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* das aplicações do Rails Rumble e o *ranking* do IDH (Índice de Desenvolvimento Humano) dos países.

Todos os *rankings* citados são ordenados de acordo com um critério. Para a definição da posição de uma seleção no *ranking* de seleções da FIFA, há um cálculo que leva em conta os alguns fatores das partidas disputadas por essa seleção nos quatro anos mais recentes. Exemplos:

- Pontos por resultado (vitória, empate ou derrota);
- Importância da partida (amistoso a copa do mundo);
- Força do adversário (posição no *ranking*).
- Força da região (confederação FIFA)
- Período (Mais recente, maior peso)

Com o cálculo feito para cada seleção vinculada à FIFA, define-se o *ranking* das seleções. Pode-se considerar que o topo do *ranking* possui as melhores seleções em um passado recente, enquanto na base estão as piores seleções.

Em termos gerais, *ranking* pode ser enunciado como a tarefa de ordenar os elementos de um conjunto de acordo com algum critério.

Todos os exemplos citados acima compartilham um aspecto comum: a ordem dos elementos do ranking é feita sobre uma pontuação de cada elemento obtida a partir de

algumas características. Da mesma forma, o problema a ser estudado consiste em ordenar elementos de acordo com uma pontuação.

Dado um conjunto de instâncias, espera-se obter um arranjo desse que considere alguma característica de tais instâncias.

Pode-se descrever o problema como a busca pela ordenação dos elementos de um conjunto. A ordem é regida por uma pontuação obtida por cada elemento, assim podemos reduzir o problema a determinação da pontuação para todos os elementos do conjunto.

2 Ranking: *Noções Preliminares*

Cada instância possui uma característica binária C em comum, isso significa que C pode assumir um de dois valores possíveis para uma dada instância. Fixando um valor V para a característica C como base, podemos dizer, de forma simplória, que a pontuação dessa instância é a probabilidade de C possuir o valor V.

2.1 Introdução ao problema

Algumas definições são necessárias para a descrição do problema feita acima. Começando pelo conceito de *ranking*, o conjunto a ser ordenado é chamado de base. Os elementos do conjunto a serem ordenados são denominados instâncias. As características dos elementos a que nos referimos são chamadas de atributos das instâncias. Dentre os atributos, um é nomeado classe, encarregado da ordenação no *ranking*, além disso, para nosso problema, o atributo classe deve ter um domínio binário $\{0, 1\}$.

Tendo como entrada uma base composta por instâncias para as quais os valores das classes são desconhecidos, espera-se como saída uma ordenação de tais instâncias que siga o critério: as instâncias com maior chance de pertencer à classe 0 devem preceder as com maior chance de pertencer à classe 1. Isso deve ser concluído com base apenas nos atributos das instâncias.

A via para definir uma solução passa pelo aprendizado de máquina. O algoritmo projetado funciona em duas etapas. Na primeira, denominada treinamento, recebe uma base com instâncias compostas por atributos e classe e tenta extrair algum conhecimento dessas. Na segunda etapa, o algoritmo recebe uma base com instâncias compostas apenas por atributos e deve usar o conhecimento obtido na primeira etapa para ordenar tais instâncias.

Uma medida comum de sucesso 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.

O uso da classe como critério para ordenação sugere uma relação com os problemas de classificação e regressão em aprendizagem de máquina. Realmente, pode-se derivar uma ordenação diretamente das pontuações obtidas em uma classificação ou regressão. Se desejamos que instâncias da classe 0 ocupem as primeiras posições do *ranking*, basta ordenar de forma crescente as pontuações obtidas considerando a classe 0.

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 uma forma pontual.

