Alocação de Clientes em Grupos Usando Classificação via Boosting : Uma Comparação com os Métodos Tradicionais de Classificação

Alexandre Rübesam *, Ronaldo Dias †

16 de julho de 2004

Depto. de Estatística - IMECC - Universidade Estadual de Campinas

*rubesam@ime.unicamp.br, financiado pela FAPESP
†dias@ime.unicamp.br

Resumo

Os métodos estatísticos tradicionamente usados em classificação são, basicamente, regressão logística e análise discriminante. Outros métodos que recentemente aparecem nas aplicações são árvores de classificação e redes neurais. Os métodos tradicionais apresentam baixo custo computacional, mas são pouco flexíveis, enquanto os métodos como redes neurais são muito flexíveis, mas caros e com menor interpretabilidade. A metodologia apresentada, *Boosting*, é flexível, fácil de aplicar e tem um custo computacional baixo. *Boosting* funciona combinando seqüencialmente classificadores simples, dando maior peso em cada passo às observações classificadas incorretamente no passo anterior.

Este trabalho apresenta a metodologia *boosting*, e mostra dois exemplos de sua aplicação : uma em dados simulados, e outra em dados reais. A aplicação em dados reais consiste em alocar clientes de uma loja de varejo em grupos que definem os perfis destes clientes. Os resultados obtidos com *boosting* nas duas aplicações são comparados aos resultados dos métodos tradicionais.

Abstract

The traditional statistical methods used in classification are, basically, logistic regression and discriminant analysis. Recently, other methods have been used in applications, e.g., classification trees and neural networks. The traditional methods are computationally cheap, but lack flexibility, while methods such as neural networks are flexible, but computationally expensive and complicated. The method presented in this work, boosting, combines simple base classifiers, increasing at each step the weights of the observations that were previously misclassified.

This work presents the method of boosting, as well as two applications. One is on simulated data, and the other on real data. The real-data application consists of classifying clients of a retail store into previously defined groups, which define the clients' profiles. The results obtained with boosting in both applications are compared to the results of the traditional methods.

1 Introdução

Os métodos de classificação (ou discriminação) mais conhecidos e mais utilizados em estatística aplicada são, provavelmente, regressão logística [Agresti 1996, Hosmer e Lemeshow 1989] e análise discriminante [Johnson e Wichern 2002]. Esses métodos, cujos fundamentos estão estabelecidos na teoria estatística há bastante tempo, são lineares, têm baixo custo computacional, e em geral produzem resultados razoáveis, apesar de não raro serem utilizados em situações que contradizem suas suposições paramétricas distribucionais. As propriedades estatísticas destes métodos são conhecidas há muito tempo, e suas interpretações são claras.

Mais recentemente, surgiram métodos do tipo árvores de classificação e regressão (CART) [Breiman 1984], um método altamente interpretável, e cujo uso está bastante disseminado. Este método gera regras de decisão que podem ser interpretadas diretamente em termos das variáveis explanatórias. O desempenho deste método, entretanto, é baixo quando a fronteira de decisão ótima não pode ser aproximada por um conjunto de hiper-retângulos.

O aumento da capacidade computacional também tornou populares os chamados métodos de "inteligência artifical", como as redes neurais. Esses métodos têm, em geral, desempenho superior em relação aos métodos citados acima, especialmente quando a fronteira de decisão ótima é não linear mas com um custo computacional bem mais alto. Além disso, os algoritmos de otimização popularmente utilizados são sensíveis aos ajustes específicos que devem ser feitos (características dos algoritmos de otimização numérica, e.g. métodos de descida de gradiente, métodos de Newton), e é preciso atenção no processo de estimação para evitar o problema de *overfitting*. Outra questão problemática é a escolha da topologia da rede (número de camadas intermediárias e número de unidades em cada camada). Métodos recentes procuram

tratar a questão da generalização ou do desempenho da rede através do controle automático dos parâmetros da rede. A interpretação do modelo pode ser realizada através do uso de algoritmos de extração de regras ou da simplificação do mesmo.

