Resumo

Aqui começa o resumo escrito em língua vernácula.

Abstract

Here the abstract written in foreign language begins

1 Introdução

Segundo o dicionário Cambridge, a palavra *rank* significa uma posição particular, mais alta ou mais baixa que outras. Por exemplo, a seleção brasileira de futebol masculino ocupa uma posição na classificação das seleções da FIFA.

A posição ocupada pela seleção brasileira somada as posições ocupadas pelas outras seleções constituem o *ranking* das seleções de futebol masculino da FIFA. Pode-se citar como outros exemplos de *ranking*: o *ranking* do IDH (Índice de Desenvolvimento Humano) dos países, o *ranking* da competição da Yahoo! [1] sobre *Learning to rank*, entre outros.

Todos os *rankings* citados são ordenados de acordo com um critério. Por exemplo, para a definição do *ranking* de seleções da FIFA, calcula-se o total de pontos para cada seleção vinculada à FIFA e ordenam-se as seleções com base nas pontuações. As seleções com maior pontuação ocupam o topo do *ranking*, enquanto as seleções com menor pontuação ocupam a base.

A pontuação total de uma seleção em um período de quatro anos é definida por meio da soma da média de pontos ganhos em partidas disputados nos 12 últimos meses e da média de pontos ganhos em partidas que ocorreram há mais de 12 meses, a qual se deprecia anualmente. Há uma fórmula matemática para definir a pontuação de uma equipe em uma partida.

Podemos afirmar que as fórmulas utilizadas pela FIFA para definir as pontuações das seleções constituem um modelo que determina uma ordem parcial do conjunto de seleções FIFA. Dizer que o as fórmulas utilizadas pela FIFA constituem um modelo caracteriza a tarefa de ordenação das seleções FIFA como computável, ou seja, o modelo pode ser escrito em forma de algoritmo e avaliado, junto aos dados necessários, por um computador.

Outra característica do modelo FIFA é que ele foi definido por funcionários da instituição, ou seja, o modelo é fruto do intelecto humano. Apesar disso, seria possível a aprendizagem automática de um modelo que ordenasse o conjunto de seleções vinculadas

[1] http://learningtorankchallenge.yahoo.com/leaderboard.php

à FIFA.

As características de computabilidade e possibilidade de aprendizagem automática, observadas no exemplo do *ranking* da FIFA, podem ser encontradas em diversos outros tipos de ordenações.

Talvez não haja muita valia em automatizar o aprendizado de um modelo para ordenar as seleções vinculadas à FIFA; o modelo definido pela FIFA já é bastante adequado. Porém, para uma grande quantidade de aplicações, aprender modelos que ordenem uma base de dados pode ser útil, ajudando a lidar com problemas como quantidade de dados e variáveis a serem consideradas.

Um problema beneficiado pela aprendizagem de um modelo de ordenação seria: Dado um conjunto de documentos em que cada documento pode receber um rótulo, atribuir uma posicao a cada documento do conjunto, tendo como insumo pares documento-rótulo ou uma relação de ordem total ou parcial entre os documentos.

O problema de ordenação de documentos é um dos problemas de interesse de uma área conhecida como *Learning to Rank* que tem atraído a atenção de muitos pesquisadores nos últimos anos. Já conta com sites especializados no assunto[2], competições[3] além do apoio das gigantes de TI.

Essa área de pesquisa está inserida no contexto de *inteligência artificial* e compartilha conhecimentos com áreas como *Aprendizado de máquina*, *Recuperação de informação* e *processamento de linguagem natural*.

Estudaremos a implantação e avaliação de uma técnica de aprendizado para ordenação de um conjunto no qual os elementos são passíveis de rotulação. Tal técnica reduz o problema de ordenação a uma problema de classificação de acordo com o descrito em [[**?**]].

Essa técnica, que havia sido especificada apenas teoricamente, foi implantada, avaliada e comparada com outras técnicas. Essa é a principal contribuição desse estudo.

Para a implantação do algoritmo de *ranking* foi escolhida a ferramenta *WEKA*[4], descrita no livro [[**?**]]. Essa ferramenta fornece vários recursos para *aprendizado de máquina*.

O restante dessa monografia está organizado da seguinte forma: O capítulo 2 faz uma introdução aos conceitos necessários para o entendimento do algoritmo *Ranking*; o

[2]http://research.microsoft.com/en-us/um/beijing/projects/letor/
[3]http://learningtorankchallenge.yahoo.com/
[4]http://www.cs.waikato.ac.nz/ml/weka/

capítulo 3 mostra os problemas e soluções que derivaram da implantação do *Ranking*; o capítulo 4 descreve a estrategia usada para avaliar o algoritmo e os resultados obtidos; e o capítulo 5 apresenta as principais descobertas obtidas nesse estudo.

2 Ranking: *Noções Preliminares*

Na área de *aprendizado de máquina*, classificação é tarefa de atribuir rótulos a cada elemento de um dado conjunto tendo como entrada pares elemento-rótulo. Comparativamente, *ranking* é a tarefa de atribuir posições a cada elemento de um dado conjunto tendo como entrada pares elemento-rótulo, uma relação de ordem parcial entre os elementos, ou uma relação de ordem total entre os elementos [**?**].

Embora desempenhem tarefas diferentes com saídas diferentes, algoritmos de classificação e de *ranking* podem compartilhar o mesmo tipo de entrada: pares elemento-rótulo. Isso é uma evidência de que se pode usar um algoritmo de classificação para compor um algoritmo de *ranking*.

De fato, a técnica de *ranking* descrita em [**?**] propõe envolver um algoritmo de classificação com etapas de pré-processamento e pós-processamento de forma que o produto final seja uma ordenação. Nesse estudo, essa técnica é chamada de *ranking reduzido a classificação*.