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

atributos	classe	previsão
$a_{0,1}, \cdots, a_{0,n}$	0	0
$a_{1,1}, \cdots, a_{1,n}$	0	0
$a_{2,1}, \cdots, a_{2,n}$	1	0
$a_{3,1}, \cdots, a_{3,n}$	0	0
$a_{4,1}, \cdots, a_{4,n}$	0	0
$a_{5,1}, \cdots, a_{5,n}$	0	0
$a_{6,1}, \cdots, a_{6,n}$	0	0
$a_{7,1}, \cdots, a_{7,n}$	0	0
$a_{8,1}, \cdots, a_{8,n}$	0	0
$a_{9,1}, \cdots, a_{9,n}$	1	1

Tabela 1: Exemplo de *ranking* e classificação

No conjunto hipotético representado ao lado, temos dez 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 terceira instância.

Calculando a acurácia, temos nove acertos em dez possíveis, ou seja, 90%. Calculando a perda da AUC considerando como base a classe 0, a terceira instância precisa retroceder

seis posições para uma ordenação perfeita, normalizando pelo número de 0s vezes o número de 1s, temos $(1 - AUC) = 6 \div (2 * 8) = 0,325$, logo a AUC vale $62,5\%$. Isso comprova que a AUC, comparada à acurácia, pode sofrer um impacto maior devido a erros de classificação.

Langford explica que 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. Esse efeito se intensifica a medida que aumenta o desbalanceamento de classes do conjunto usado durante o processo de treinamento pois, quanto mais desbalanceadas as classes, mais provável que o classificador resultante seja tendencioso para a classe majoritária.

2.2 Formulação Matemática

A formulação matemática a seguir tem como objetivo possibilitar a descrição do algoritmo de *Ranking* proposto em Langford.

Definition 1. *Uma base B é composta por um espaço de n atributos $A \in a_0, ..., a_n$*

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 2 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 2: 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 3, 4, 5 e 6; 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 3: 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 4: 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)

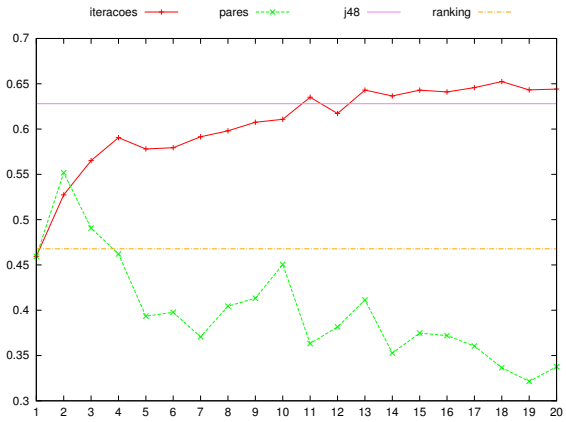
4.2.4 Desempenho para *Support Vector Machine* (functions.SMO)

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)

