile2.delete();

                        data1 = writeFile1.renameTo(new File(CoTrainingDU1));
                        data2 = writeFile2.renameTo(new File(CoTrainingDU2));

                }

                catch (IOException e) {
                        System.out.println("Exception when opening write file" + e);
                        System.exit(-1);
                }

                return(flag);
        }

        private static String intArraytoString(int[] array){

                String result = "";
                Integer aux;

                for(int i = 0; i < array.length; i++){

                        if(i == 0){
                                aux = new Integer(array[i]);
                                result = result.concat(aux.toString());
                        }
                        else{
                                aux = new Integer(array[i]);
```

```
                                result = result.concat("," + aux.toString());
                        }
                }

                return result;
        }

        private static String atribuiClassePrevista(String oper, int classe){

                classe++;

                StringTokenizer st = new StringTokenizer(oper, ",");
                String tokens[] = new String[st.countTokens()];
                for(int i = 0; i < tokens.length; i++)
                        tokens[i] = st.nextToken();
                tokens[0] = new Integer(classe).toString();

                oper = "";
                for (int i=0; i<tokens.length; i++) {
                        if(i != 0 )
                                oper = oper.concat("," + tokens[i]);
                        else
                                oper = oper.concat(tokens[i]);
                }

                return oper;
        }

        private static void openFile(String file1, String file2) throws IOException {

                unlabeledDataFile1 = new File(file1);
                unlabeledReader1 = new BufferedReader(new FileReader(unlabeledDataFile1));
                unlabeledDataFile2 = new File(file2);
                unlabeledReader2 = new BufferedReader(new FileReader(unlabeledDataFile2));

        }
          /*
           * Call appropriate classifier builder.
           */
        private static BayesNet runClassifier(String typeClassifier,
                                              BayesNet startingPointNetwork,
                                              boolean useOnlyStartingStructure,
                                              String trainingFilename,
                                             boolean discardUnlabeledData,
                                        int numberIterations,
                                        double likelihoodChange,
                                              boolean removeZerosFromStartingPoint,
                                                      String testFilename,
                                              PrintStream out) {

                BayesNet classificador = new BayesNet();

                classificador = NAN.run(startingPointNetwork, useOnlyStartingStructure,
                                trainingFilename, discardUnlabeledData,
                                numberIterations, likelihoodChange,
                                removeZerosFromStartingPoint, testFilename, out);

                return(classificador);
        }

        private static Vector readParameters(String filename) throws IOException {

                Vector pars = new Vector();
                File parameterFile = new File(filename);
                BufferedReader parameterReader = new BufferedReader(new FileReader(parameterFile));
                String line;
                while ((line = getString(parameterReader)) != null) {
                   StringTokenizer st = new StringTokenizer(line, "\t ,");
                   pars.addElement(new String[] { st.nextToken(), st.nextToken() });
                }

                parameterReader.close();
```

```
                return(pars);
        }

        private static String getString(BufferedReader dataReader) throws IOException {
                String nextLine;
                String trimmed;
                do {
                  nextLine = dataReader.readLine();
                  if (nextLine == null)
                        return(null);
                  trimmed = nextLine.trim();
                } while ( (trimmed.equals("")) || (trimmed.startsWith("%")) );

                return(nextLine);
        }

        private static String getPar(String key, Vector pars) {
                for (Enumeration e = pars.elements(); e.hasMoreElements(); ) {
                  String keyValue[] = (String[])(e.nextElement());
                  if (keyValue[0].equals(key))
                        return(keyValue[1]);
                }
                return(null);
    }

    private static int getInt(String key, Vector pars, int def) {
                String value = getPar(key, pars);
                if (value != null)
                  return( Integer.parseInt(value) );
                else
                  return( def );
    }

    private static double getDouble(String key, Vector pars, double def) {
                String value = getPar(key, pars);
                if (value != null)
                  return( Double.parseDouble(value) );
                else
                  return( def );
    }

    private static boolean getBoolean(String key, Vector pars, boolean def) {
                String value = getPar(key, pars);
                if (value != null)
                  return( Boolean.valueOf(value).booleanValue() );
                else
                  return( def );
    }
}
```

14.2 Precotraining

```
/*
 * Created on 27/04/2005
 *
 * To change the template for this generated file go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */

import java.io.*;
import java.sql.*;
import java.util.Random;

public class preCotraining {

      public boolean preparaArquivos (int porcLabeled){

                String url = "jdbc:odbc:cotrainingDB";