Um método de classificação moderno que têm apresentado bons resultados é o método conhecido como *boosting*. Este método, que surgiu na área de computação [Schapire 1990], atinge um desempenho muito bom nas aplicações testadas, com um custo computacional relativamente baixo, mas sofre do problema de interpretação comum aos métodos modernos. O algoritmo de *boosting* funciona aplicando-se sequencialmente um algoritmo de classificação qualquer (chamado de base) a versões iterativamente reponderadas do conjunto de dados de treinamento. Em cada iteração, as observações classificadas incorretamente na iteração anterior recebem um peso maior. A saída final combina os classificadores construídos em cada iteração, produzindo um comitê de classificação. O desempenho deste comitê é perto de ótimo na maioria dos casos, e a questão de *overfitting* pode ser facilmente controlada. [Friedman, Hastie e Tibshirani 2000] mostraram que o algoritmo AdaBoost, um algoritmo de *boosting* apresentado neste trabalho, é um algoritmo que ajusta aproximadamente um modelo logístico aditivo, no qual o número de iterações é o número de funções usadas na representação aditiva, ou seja, o número de funções somadas para aproximar a função estimada, o que torna o assunto atraente de um ponto de vista estatístico. Mais recentemente, foi mostrado que os vários algoritmos de *boosting* existentes pertencem a uma classe de algoritmos que minimizam um funcional de custo suave através de algum método numérico [Mason 1999], e a atenção da comunidade estatística voltou-se para características como consistência do método, analisada em uma variedade de artigos [Breiman 2000, Jiang 2000a, Jiang 2000b, Lugosi e Vayatis 2004].

O objetivo deste trabalho é apresentar a metodologia *boosting*, pouco conhecida

na área de estatística, especialmente no Brasil, e mostrar que esta ferramenta possui desempenho melhor, quando comparada às metodologias tradicionais, com um custo computacional relativamente baixo. As aplicações apresentadas visam mostrar que o método é de fácil implementação, mesmo em tarefas muito complexas, onde a quantidade de informação é extremamente grande, assim como o número de classes.

O trabalho está dividido da seguinte maneira. A seção 2 apresenta o problema de classificação. A seção 3 apresenta os algoritmos AdaBoost e LogitBoost para o caso de classificação em duas classes e em J classes. A seção 4 apresenta os resultados obtidos com simulação de dados, e a seção 5, os resultados com dados reais. A seção 6 contém considerações finais e conclusão.

2 Classificação

Um procedimento de classificação, regra de classificação ou classificador é algum método que (possivelmente de maneira automática) classifique objetos em classes. Em geral, um procedimento de classificação é construído com base na experiência passada, e o interesse é utilizá-lo para classificar novos objetos.

Para definir uma tarefa de classificação, considere que temos uma amostra de treinamento $L = \{\mathbf{x}_i, y_i\}_{i=1}^N$, onde $\mathbf{x}_i = (x_{i,1}, ..., x_{i,p})$ é um vetor p-dimensional que contém medições feitas no indivíduo i, y_i é o rótulo de classe do indivíduo i e N é o tamanho da amostra. Considere que o número possível de classes é J, e que estas estão contidas em um conjunto $\mathcal{C} = \{1, 2, \ldots, J\}$. Além disso, o espaço de todos os possíveis valores de $\mathbf{x}$ é designado por $\mathcal{X}$.

A tarefa de classificação consiste em obter um regra $F : \mathcal{X} \to \mathcal{C}$ a partir da amostra L, tal que para cada $\mathbf{x} \in \mathcal{X}$, $F(\mathbf{x})$ designa uma classe em $\mathcal{C}$. É desejável que o classificador F tenha erro pequeno não somente sobre a amostra L, mas

em todo o espaço $\mathcal{X}$. Se definirmos uma medida de probabilidade P em $\mathcal{X} \times \mathcal{C}$, podemos definir a medida de erro por $pce^{nc} = P(F(\mathbf{X}) \neq Y)$, a probabilidade não condicional de classificação incorreta (ou taxa de erro), onde Y é uma v.a. que assume valores em $\mathcal{C}$. Obviamente, $pce^{nc} \in [0, 1]$. Este erro pode ser estimado de algumas maneiras : usando o próprio conjunto de treinamento, usando um conjunto de dados independente, chamado de conjunto de teste e ainda usando validação cruzada.