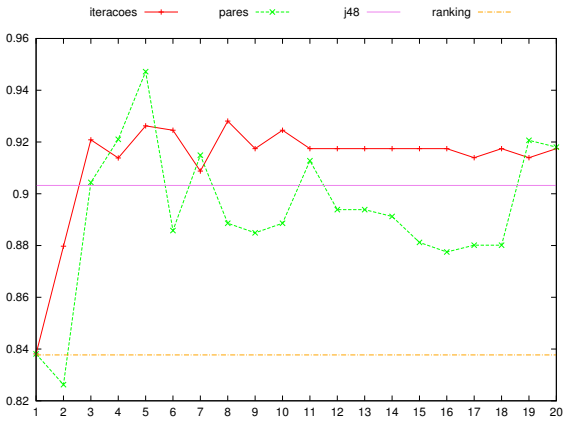
Tabela 5: Desempenho para Logistic

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)**

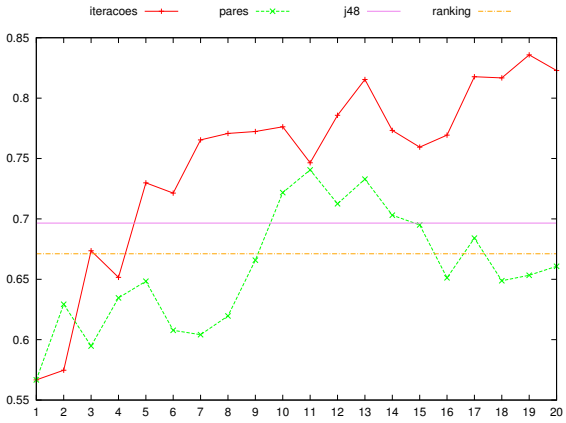
Tabela 6: Desempenho para SMO



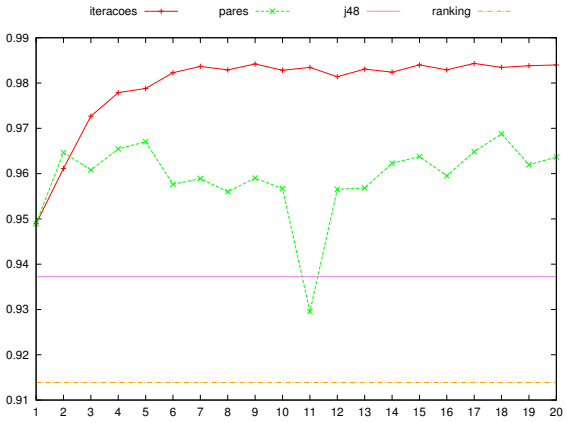
(a) Breast cancer



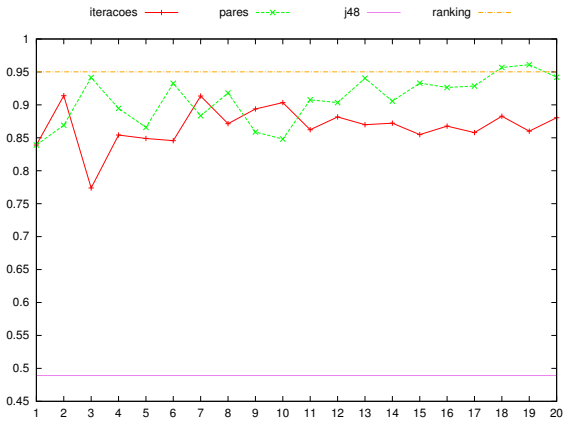
(b) Glass



(c) Hepatitis

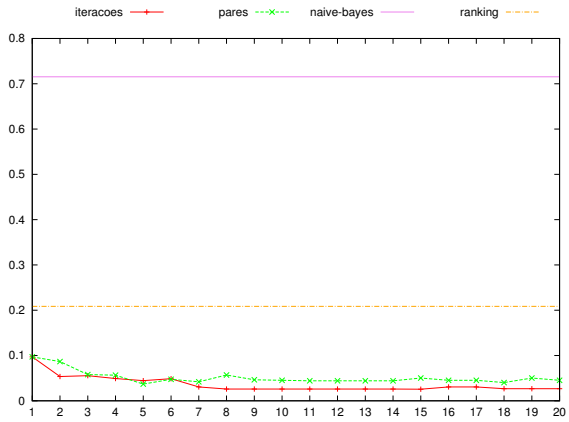


(d) Vehicle

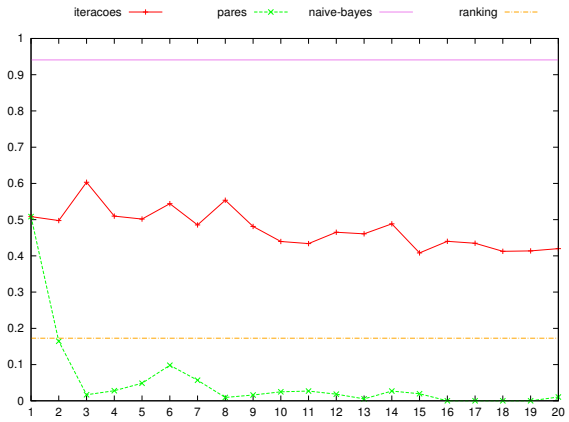


(e) Yeast

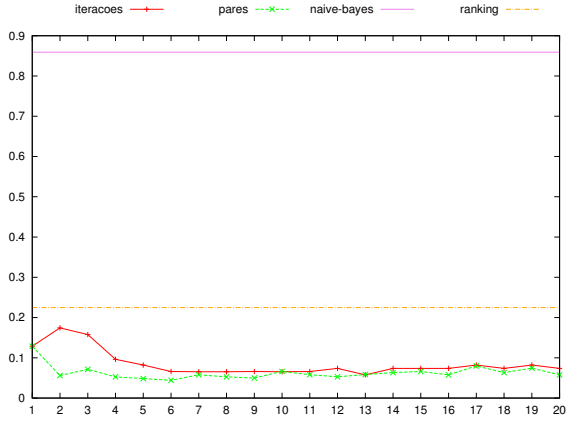
Figura 1: Gráficos de desempenho para árvore de decisão C4.5