2.1 Introdução ao problema

Dado um conjunto composto por elementos aos quais é possível atribuir um rótulo de valor 0 ou 1, deseja-se encontrar uma permutação dos elementos de maneira que os elementos que apresentem maior chance de receber o rótulo 0 devem preceder os com maior chance de receber o rótulo 1. Cada elemento é composto por um conjunto de características e a chance de um elemento receber o rótulo 0 ou o rótulo 1 deve ser calculada com base nessas características.

Substituindo-se a palavra rótulo pela palavra classe no enunciado acima, percebe-se que o problema de ordenação pode ser tratado como um problema de classificação. Mais especificamente, um problema de classificação binária, pois os valores de classe possíveis são 0 e 1.

Um algoritmo de aprendizado para classificação recebe como entrada um conjunto em que cada elemento possui uma classe assinalada e, partindo dessa entrada, deve aprender um classificador capaz atribuir uma classe a qualquer elemento. A etapa em que o aprendizado ocorre é chamada de treinamento.

O treinamento de um classificador é um aprendizado supervisionado, pois cada elemento submetido à etapa de treinamento está vinculado à sua classe verdadeira. Dessa forma, pode-se medir a qualidade do classificador aprendido usando elementos com classes conhecidas, porém suprimindo-as quando forem os elementos forem submetidos a classificação.

O produto da classificação de um elemento é a previsão da classe à qual aquele elemento pertence. Essa previsão é composta pelo valor da classe e pela probabilidade do elemento pertencer à classe prevista. Nesso caso, por se tratar de um problema de classificação binária, se a chance de um elemento pertencer à classe 0 é p, a chance dele pertencer à classe 1 é $1 - p$.

Supondo que se submeta um conjunto a um classificador, de posse das previsões feitas para todas os elementos do conjunto, é possível ordenar tais elementos, de forma decrescente, de acordo com a probabilidade de cada instância pertencer à classe 0. De fato, essa é uma possível solução para o problema de ordenação.

Segundo [**?**], esse método para ordenação dos elementos pode resultar em um erro teórico máximo muito alto, dependendo da performance do classificador. A técnica de *ranking reduzido a classificação*, proposta no mesmo artigo, reduz o erro teórico máximo envolvendo o treinamento do classificador com uma etapa de pré-processamento e a avaliação com uma etapa de pós-processamento.

Na técnica de *ranking reduzido a classificação*, o treinamento do classificador binário ocorre após uma etapa inicial em que se aplica uma transformação nos exemplos da base de treinamento. Após o treinamento, a definição da ordem das instâncias na base de avaliação é feita por um algoritmo chamado *torneio*, que usa as previsões do classificador treinado.

2.2 Definições

Como dito anteriormente, o treinamento e avaliação de um classificador é um processo que manipula conjuntos compostos por elementos. Definições mais rigorosas são

necessárias para a descrição da implantação da técnica de *ranking reduzido a classificação* no capítulo 3. As definições abaixo foram feitas levando-se em consideração o funcionamento da ferramenta WEKA e os requisitos necessários para se modelar a solução de *ranking reduzido a classificação* exposta em [**?**].

- Uma base B é um conjunto de instâncias com cardinalidade n.

$$B = \{i_1, ..., i_n\} \qquad |B| = n$$

- Uma instância I é uma tupla A, C na qual A é um conjunto de atributos com cardinalidade m e C é a classe da instância e pode valer 0 ou 1.

$$I = (A, C) \qquad A = \{a_1, ..., a_m\} \qquad C = x \mid x \in \{0, 1\}$$

Não será necessária a definição do conceito de atributo. Uma vez que os atributos são manipulados pelo classificador e o funcionamento dos classificadores nas soluções aqui propostas é uma caixa preta, os atributos são transparentes para os propósitos desse trabalho.

panorama	temperatura	humidade	ventoso	adequado para jogo
ensolarado	quente	alta	falso	não
ensolarado	quente	alta	verdadeiro	não
nublado	quente	alta	falso	sim
chuvoso	branda	alta	falso	sim
chuvoso	frio	normal	falso	sim
chuvoso	frio	normal	verdadeiro	não
nublado	frio	normal	verdadeiro	sim
ensolarado	branda	alta	falso	não
ensolarado	frio	normal	falso	sim
chuvoso	branda	normal	falso	sim
ensolarado	branda	normal	verdadeiro	sim
nublado	branda	alta	verdadeiro	sim
nublado	quente	normal	falso	sim
chuvoso	branda	alta	verdadeiro	não

Tabela 1: Base de dados de tempo

A tabela **??** aprensenta um exemplo de base extraido do livro [**?**]. Essa base indica se há condições para a prática de um esporte de acordo com medições meteorológicas. Um exemplo dessa base apresenta um conjunto de medições meteorológicas e, de acordo com essas medições, a classe do exemplo pode ser *sim* ou *não*.

Nessa tabela, a primeira linha nomeia os quatro atributos da base e a classe (adequado para jogo), as linhas seguintes representam as instâncias da base. Nota-se que a classe recebe valores *sim* e *não*, apesar de serem valores diferentes de 0 e 1, como dito na definição, ainda se trata de uma base com classe binária.

Escrevendo a primeira instância da base na notação definida acima, tem-se: (ensolarado, quente, alta, falso, não). O primeiro elemento da tupla é o conjunto de atributos e o segundo elemento é a classe da instância.

2.3 Medidas de desempenho

Geralmente, a medida de eficiência mais utilizada para classificação é a acurácia: uma razão entre o número de instâncias corretamente classificadas sobre o número total de instâncias no conjunto de avaliação.

O erro decorrente de uma classificação afeta a acurácia de maneira linear. Como, para ordenações, a quantidade de acertos não é tão relevante quanto a posição das instâncias ordenadas, propõe-se outro tipo de medida para avaliação do *ranking* aprendido.