No primeiro caso, a estimativa da taxa de erro para o classificador F, denotada $\hat{pce}^{nc}(F)$, é dada abaixo :

$$\hat{pce}^{nc}(F) = \frac{1}{N} \sum_{i=1}^{N} I(F(\mathbf{x}_i) \neq y_i), \tag{1}$$

onde $I(\cdot)$ denota a função indicadora do evento dentro dos parênteses, isto é, vale 1 se o argumento é verdadeiro, e 0 se for falso. Esta estimativa é chamada de estimativa por resubstituição [Breiman 1984]. Como ela usa os mesmos dados que foram usados para estimar o classificador, em geral a estimativa é otimista. Intuitivamente, é razoável que nos dados de treinamento o desempenho seja ligeiramente melhor do que em dados que não foram usados na estimação, uma vez que o classificador é construído de maneira a minimizar algum critério relacionado à taxa de erro usando os valores específicos da amostra de treinamento.

Uma maneira de se obter uma estimativa mais honesta da taxa de erro é a estimação por amostra de teste. Esta consiste em dividir o conjunto de dados de treinamento $L = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ em duas amostras independentes, amostras L_1, de tamanho N_1, e L_2, de tamanho N_2. Usamos L_1 para construir o classificador F (ou seja, L_1 é agora a amostra de treinamento) e o testamos na amostra L_2, que funciona agora como amostra de teste (*test sample*). A estimativa de pce^{nc} é então

$$\hat{pce}^{nc}_{teste}(F) = \frac{1}{N_2} \sum_{i=1}^{N_2} I(F(\mathbf{x}_i) \neq y_i). \tag{2}$$

Uma divisão sugerida heuristicamente é usar 2/3 dos dados para estimação, e 1/3 para teste. O problema com esta estimativa é que, se o conjunto de dados não for muito grande, perde-se informação que poderia ser utilizada na estimação do classificador.

A estimação da taxa de erro por validação cruzada consiste em dividir o conjunto original de tamanho N em V partes, ou seja, temos $L_1, L_2, \ldots, L_V$, cada um com tamanho N_v. Para cada parte v, construa o classificador $F^{(v)}$ usando as $v-1$ partes e teste na restante. Então a estimativa de $pce^{\ nc}$ usando a parte v pode ser calculada por 2, substituindo N_2 por N_v, $v = 1, 2, \ldots, V$ e fazendo a soma sobre os casos em N_v. Então a estimativa de $pce^{\ nc}$ por validação cruzada é dada por

$$pce_{VC}^{nc}(F) = \frac{1}{V}\sum_{v=1}^{V} pce_{teste}^{nc}(F^{(v)}). \tag{3}$$

A comparação do método apresentado como os métodos tradicionais será feita tendo como medida as estimativas das taxas de erro usando amostra teste.

2.1 A Regra de Bayes

A regra de Bayes é o melhor classificador construível, no sentido de minimizar a perda (5) definida abaixo. Esta regra só pode ser contruída quando se sabe a distribuição dos dados (o que não ocorre na prática), mas além de dar contribuição teórica no desenvolvimento de classificadores, ela é útil na comparação de classificadores em dados simulados, por fornecer uma base de referência, com uma taxa de erro que é a princípio a menor atingível [Ripley 1996].

Denotemos por $\mathbf{X}$ o vetor aleatório das variáveis medidas em cada objeto e por Y a variável aleatória que assume valores em $\mathcal{C}$, ou seja, a classe a que o objeto pertence. Vamos assumir que o vetor $\mathbf{X} \in \Re^p$, ou seja, assumimos que as variáveis são todas contínuas.

Sejam: $p(\mathbf{x}|k)$ a densidade de $\mathbf{X}$ dado $Y = k$

$\pi_k = P(Y = k)$

$p(k|\mathbf{x}) = P(Y = k|\mathbf{X} = \mathbf{x})$ a *posteriori* da classe k dado $\mathbf{X} = \mathbf{x}$.

O objetivo é obter um classificador $\hat{F} : \mathcal{X} \to \mathcal{C}$.