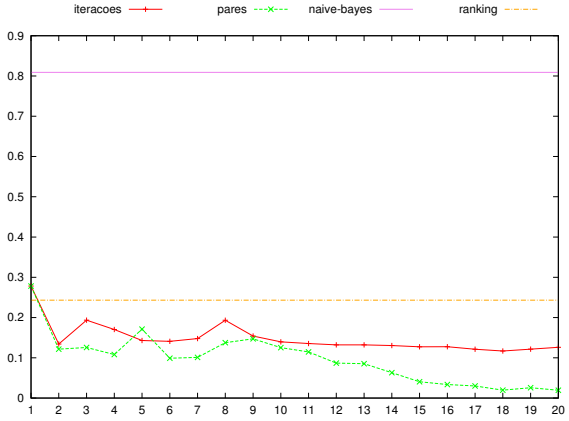
(a) Breast cancer



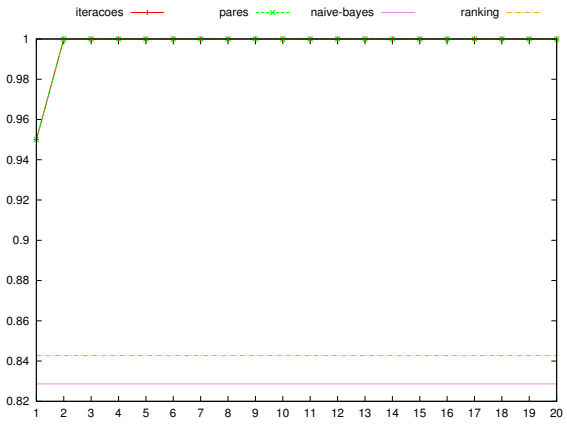
(b) Glass



(c) Hepatitis

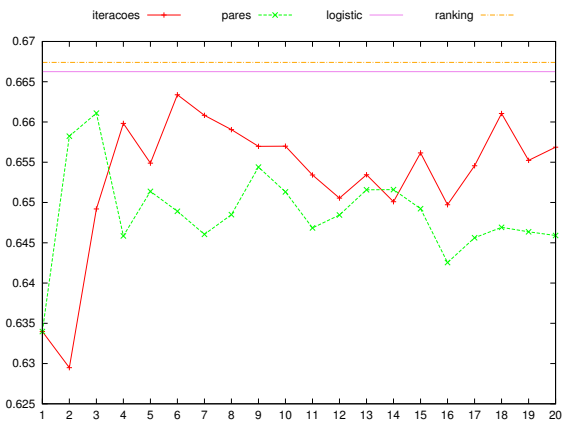


(d) Vehicle

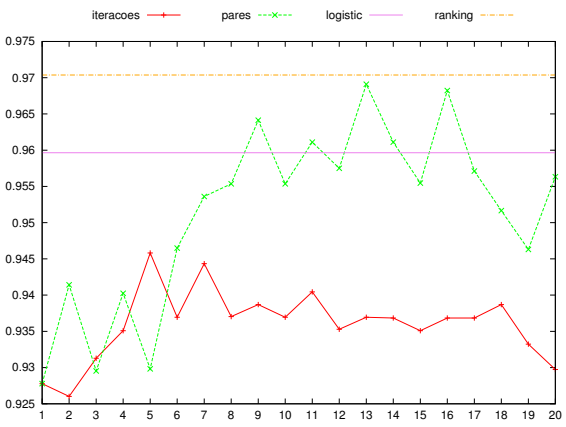


(e) Yeast

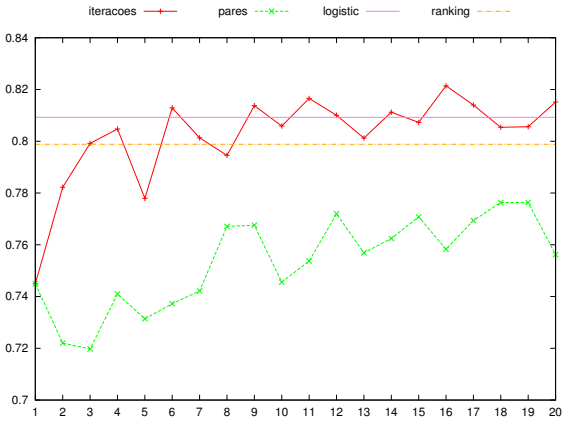
Figura 2: Gráficos de desempenho para Naïve Bayes