Uma medida comum de avaliação para algoritmos de *ranking* é a área sobre a curva *ROC (Receiver Operating Characteristic)*, comumente chamada de *AUC (Area Under the Curve)*.

A perda, $1 - AUC$, associada a essa medida é calculada pelo número de instâncias, normalizado pela quantidade de 0s vezes a quantidade de 1s, que necessitam ser trocadas para um *ranking* perfeito.

Uma ordenação é perfeita, quando todas as instâncias com classe 0 precedem as com classe 1, nesse caso a perda na AUC é 0. No pior caso, em que todos os 1s precedem os 0s, a perda na AUC é 1.

Comparativamente, um erro de classificação pode ter maior influência na medida AUC que na acurácia afetando consideravelmente um *ranking*, mas não a classificação. A causa disso é a AUC considerar a relação entre as instâncias ordenadas, enquanto a acurácia considera apenas erros e acertos pontualmente. Abaixo ilustramos através de um exemplo uma relação entre essas medidas que comprova o intuído sobre erros na classificação.

No exemplo acima, temos quatorze instâncias ordenadas em um *ranking* com os atributos, as classes e as previsões dadas por um classificador. Podemos perceber que o classificador errou apenas a classe da primeira instância.

panorama	temperatura	humidade	ventoso	classe	previsão
ensolarado	quente	alta	falso	não	sim
nublado	quente	alta	falso	sim	sim
chuvoso	branda	alta	falso	sim	sim
chuvoso	frio	normal	falso	sim	sim
nublado	frio	normal	verdadeiro	sim	sim
ensolarado	frio	normal	falso	sim	sim
chuvoso	branda	normal	falso	sim	sim
ensolarado	branda	normal	verdadeiro	sim	sim
nublado	branda	alta	verdadeiro	sim	sim
nublado	quente	normal	falso	sim	sim
ensolarado	quente	alta	verdadeiro	não	não
chuvoso	frio	normal	verdadeiro	não	não
ensolarado	branda	alta	falso	não	não
chuvoso	branda	alta	verdadeiro	não	não

Tabela 2: Exemplo de *ranking* e classificação na base weather

Calculando a acurácia, temos treze acertos em quatorze possíveis, o que equivale a aproximadamente 93% de acerto.

Calculando a perda da AUC considerando como base a classe *sim*, a primeira instância precisa retroceder nove posições para uma ordenação perfeita, normalizando pelo número de *não*s vezes o número de *sim*s, temos $(1 - AUC) = 9 \div (5 * 9) = 0,2$, logo a AUC vale 80%. Esse exemplo comprova que a AUC sofre um impacto maior devido a erros de classificação se comparada à acurácia.

De acordo com [**?**], um classificador que gere um erro de ordem α na acurácia pode gerar um erro teórico máximo de $\alpha \cdot n$ na AUC, onde n é a cardinalidade do conjunto de instâncias avaliado. Enquanto para o mesmo classificador, a técnica de *ranking reduzido a classificação* apresenta um erro teórico máximo de $\alpha \cdot 2$.

O erro na AUC se intensifica a medida que o desbalanceamento de classes do conjunto usado no treinamento aumenta pois, quanto mais desbalanceadas as classes, mais provável que o classificador resultante seja tendencioso para a classe majoritária.

3 Ranking: *Implantação*

O algoritmo proposto em langford08 é composto de duas etapas: uma de treinamento e outra de ordenação que gera o *ranking*. Porém, o custo computacional desse algoritmo é alto em ambas etapas. Com bases de dados extensas o desempenho do algoritmo degrada consideravelmente, isso motivou a busca de alternativas para reduzir o tempo tanto de treinamento quanto de ordenação.

Duas abordagens foram consideradas para reduzir a complexidade do algoritmo: uma para a fase de treinamento e outra para a fase de geração do *ranking*. Para a fase de classificação, foi pensada uma estratégia de votação entre classificadores e para a geração do *ranking*, foi pensada uma estratégia de torneio baseada em *quicksort*. Nesse capítulo, estão explicados tanto o algoritmo original quanto essas abordagens para reduzir a complexidade.

3.1 Otimização do treinamento: Amostragem e Votação

O algoritmo original chama a etapa de treinamento de AUC-Train. Nessa etapa, são feitas combinações entre as instâncias de classe 0 e de classe 1, como mostrado no *algoritmo 1*.

Algorithm 1 AUC-Train

Let $S' = \{\langle (x_1, x_2), 1 \cdot (y_1 < y_2) \rangle : (x_1, y_1), (x_2, y_2) \in S \wedge y_1 \neq y_2\}$
return $c = A(S')$

O AUC-Train faz a combinação de cada instância com classe 0 com todas as instâncias de classe 1 nas formas (z, u) e (u, z), onde z é a instância com classe 0 e u é a instância com classe 1.

Para o pior caso, com um conjunto de treinamento em que metade das instâncias é da classe 0 e a outra metade é da classe 1, a complexidade assintótica é de $O(n^2)$.

Esse algoritmo de mesclagem das instâncias é executado uma vez e seu resultado é usado como massa de dados para um algoritmo de aprendizado; no caso específico dessa implantação, um classificador.

O custo total do AUC-Train é composto pelos custos da etapa de mesclagem e do algoritmo de aprendizagem somados. Vale ressaltar que a etapa de mesclagem tem um tempo maior de execução; a otimização endereça esse problema especificamente.

Na estratégia adotada para reduzir o tempo de mesclagem figuram duas técnicas conhecidas em *data mining*, a primeira é efetuar uma amostragem do conjunto original e a segunda é promover uma votação entre vários classificadores treinados com as amostragens como massa de dados.