/*
                VOTES
```

```
                double dPercClass1 = 0.40 ;
                String strCabecalho1 = "C,F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F16";
                String strCabecalho2 = "C,F9,F10,F11,F12,F13,F14,F15,F16";
                String strClasses1 = "2,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3";
                String strClasses2 = "2,3,3,3,3,3,3,3,3";
*/

/*
                CARS

                double dPercClass1 = 0.7 ;
                String strCabecalho1 = "C,F1,F2,F3,F4,F5,F6";
                String strCabecalho2 = "C,F7,F8,F9";
                String strClasses1 = "2,4,4,5,5,3,3";
                String strClasses2 = "2,5,3,3";
*/

/*


                Breast - Cancer

                double dPercClass1 = 0.65 ;
                String strCabecalho1 = "C,F1,F2,F3,F4,F5,F6,F7,F8,F9";
                String strCabecalho2 = "C,F10,F11,F12,F13,F14";
                String strClasses1 = "2,11,11,11,11,11,11,11,11,11";
                String strClasses2 = "2,11,11,11,11,11";
 */

/*
                Chess

                double dPercClass1 = 0.52 ;
                String strCabecalho1 =
"C,F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F16,F17,F18,F19,F20,F21,F22,F23,F24,F25,F26,F27,F28,F29,F30,F31,F32,F33,F34,F35,F36";
                String strCabecalho2 = "C,F19,F20,F21,F22,F23,F24,F25,F26,F27,F28,F29,F30,F31,F32,F33,F34,F35,F36";
                String strClasses1 = "2,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7";
                String strClasses2 = "2,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7";
*/

/*
                Adults

                */
                double dPercClass1 = 0.75 ;
                String strCabecalho1 = "C,F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13";
                String strCabecalho2 = "C,F7,F8,F9,F10,F11,F12,F13";
                String strClasses1 = "2,9,8,16,7,7,14,6,5,2,15,15,6,41";
                String strClasses2 = "2,6,5,2,15,15,6,41";


/*
                Satimages

                double dPercClass1 = 0.21 ;
                String strCabecalho1 =
"C,F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F16,F17,F18,F19,F20,F21,F22,F23,F24,F25,F26,F27,F28,F29,F30,F31,F32,F33,F34,F35,F36";
                String strCabecalho2 = "C,F19,F20,F21,F22,F23,F24,F25,F26,F27,F28,F29,F30,F31,F32,F33,F34,F35,F36";
                String strClasses1 = "2,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10";
                String strClasses2 = "2,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10";
    */


/*

    Dermathology


                double dPercClass1 = 0.30 ;
                String strCabecalho1 =
"C,F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F16,F17,F18,F19,F20,F21,F22,F23,F24,F25,F26,F27,F28,F29,F30,F31,F32,F33,F34";
```

```
		String strCabecalho2 = "C,F35,F36,F37,F38,F39,F40,F41,F42,F43,F44,F45,F46,F47,F48,F49,F50,F51";
		String strClasses1 = "2,4,4,4,4,4,4,4,4,4,4,4,2,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4";
		String strClasses2 = "2,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4";
	*/

/*
	Sonar


		double dPercClass1 = 0.53 ;
		String strCabecalho1 =
"C,F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F16,F17,F18,F19,F20,F21,F22,F23,F24,F25,F26,F27,F28,F29,F30,F31,F32,F33,F34,F35,F36,F
37,F38,F39,F40,F41,F42,F43,F44,F45,F46,F47,F48,F49,F50,F51,F52,F53,F54,F55,F56,F57,F58,F59,F60";
		String strCabecalho2 =
"C,F31,F32,F33,F34,F35,F36,F37,F38,F39,F40,F41,F42,F43,F44,F45,F46,F47,F48,F49,F50,F51,F52,F53,F54,F55,F56,F57,F58,F59,F60";
		String strClasses1 =
"2,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,
10,10,10,10,10,10,10,10,10,10,10,10";
		String strClasses2 = "2,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10";
	*/


/*
	images




		double dPercClass1 = 0.14 ;
		String strCabecalho1 = "C,F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F16,F17,F18";
		String strCabecalho2 = "C,F10,F11,F12,F13,F14,F15,F16,F17,F18";
		String strClasses1 = "2,3,12,2,2,11,6,9,7,15,12,12,15,9,11,12,14,13,9";
		String strClasses2 = "2,12,12,15,9,11,12,14,13,9";
 */


		try{
			Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
		}catch(java.lang.ClassNotFoundException e) {
			System.err.print("ClassNotFoundException: ");
			System.err.println(e.getMessage());
		}

		try{
			Connection con = DriverManager.getConnection(url, "Admin", "Admin");