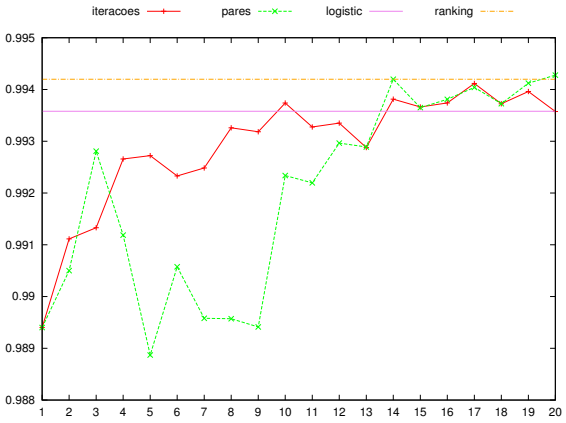
(a) Breast cancer



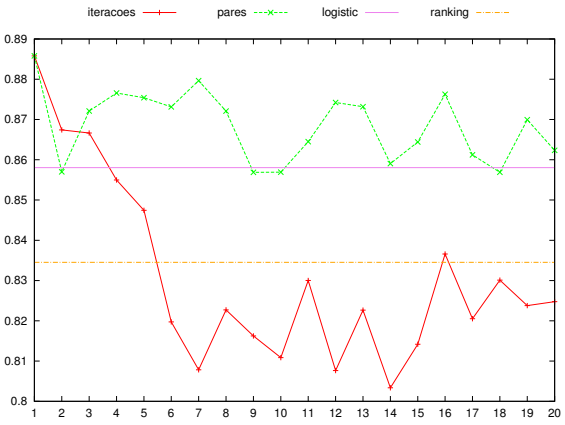
(b) Glass



(c) Hepatitis

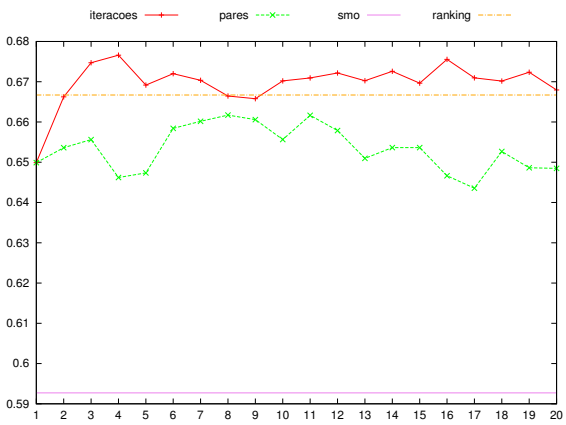


(d) Vehicle

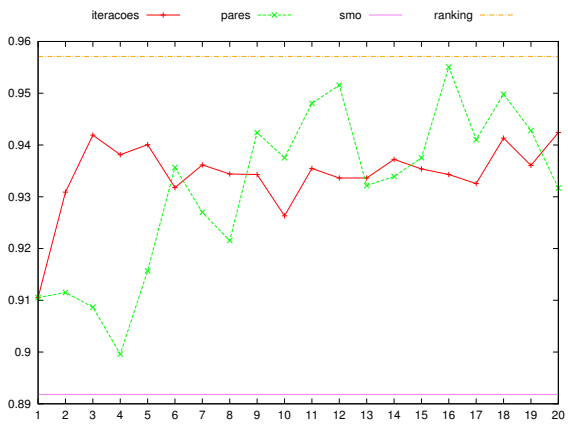


(e) Yeast

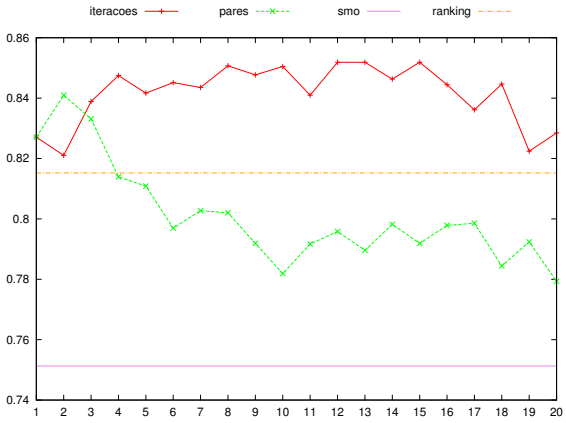
Figura 3: Gráficos de desempenho para Curva Logística