Algorithm 2 AUC-Train com amostragem

Let $S_0 = \{\langle (x, y)\rangle : (x, y) \in S \wedge y = 0\}$
Let $S_1 = S - S_0$
$V \leftarrow \emptyset$
for $i = 1,\ i \rightarrow iterations$ **do**
  Let $S_{sample} = S' : S' \subset S_1 \wedge |S'| = limit$
  Let $S_{train} = \{\langle ((x_0,\ x_s),\ 1),\ ((x_s,\ x_0),\ 0)\rangle : (x_0,\ y_0) \in S_0 \wedge (x_s,\ y_s) \in S_{sample}\}$
  $c \leftarrow A(S_{train})$
  $V \leftarrow V \bigcup \{c\}$
end for

O produto gerado pelo algoritmo 2 é um conjunto de classificadores treinados a partir de amostragens do conjunto original. Se passarmos os parâmetros $limit = all$ e $iterations = 1$, ele gera o mesmo resultado do algortimo descrito por langford.

3.2 Otimização da ordenação: Quicksort

A etapa de ordenação original é chamada de *torneio*. Nessa abordagem, todas as instâncias do conjunto a ser ordenado são comparadas entre si com base no classificador obtido no treinamento e recebem uma pontuação. As instâncias de maior pontuação assumem as primeiras posições no *ranking*. A proposta de otimização baseia-se em uma adaptação do algoritmo de quicksort para ordenar as instâncias do *ranking*.

Algorithm 3 Degree

for all $x \in U$ **do**
  $deg(x) = |\{x' : c(x, x') = 1, x' \in U\}|$
end for
Ordene U de forma descendente com base em $deg(x)$, resolva os empates arbitrariamente

3.3 Definições

Algumas definições necessárias para o funcionamento do algoritmo final.

3.4 Algoritmo final

Algorithm 4 Algoritmo final do *Ranking*

$c \leftarrow train(TS, LA, pairs, iterations)$
$rank(RS, c, quicksort)$

4 *Avaliação do* Ranking

De acordo com o artigo Langford, o algoritmo de Ranking proposto tem um desempenho melhor quando aplicado sobre bases com alto desbalanceamento entre as classes. Partindo desse princípio, foi montada uma estratégia de testes com cinco bases de dados com diferentes níveis de desbalanceamento. Tais bases podem ser encontradas no repositório da University of California, Irvine (UCI) em *http://archive.ics.uci.edu/ml/*.

4.1 Características das bases

As bases usadas foram as seguintes: Breast Cancer; Statlog (Vehicle Silhouettes), chamada de Vehicle; Hepatitis; Glass Identification, chamada de Glass e Yeast. Como o algoritmo proposto em langford tem um custo computacional alto, $O(n^2)$ para treinamento e $O(n^2)$ para gerar o ranking; isso aliado ao baixo controle sobre o ambiente de execução culminou na escolha de bases pequenas para avaliação do experimento.

As bases Vehicle, Glass e Yeast tratam, originalmente, de problemas multi classe. A essas bases foram aplicadas transformações a fim de torná-las problemas de classe binária. Para as bases Vehicle e Glass, foram unidas todas as classes, exceto a minoritária, em uma nova classe. Para a base Yeast, apenas as classes de valor 'CYT' ou 'POX' foram consideradas.

Cada base utilizada apresenta diferentes características como: desbalanceamento entre as classes, número de atributos, número de instâncias, entre outros. A tabela 3 mostra um resumo sobre as características das bases.

4.2 Execução e Avaliação do Algoritmo

Foram escolhidos quatro classificadores base para avaliar o algoritmo: J48, Naïve Bayes, Logistic e SMO. O motivo da escolha desses classificadores é o reconhecimento

Bases	Atributos		Instâncias	Classe		
	Contínuos	Discretos		Minoritária	Majoritária	Distribuição
breast-cancer	0	9	286	85	201	30% - 70%
vehicle	18	0	846	199	647	23% - 77%
hepatitis	6	13	155	32	123	20% - 80%
glass	9	0	214	29	185	13% - 87%
yeast	8	0	483	20	463	4% - 96%

Tabela 3: Dados sobre as bases usadas para *ranking*

de tais como padrões em suas famílias, J48 para árvores (implanta a árvore de decisão C4.5), Naïve Bayes para estatísticos, Logistic (implanta curva logística) e SMO (implanta Support Vector Machine) para classificadores baseados em funções.

Para cada classificador base houve quatro baterias de execução com diferentes configurações:

1. Somente o classificador;
2. O classificador como base para o algoritmo de *ranking* original;
3. O classificador como base para o algoritmo de ranking com configurações de 1 par por instância e 10 classificadores na votação;
4. O classificador como base para o algoritmo de ranking com configurações de 10 pares por instância e 1 classificador na votação.

Nas três baterias que envolveram o algoritmo de *ranking*, o método de ordenação usado para gerar o resultado final foi o *torneio*, como explicado no artigo langford. Em todas as baterias foi executada uma validação cruzada com 10 partições e os resultados apresentados nas tabelas é a média das $AUCs$ obtidas para cada partição.

Para as tabelas de número 4, 5, 6 e 7; o símbolo * significa ranking aplicado com 1 par por instância e 10 classificadores na votação. O símbolo ** significa ranking aplicado com 10 pares por instância e 1 classificador na votação.