			Statement stmt = con.createStatement();
			Statement stmt2 = con.createStatement();

			int result = 0 ;

			ResultSet rs = stmt.executeQuery("SELECT ID1, CLASS, DADOS1, DADOS2 , Rnd(ID1) FROM COTRA_MAIN ORDER
BY 5");

			int iNumeroRegistros = 0 ;

			result = stmt2.executeUpdate("DELETE * FROM COTRA_MAIN_AUX");

			// Carrega tabela auxiliar , igual a tabela fonte COTRA_MAIN
			if (rs.next()){

				do {
					String s1 = rs.getString("CLASS");
					String s2 = rs.getString("DADOS1");
					String s3 = rs.getString("DADOS2");
					result = stmt2.executeUpdate("INSERT INTO COTRA_MAIN_AUX (CLASS, DADOS1,DADOS2)
VALUES ('" + s1 + "','" + s2 + "','" + s3 + "')");
					iNumeroRegistros++;

				} while (rs.next()) ;
			}

			// Separa 20% dos registros para teste, movendo-os para COTRA_TEST
```

```
                    // Serao separados registros da classe 1 e classe 2, de forma a manter uma quantidade semelhante

                    Statement stmt3 = con.createStatement();
                    ResultSet rsClasse1 = stmt3.executeQuery("SELECT ID, CLASS, DADOS1, DADOS2, Rnd(ID) FROM
COTRA_MAIN_AUX WHERE CLASS = '1' ORDER BY 5");
                    // ResultSet rsClasse1 = stmt3.executeQuery("SELECT ID, CLASS, DADOS1, DADOS2, Rnd(ID) FROM
COTRA_MAIN_AUX ORDER BY 5");
                    int iNumeroClasse1 = (iNumeroRegistros/10);

                    stmt.executeUpdate("DELETE FROM COTRA_TEST");

                    if (rsClasse1.next()){
                            while (iNumeroClasse1>0){
                                    stmt.executeUpdate("INSERT INTO COTRA_TEST (CLASS, DADOS1,DADOS2) VALUES ('1','" +
rsClasse1.getString("Dados1") + "','" + rsClasse1.getString("Dados2") + "')");
                                    stmt.executeUpdate("DELETE FROM COTRA_MAIN_AUX WHERE ID = " +
rsClasse1.getString("ID"));

                                    rsClasse1.next();
                                    iNumeroClasse1--;
                            }
                    }

                    ResultSet rsClasse2 = stmt3.executeQuery("SELECT ID, CLASS, DADOS1, DADOS2, Rnd(ID) FROM
COTRA_MAIN_AUX WHERE CLASS = '2' ORDER BY 5");
                    /// ResultSet rsClasse2 = stmt3.executeQuery("SELECT ID, CLASS, DADOS1, DADOS2, Rnd(ID) FROM COTRA_MAIN_AUX
ORDER BY 5");
                    int iNumeroClasse2 = (iNumeroRegistros/10);

                    if (rsClasse2.next()){
                            while (iNumeroClasse2>0){
                                    stmt.executeUpdate("INSERT INTO COTRA_TEST (CLASS, DADOS1,DADOS2) VALUES ('2','" +
rsClasse2.getString("Dados1") + "','" + rsClasse2.getString("Dados2") + "')");
                                    stmt.executeUpdate("DELETE FROM COTRA_MAIN_AUX WHERE ID = " +
rsClasse2.getString("ID"));

                                    rsClasse2.next();
                                    iNumeroClasse2--;
                            }
                    }

                    // Prepara as tabelas contendo os registros labeled e unlabeled

                    result = stmt2.executeUpdate("DELETE * FROM COTRA_LABELED");
                    result = stmt2.executeUpdate("DELETE * FROM COTRA_UNLABELED");

                    ResultSet rsLabeled = stmt3.executeQuery("SELECT ID, CLASS, DADOS1, DADOS2, Rnd(ID) FROM
COTRA_MAIN_AUX WHERE CLASS = '1' ORDER BY 5");
                    int contLabeled = (( iNumeroRegistros * 8/10 )* porcLabeled/100);
                    contLabeled = (int)(contLabeled * dPercClass1);

                    int contLabeled1aux = contLabeled;

                    if (rsLabeled.next()){
                            while (contLabeled>0){
                                    String strClass = rsLabeled.getString("CLASS");
                                    String strDados1 = rsLabeled.getString("DADOS1");
                                    String strDados2 = rsLabeled.getString("DADOS2");
                                    stmt.executeUpdate("INSERT INTO COTRA_LABELED (CLASS,DADOS1, DADOS2) VALUES ('" +
strClass + "','" + strDados1 + "','" + strDados2 + "')");
                                    stmt.executeUpdate("DELETE FROM COTRA_MAIN_AUX WHERE ID = " + rsLabeled.getString("ID"));
                                    rsLabeled.next();
                                    contLabeled--;
                            }
                    }