Se $p(\mathbf{x}|k)$ e π_k são conhecidos, então $p(k|\mathbf{x})$ pode ser obtido através da fórmula de Bayes:

$$p(k|\mathbf{x}) = \frac{\pi_k p(\mathbf{x}|k)}{\sum_{j=1}^{J} \pi_j p(\mathbf{x}|j)}. \tag{4}$$

Seja $L(k, l)$ a perda que se tem por tomar a decisão l, quando a classe verdadeira é k. Se os erros de classificação são homogêneos nas classes, isto é, se uma decisão errada é igualmente ruim, independente de em qual classe o objeto foi erroneamente alocado, a seguinte função perda pode ser utilizada :

$$L(k, l) = \begin{cases} 0 & \text{se } l = k \text{ (decisão correta)} \\ 1 & \text{se } l \neq k \text{ (decisão errada)} \end{cases}. \tag{5}$$

Quando $p(\mathbf{x}|k)$ e π_k são conhecidos, o melhor classificador, considerando a perda (5), é dado na proposição abaixo:

Proposição 2.1 *A regra de classificação que minimiza o risco de Bayes sob a perda (5) é*

$$F_B(\mathbf{X}) = k \;\; se \;\; p(k|\mathbf{x}) = \max_{j=1,2,\ldots,J} p(j|\mathbf{x}). \tag{6}$$

A regra de Bayes diz que se deve alocar um objeto na classe com maior probabilidade *a posteriori*. Se duas classes atingem o mesmo valor de $p(k|\mathbf{x})$, o objeto pode ser alocado em qualquer uma delas arbitrariamente.

Para a aplicação com dados simulados (seção 4), comparamos o desempenho dos algoritmos de *boosting* com os métodos tradicionais, e ainda ao desempenho do classificador de Bayes, que pode ser construído neste caso específico.

3 Boosting

O método conhecido como *boosting* nasceu na área de computação (de maneira geral, uma comunidade conhecida pelo nome *machine learning* (aprendizado de máquina, numa tradução literal). Dentro dessa comunidade, foi proposto um problema teórico chamado de problema de *boosting*, que pode ser informalmente exposto da seguinte maneira: "Suponha que existe um método de classificação que é ligeiramente melhor do que uma escolha aleatória, para qualquer distribuição em $\mathcal{X}$. Esse método é chamado de *weak learner*, ou classificador fraco. A existência de um classificador fraco implica na existência de um classificador forte (*strong learner*), com erro pequeno sobre todo o espaço $\mathcal{X}$?"

Em Estatística, isso equivale a perguntar se, dado um método razoável de estimação, é possível obter um método próximo de ótimo.

Este problema foi resolvido por [Schapire 1990], que apresentou um algoritmo que transformava um classificador fraco em um forte. A partir de então, foram desenvolvidos vários algoritmos dentro do contexto de *boosting*. Um dos mais recentes e bem sucedidos deles é o algoritmo conhecido como AdaBoost, apresentado em [Freund e Schapire 1997]. Este nome vem de ***Ada****ptative* ***Boost****ing*, e é oriundo do fato de que AdaBoost gera em cada passo (de forma determinística, mas adaptativa) uma distribuição sobre as observações da amostra, dando maior peso (maior probabilidade de estar na amostra perturbada) às observações classificadas incorretamente no passo anterior.

[Breiman 1998] chamou os algoritmos do tipo *boosting*, em particular do tipo apresentado por [Freund e Schapire 1997], de *arcing*, um acrônimo para ***A****daptatively* ***R****esampling and* ***C****ombin****ing***. Breiman também criou um algoritmo do tipo *arcing*, que mostrou ter desempenho tão bom quanto AdaBoost. Isso mostrou que o fun-

cionamento do algortimo AdaBoost não estava relacionado diretamente à sua forma específica, e que existia uma classe de algoritmos que operava daquela maneira.

[Friedman, Hastie e Tibshirani 2000] mudaram totalmente o modo como *boosting* é visto, pelo menos na comunidade estatística. Eles colocaram *boosting* como uma aproximação do ajuste de um modelo aditivo na escala logística, usando máxima verossimilhança da Bernoulli como critério. Ademais, sugeriram uma aproximação mais direta, o que levou ao algoritmo LogitBoost, um algoritmo para ajustar uma regressão logística aditiva que dá resultados praticamente idênticos ao AdaBoost de Freund e Schapire.