Bases	J48	J48 acrescido de		
		Ranking Original	Ranking*	Ranking**
breast-cancer	**0,62806 (0,01005)**	0,46784 (0,03049)	0,51289 (0,01950)	0,45055 (0,02984)
vehicle	0,93725 (0,00056)	0,91389 (0,00624)	**0.98072 (0.00017)**	0,95670 (0,00071)
hepatitis	0,69655 (0,04084)	0,67112 (0,03127)	**0,74322 (0,03925)**	0,72179 (0,04571)
glass	**0,90322 (0,01578)**	0,83772 (0,03524)	0,89016 (0,02934)	0,88860 (0,02668)
yeast	0,48918 (0,00021)	**0,95009 (0,01453)**	0,78550 (0,03630)	0.84806 (0.01924)

Tabela 4: Desempenho para árvore de decisão C4.5

Bases	Naïve Bayes	Naïve Bayes acrescido de		
		Ranking Original	Ranking*	Ranking**
breast-cancer	**0,71543 (0,01899)**	0,20857 (0,00620)	0,04976 (0,00445)	0,04532 (0,00426)
vehicle	**0,80898 (0,00468)**	0,24323 (0,01559)	0,00310 (0,00004)	0,12514 (0,01546)
hepatitis	**0,85919 (0,01190)**	0,22489 (0,05188)	0,06052 (0,00265)	0.06608 (0,00101)
glass	**0,94084 (0,01099)**	0,17271 (0,02416)	0,02222 (0,00151)	0,02476 (0,00132)
yeast	0,82863 (0.06355)	0,84269 (0,03505)	**1,00000 (0,00000)**	**1,00000 (0,00000)**

Tabela 5: Desempenho para Naïve Bayes

4.2.1 Desempenho para *C4.5* (trees.J48)

4.2.2 Desempenho para *Naïve Bayes* (bayes.NaiveBayes)

4.2.3 Desempenho para a *Curva Logística* (functions.Logistic)

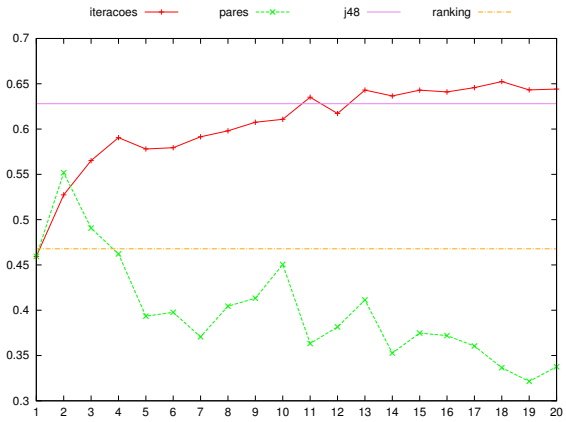
4.2.4 Desempenho para *Support Vector Machine* (functions.SMO)

Bases	Logistic	Logistic acrescido de		
		Ranking Original	Ranking*	Ranking**
breast-cancer	0,66625 (0,02322)	**0,66740 (0,02364)**	0,65784 (0,02143)	0,65132 (0,02219)
vehicle	0,99358 (0,00003)	**0,99420 (0,00003)**	0,99358 (0,00003)	0,99234 (0,00002)
hepatitis	**0,80924 (0,02598)**	0,79882 (0,02432)	0,75064 (0,04278)	0,74557 (0,05075)
glass	0,95965 (0,00308)	**0,97037 (0,00192)**	0,96101 (0,00484)	0,95536 (0,00384)
yeast	0,85805 (0,04115)	0,83453 (0,04931)	**0,87414 (0,03463)**	0,85691 (0,03997)

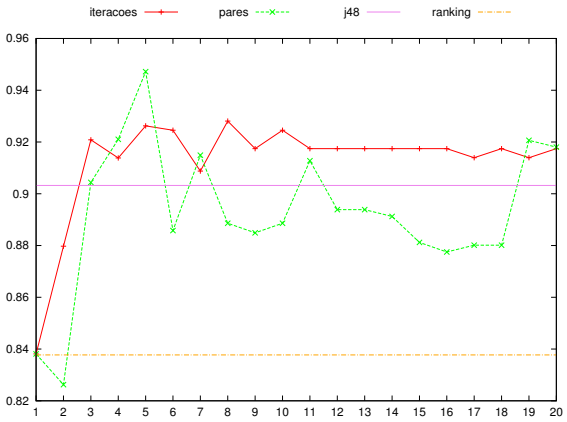
Tabela 6: Desempenho para Logistic

Bases	SMO	SMO acrescido de		
		Ranking Original	Ranking*	Ranking**
breast-cancer	0,59272 (0,00696)	**0,66670 (0,02294)**	0,65832 (0,02119)	0,65563 (0,02003)
vehicle	0,94434 (0,00169)	**0,99651 (0,00001)**	0,99396 (0,00002)	0,99380 (0,00002)
hepatitis	0,75128 (0,01618)	**0,81522 (0,01914)**	0,80759 (0,01899)	0,78189 (0,02640)
glass	0,89181 (0,00679)	**0,95712 (0,00175)**	0,93402 (0,00610)	0,93752 (0,00537)
yeast	0,77391 (0,04799)	0,83555 (0,04831)	**0,99891 (0,00001)**	**0,99891 (0,00001)**