                    ResultSet rsLabeled2 = stmt3.executeQuery("SELECT ID, CLASS, DADOS1, DADOS2, Rnd(ID) FROM
COTRA_MAIN_AUX WHERE CLASS = '2' ORDER BY 5");
                    int contLabeled2 = ( iNumeroRegistros * 8/10 )* porcLabeled/100 ;
                    contLabeled2 = contLabeled2 - contLabeled1aux ;

                    if (rsLabeled2.next()){
                            while (contLabeled2>0){
```

```
                                        String strClass = rsLabeled2.getString("CLASS");
                                        String strDados1 = rsLabeled2.getString("DADOS1");
                                        String strDados2 = rsLabeled2.getString("DADOS2");

                                        stmt.executeUpdate("INSERT INTO COTRA_LABELED (CLASS,DADOS1, DADOS2) VALUES ('" +
strClass + "','" + strDados1 + "','" + strDados2 + "')");
                                        stmt.executeUpdate("DELETE FROM COTRA_MAIN_AUX WHERE ID = " +
rsLabeled2.getString("ID"));
                                        rsLabeled2.next();
                                        contLabeled2--;
                                }
                        }

                        ResultSet rsUnLabeled = stmt3.executeQuery("SELECT ID, CLASS, DADOS1, DADOS2, Rnd(ID) FROM
COTRA_MAIN_AUX ORDER BY 5");
                        int intContUnlabeled = (iNumeroRegistros * 6/10) * (100 - porcLabeled)/100 ;

                        if (rsUnLabeled.next()){
                                while (intContUnlabeled > 0 ) {
                                        String strClass = rsUnLabeled.getString("CLASS");
                                        String strDados1 = rsUnLabeled.getString("DADOS1");
                                        String strDados2 = rsUnLabeled.getString("DADOS2");

                                        stmt.executeUpdate("INSERT INTO COTRA_UNLABELED (CLASS,DADOS1,DADOS2) VALUES
('" + strClass + "','" + strDados1 + "','" + strDados2 + "')");
                                        stmt.executeUpdate("DELETE FROM COTRA_MAIN_AUX WHERE ID = " +
rsUnLabeled.getString("ID"));
                                        rsUnLabeled.next();
                                        intContUnlabeled--;
                                }
                        }

                        // Faz a montagem dos arquivos texto que serao usados

                        try{

                                File writeFile1 = File.createTempFile("tmpTest1", "in", new File("./TESTING/"));
                                PrintWriter prtWriter1 = new PrintWriter(new BufferedWriter(new FileWriter(writeFile1, true)));

                                File writeFile2 = File.createTempFile("tmpTest2", "in", new File("./TESTING/"));
                                PrintWriter prtWriter2 = new PrintWriter(new BufferedWriter(new FileWriter(writeFile2, true)));

                                ResultSet rsArqTest = stmt3.executeQuery("SELECT * FROM COTRA_TEST");

                                prtWriter1.write(strCabecalho1);
                                prtWriter1.write("\n" + strClasses1 + "\n" );

                                prtWriter2.write(strCabecalho2);
                                prtWriter2.write("\n" + strClasses2 + "\n");

                                if (rsArqTest.next()) {
                                        do {
                                                String strClass = rsArqTest.getString("CLASS");
                                                String strDados1 = rsArqTest.getString("Dados1");
                                                String strDados2 = rsArqTest.getString("Dados2");
                                                prtWriter1.write( "\n" + strClass + "," + strDados1 );
                                                prtWriter2.write( "\n" + strClass + "," + strDados2 );
                                        } while (rsArqTest.next());
                                }

                                prtWriter1.close();
                                prtWriter2.close();

                                File unlabeledFile1 = new File("./TESTING/TesteUnlabeled1.in");
                                File unlabeledFile2 = new File("./TESTING/TesteUnlabeled2.in");

                                boolean data1 = unlabeledFile1.delete();
                                boolean data2 = unlabeledFile2.delete();