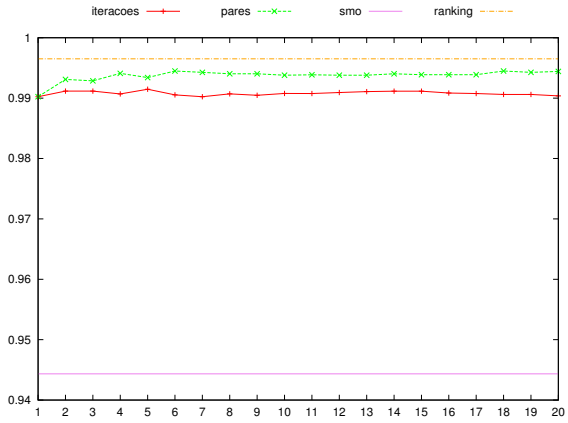
(a) Breast cancer



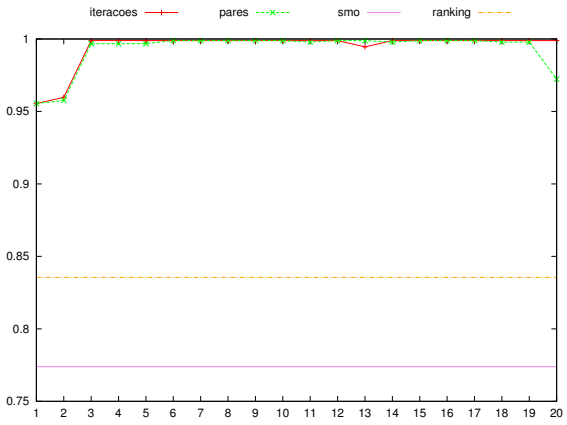
(b) Glass



(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 tentativas de melhora 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 utilizou.

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 estáveis 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.