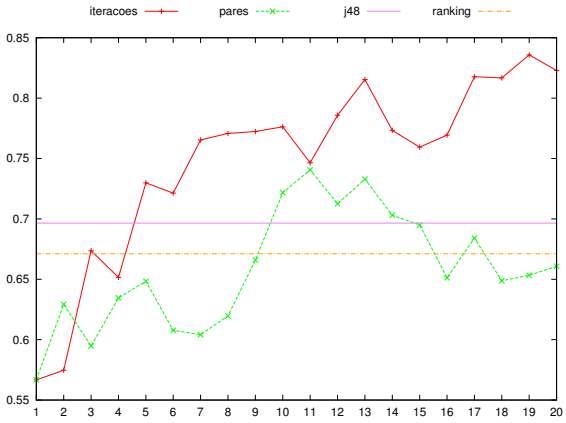
Tabela 7: Desempenho para SMO



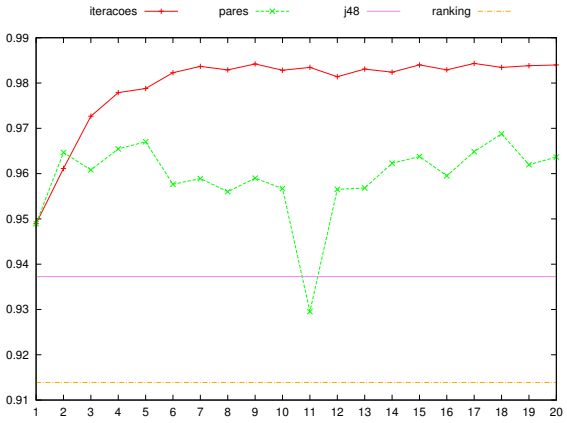
(a) Breast cancer



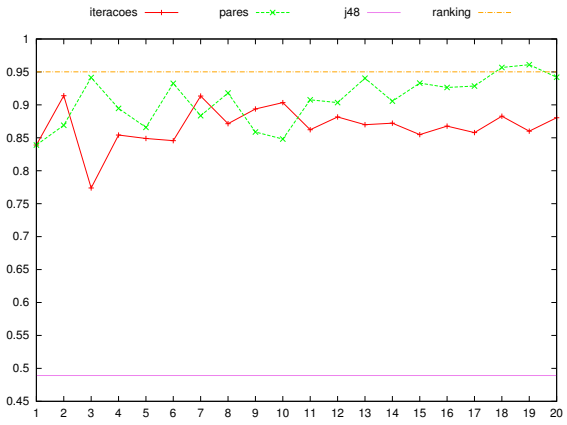
(b) Glass



(c) Hepatitis

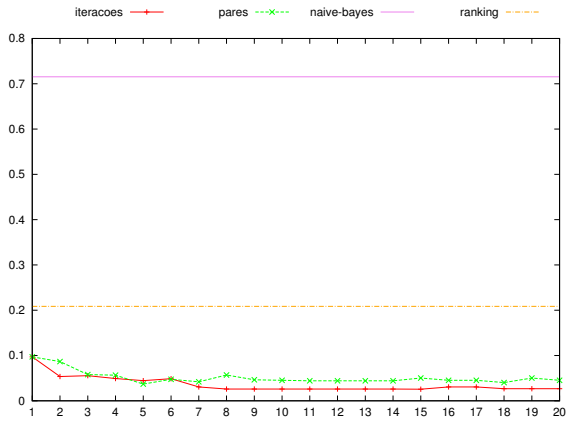


(d) Vehicle

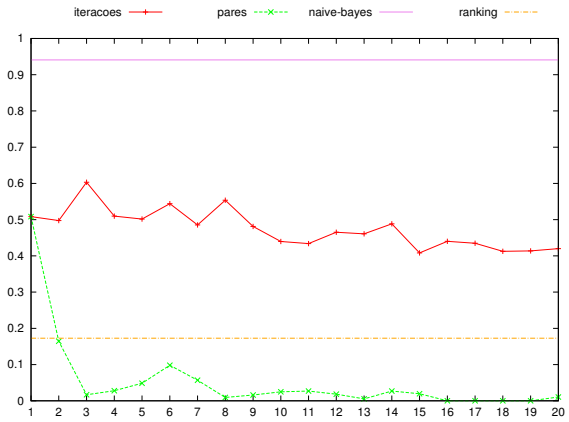


(e) Yeast

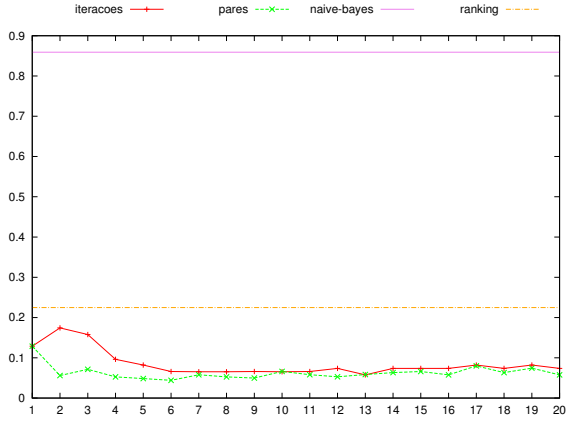
Figura 1: Gráficos de desempenho para árvore de decisão C4.5