                                data1 = writeFile1.renameTo(new File("./TESTING/TesteUnlabeled1.in"));
                                data2 = writeFile2.renameTo(new File("./TESTING/TesteUnlabeled2.in"));

                        } catch (IOException e) {
```

```
                System.out.println("Exception when opening write file" + e);
                System.exit(-1);
        }

        // Monta os arquivos Labeled

        try{

                File writeFile1 = File.createTempFile("tmpLabeled1", "in", new File("./TESTING/"));
                PrintWriter prtWriter1 = new PrintWriter(new BufferedWriter(new FileWriter(writeFile1, true)));

                File writeFile2 = File.createTempFile("tmpLabeled2", "in", new File("./TESTING/"));
                PrintWriter prtWriter2 = new PrintWriter(new BufferedWriter(new FileWriter(writeFile2, true)));

                ResultSet rsArqLabeled = stmt3.executeQuery("SELECT * FROM COTRA_LABELED");

                prtWriter1.write(strCabecalho1);
                prtWriter1.write("\n" + strClasses1 + "\n" );

                prtWriter2.write(strCabecalho2);
                prtWriter2.write("\n" + strClasses2 + "\n");

                if (rsArqLabeled.next()) {
                        do {
                                String strClass  = rsArqLabeled.getString("CLASS");
                                String strDados1 = rsArqLabeled.getString("Dados1");
                                String strDados2 = rsArqLabeled.getString("Dados2");
                                prtWriter1.write( "\n" + strClass + "," + strDados1 );
                                prtWriter2.write( "\n" + strClass + "," + strDados2 );
                        } while (rsArqLabeled.next()) ;
                }

                prtWriter1.close();
                prtWriter2.close();

                File unlabeledFile1 = new File("./TESTING/Labeled1.in");
                File unlabeledFile2 = new File("./TESTING/Labeled2.in");

                boolean data1 = unlabeledFile1.delete();
                boolean data2 = unlabeledFile2.delete();

                data1 = writeFile1.renameTo(new File("./TESTING/Labeled1.in"));
                data2 = writeFile2.renameTo(new File("./TESTING/Labeled2.in"));
        } catch (IOException e) {
                System.out.println("Exception when opening write file" + e);
                System.exit(-1);
        }

        try{

                File writeFile1 = File.createTempFile("tmpUnLabeled1", "in", new File("./TESTING/"));
                PrintWriter prtWriter1 = new PrintWriter(new BufferedWriter(new FileWriter(writeFile1, true)));

                File writeFile2 = File.createTempFile("tmpUnLabeled2", "in", new File("./TESTING/"));
                PrintWriter prtWriter2 = new PrintWriter(new BufferedWriter(new FileWriter(writeFile2, true)));

                ResultSet rsArqUnLabeled = stmt3.executeQuery("SELECT * FROM COTRA_UNLABELED");

                prtWriter1.write(strCabecalho1);
                prtWriter1.write("\n" + strClasses1 + "\n" );

                prtWriter2.write(strCabecalho2);
                prtWriter2.write("\n" + strClasses2 + "\n");

                if (rsArqUnLabeled.next()) {
                        do {
                                String strClass = rsArqUnLabeled.getString("CLASS");
                                String strDados1 = rsArqUnLabeled.getString("Dados1");
                                String strDados2 = rsArqUnLabeled.getString("Dados2");
                                prtWriter1.write( "\n" + strClass + "," + strDados1 );
                                prtWriter2.write( "\n" + strClass + "," + strDados2 );
                        } while (rsArqUnLabeled.next());
```

```
                    }

                    prtWriter1.close();
                    prtWriter2.close();

                    File unlabeledFile1 = new File("./TESTING/Unlabeled1.in");
                    File unlabeledFile2 = new File("./TESTING/Unlabeled2.in");

                    boolean data1 = unlabeledFile1.delete();
                    boolean data2 = unlabeledFile2.delete();

                    data1 = writeFile1.renameTo(new File("./TESTING/Unlabeled1.in"));
                    data2 = writeFile2.renameTo(new File("./TESTING/Unlabeled2.in"));

                } catch (IOException e) {
                    System.out.println("Exception when opening write file" + e);
                    System.exit(-1);
                }



                System.out.println("Fim do processo ! ");









            } catch(SQLException ex) {
            System.err.println("SQLException: " + ex.getMessage());
            }

            return true;

        }
    }
```

14.3 TesteCotraining

```
/**
 * Class that tests a given classifier again
 * data.
 * Author: Alan Glezer
 */

package lutil;

import BayesianNetworks.*;