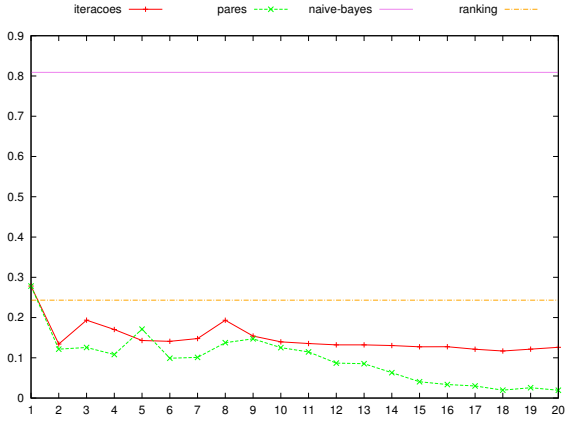
(a) Breast cancer



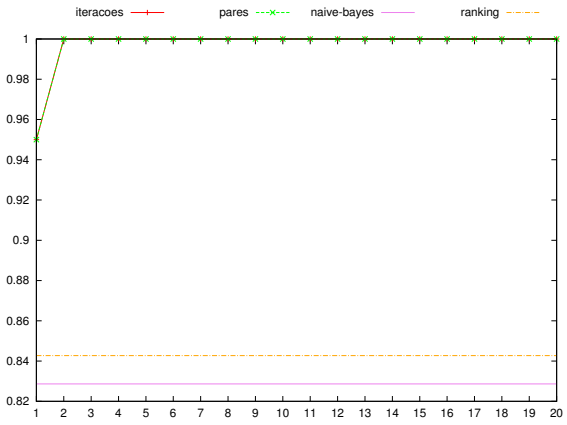
(b) Glass



(c) Hepatitis

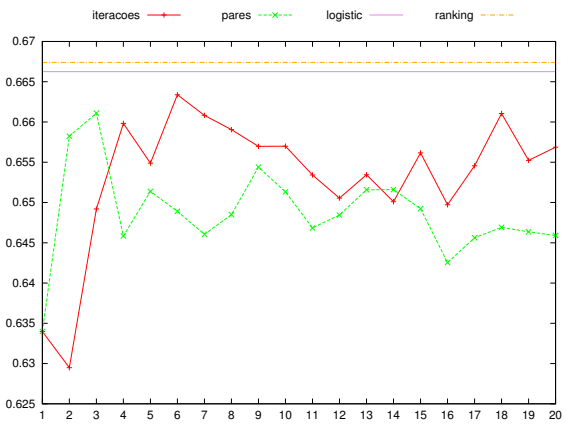


(d) Vehicle

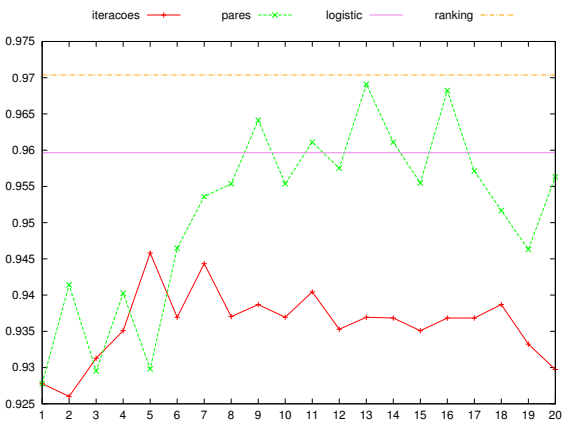


(e) Yeast

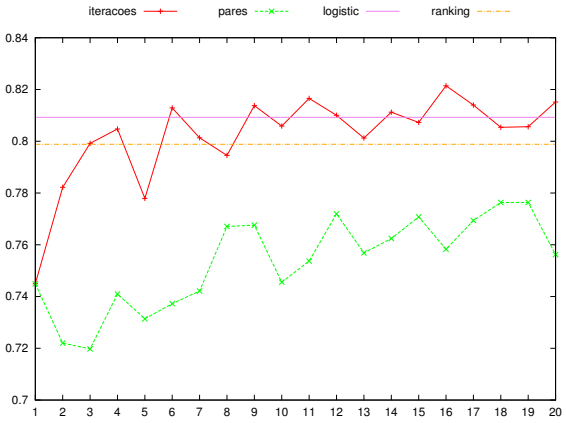
Figura 2: Gráficos de desempenho para Naïve Bayes