//import java.io.*;

public class TesteCoTraining {
  /**
   * Testing a classifier against data.
   */
  public static int[] test(ReadIn readTest, BayesNet classifier) {
    int i, j, k, aux;
    int numberRecords = 0;
    int numberMisses = 0;
    double logLikelihoodTest = 0.0;

    ProbabilityVariable pvs[] = classifier.get_probability_variables();
    ProbabilityFunction pfs[] = classifier.get_probability_functions();
    int numberLabels = (pvs[0]).number_values();
    int numberPerLabel[] = new int[numberLabels];
    int numberMissesPerLabel[] = new int[numberLabels];
```

```
int numeroClasses = readTest.getNumberOfValues()[0];
double ClasseMaxProbabilidade[][] = new double[numeroClasses][50000];
//int registros[][];

boolean hasRecord = false;

for (aux = 1; readTest.hasNextRecord(); aux++) {

        hasRecord = true;
  int record[] = readTest.readNextRecord();
  //registros[aux] = record;
  // The observed values in record are decremented by one.
  // This is necessary because the data is assumed to
  // start from 1 (and 0 indicates missing values), whereas
  // the usual convention in JavaBayes is that indexes
  // start at 0 --- following the conventions of Java.
  for (i=0; i<record.length; i++)
        record[i]--;

  // Maximize p(C|X1...XN), proportional
  // to p'(C) x p'(X1|C) x ... x p'(XN|C)
  double likelihoodClass[] = new double[numberLabels];
  int correctLabel = record[0];
  for (j=0; j<numberLabels; j++) {
        record[0] = j;
        double prod = (pfs[0]).get_value(j);
        for (i=1; i<pfs.length; i++) {
        prod *= (pfs[i]).evaluate(pvs, record);
        }
        likelihoodClass[j] = prod;
        ClasseMaxProbabilidade[j][aux-1] = likelihoodClass[j];
  }
  record[0] = correctLabel;
  // Find maximum of likelihood.
  int positionMax = 0;
  double likelihoodMax = likelihoodClass[positionMax];
  for (j=1; j<numberLabels; j++) {
        if (likelihoodClass[j] > likelihoodClass[positionMax]) {
        positionMax = j;
        likelihoodMax = likelihoodClass[j];
        }
    }

  //ClasseMaxProbabilidade[j-2][aux-1] = likelihoodMax;
    //ClasseMaxProbabilidade[0][0] = likelihoodMax;

  // Compare with correct label and store relevant information.
  numberRecords++;
  numberPerLabel[record[0]]++;
  if (positionMax != record[0]) {
    numberMisses++;
    numberMissesPerLabel[record[0]]++;
  }
  // Update overall log-likelihood.
  logLikelihoodTest += Math.log(likelihoodClass[record[0]]);

}
int resultado[];
if(hasRecord){
    resultado = new int[numeroClasses];;
    for(i = 0; i<resultado.length;i++){
        double likelihoodMax = 0.0;
        for(aux = 0; aux < ClasseMaxProbabilidade[i].length; aux++){
            boolean isAlreadyPicked = alreadyHasRecord(resultado, aux, i);
            if(ClasseMaxProbabilidade[i][aux] > likelihoodMax && !isAlreadyPicked){
                resultado[i] = aux;
                likelihoodMax = ClasseMaxProbabilidade[i][aux];
            }
        }
    }
}
else
    resultado = null;
```

```
/*
      StringBuffer sb = new StringBuffer("\n//Test " + numberRecords + " " +
                                                 numberMisses + " " +
                                                 (((double)numberMisses)/((double)numberRecords)));
      for (k=0; k<numberPerLabel.length; k++)
        sb.append(numberPerLabel[k] + " ");
      for (k=0; k<numberMissesPerLabel.length; k++)
        sb.append(numberMissesPerLabel[k] + " ");
      for (k=0; k<numberMissesPerLabel.length; k++)
        sb.append( (((double)numberMissesPerLabel[k])/
              ((double)numberPerLabel[k])) + " ");
      sb.append(logLikelihoodTest + " " +
                              (logLikelihoodTest/((double)numberRecords)) + "\n");
*/    return(resultado);
    }
    private static boolean alreadyHasRecord(int result[], int auxiliar, int classe){
      for(int i = 0; i < classe; i++)
                              for(int k = 0; k < result.length; k++)
                              if(result[k]==auxiliar)
                                      return true;
      return false;
    }
}
```