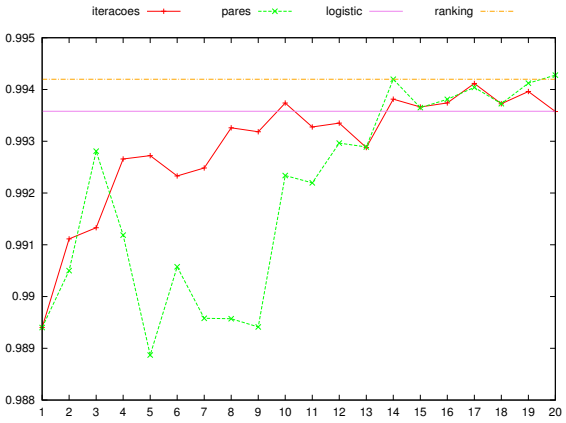
(a) Breast cancer



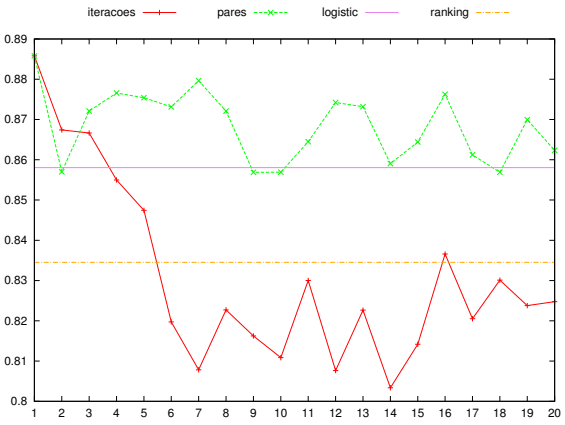
(b) Glass



(c) Hepatitis

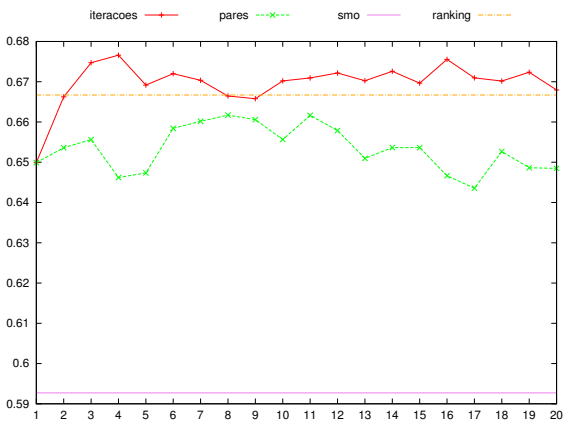


(d) Vehicle

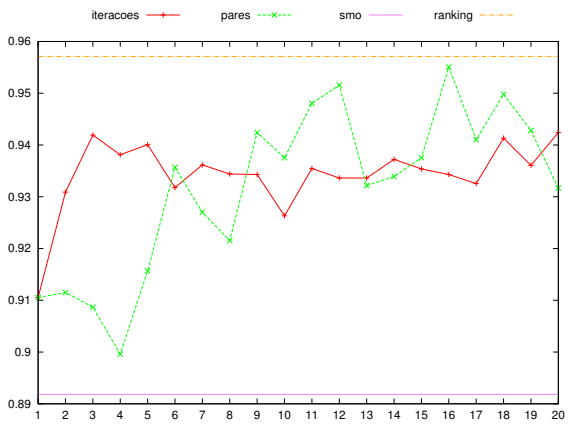


(e) Yeast

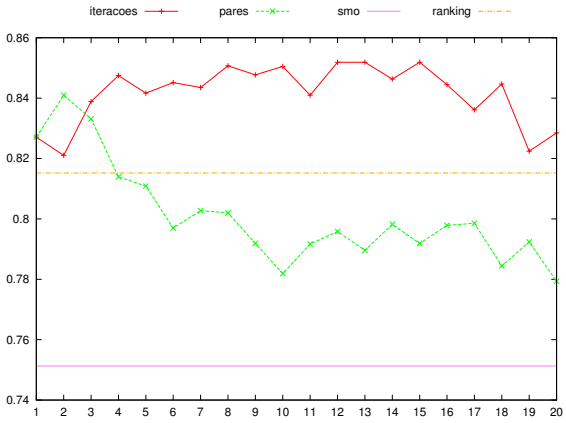
Figura 3: Gráficos de desempenho para Curva Logística



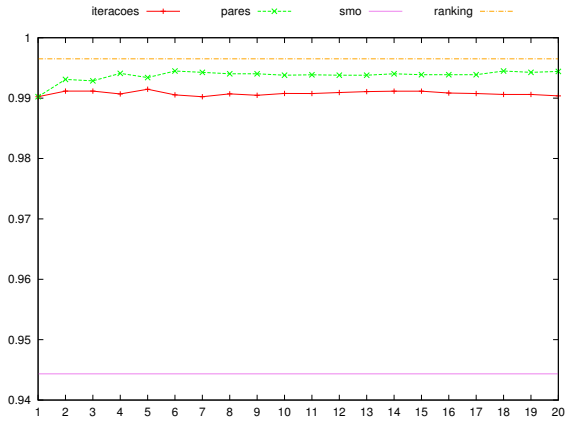
(a) Breast cancer



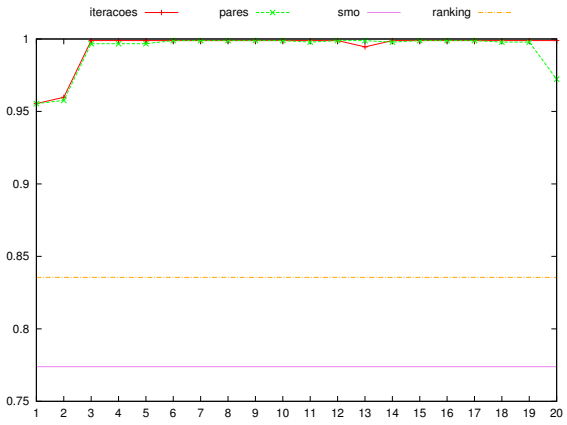
(b) Glass



(c) Hepatitis



(d) Vehicle



(e) Yeast

Figura 4: Gráficos de desempenho para Curva Logística

5 Conclusão

A primeira conclusão que pode ser tirada é que o algoritmo testado aqui tem uma performance muito baixa no que diz respeito a tempo computacional, tanto para treinamento, quanto para avaliação. As estratégias para melhoria no tempo de treinamento como uso de votação e de um número reduzido de pares por instância ajudaram nesse aspecto e criaram resultados por vezes superiores ao algoritmo original e ao classificador base.

Os esforços para otimizar o tempo na etapa de avaliação consistiram em implantar um algoritmo baseado no algoritmo *Quicksort*, que teria tempo de execução médio $n \cdot \log n$, um avanço comparado ao tempo de execução médio do torneio que é n^2. Embora esta estratégia tenha sido completamente implantada, não houve nenhum experimento que a utilizasse.

O classificador Naïve Bayes teve um desempenho incomum quando combinado com o algortimo de *Ranking*. Esse classificador gerou resultados muito abaixo do esperado para todas as bases testadas, exceto para a base *yeast*.

Pode-se reparar que, para a base *glass*, os resultados começam a melhorar em relação aos obtidos para as bases *breast-cancer*, *vehicle* e *hepatitis*. Já para a base *yeast*, o Naïve Bayes aliado ao algoritmo de *Ranking* teve uma performance perfeita acertando quase todos as ordenações. Esse comportamento bipolar não foi elucidado nesse estudo.

Comparando as estratégias implantandas para acelerar a etapa de treinamento de forma isolada, pode-se chegar a seguinte conclusão: o aumento do número de classificadores na votação cria resultados mais com menor varição na AUC que o aumento de pares por instância no treinamento.

Olhando as curvas geradas para cada uma dessas estratégias nos gráficos do capítulo AVALIACAO, percebe-se que a tendência para a estratégia de aumento de classificadores na votação é, na maioria dos casos, crescente. Enquanto a tendência para a estratégia de aumento do número de pares por instância não pode ser definida com clareza em alguns

casos.

O artigo [[**?**]] mostra que um classificador que produza um erro r na classificação pode produzir um erro teórico máximo de $n \cdot r$, onde n é o número de exemplos a serem ordenados, na ordenação. Mostra também que o *Ranking* gera produz um erro teórico máximo de $2 \cdot n$.

Não foi possível verificar a validade dessa prova teórica. Na maioria dos casos, a ordenação derivada apenas do classificador teve erro inferior ao erro na ordenação via algoritmo *Ranking*.