Mais recentemente, notou-se que, se um algoritmo de *boosting* for executado por um tempo (número de iterações) muito grande, da ordem de dezenas de milhares, isso ocasionará *overfitting*. [Friedman, Hastie e Tibshirani 2000] dão um exemplo em que isso ocorre. Algumas abordagens para este problema foram tentadas. [Jiang 2000a] mostrou que, sob certas condições de regularidade, o algoritmo AdaBoost é consistente em processo (*process consistent*), no sentido de que, durante o treinamento, ele gera uma seqüência de classificadores com erro que converge para o erro do classificador (regra) de Bayes. [Lugosi e Vayatis 2004] mostraram um resultado importante de consistência para algoritmos de *boosting* com regularização.

Apresentamos primeiramente duas versões do algoritmo AdaBoost e o algoritmo LogitBoost no caso em que existem $J = 2$ classes. Após isso, mostramos o caso em que há $J > 2$ classes.

3.1 Caso de Duas Classes

O algoritmo AdaBoost discreto pode ser descrito da seguinte maneira. Considere o conjunto $L = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$, onde as classes estão rotuladas $\{-1, 1\}$.

Defina $F(\mathbf{x}) = \sum_1^M c_m f_m(\mathbf{x})$, onde f_m é um classificador base que retorna valores $\{-1, 1\}$, os valores c_m são constantes e a predição correspondente é o sinal de $F(\mathbf{x})$, ou seja, $sign(F(\mathbf{x}))$. O algoritmo AdaBoost ajusta classificadores base f_m em amostras reponderadas do conjunto de treinamento, dando maior peso, ou ponderação aos casos que são classificados erroneamente. Os pesos são ajustados adptativamente em cada iteração, e o classificador final é uma combinação linear dos classificadores f_m.

Algoritmo 3.1 (AdaBoost Discreto)

1. *Inicialize os pesos* $w_i = 1/N, i = 1, 2, \ldots, N$
2. *Repita para* $m = 1, 2, \ldots, M$ *:*
   (a) *Ajuste o classificador* $f_m(\mathbf{x}) \in \{-1, 1\}$ *usando os pesos* w_i *nos dados de treinamento*
   (b) *Calcule* $\epsilon_m = E_w[I_{(y \neq f_m(\mathbf{x}))}] = \frac{1}{N} \sum_{i=1}^N w_i I(y_i = f_m(\mathbf{x}_i))$,
   $c_m = \log((1 - \epsilon_m)/\epsilon_m)$
   (c) *Faça* $w_i \leftarrow w_i \exp\left[c_m I_{(y_i \neq f_m(\mathbf{x}_i))}\right], i = 1, 2, \ldots, N$ *e renormalize para que* $\sum_i w_i = 1$
3. *Saia com o classificador final* $sign(F(\mathbf{x})) = sign\left(\sum_{m=1}^M c_m f_m(\mathbf{x})\right)$

No algoritmo acima, E_w representa a média ponderada (esperança no conjunto de treinamento) com pesos $w = (w_1, \ldots, w_N)$. Em cada iteração m, os ϵ_m são calculados com base no vetor de pesos w e no acerto do classificador base f_m. Nos algoritmos de *boosting*, os classificadores base são ajustados da maneira usual com que seriam ajustados se fossem usados normalmente, mas os dados com que eles são ajustados (os dados de treinamento) são gerados, a princípio, de uma distribuição de

probabilidade baseado nos pesos w (na maioria dos algoritmos não há a amostragem aleatória em si; os pesos são incorporados diretamente no algoritmo).

Outra versão do algoritmo AdaBoost é o AdaBoost real, onde o classificador base f_m é um classificador que retorna a estimativa $p_m(\mathbf{x}) = \hat{P}_w(y = 1|\mathbf{x}) \in [0, 1]$. O algoritmo usa essas estimativas para construir as contribuições reais $f_m(\mathbf{x})$.

Algoritmo 3.2 (AdaBoost Real)

1. *Inicialize os pesos* $w_i = 1/N, i = 1, 2, \ldots, N$
2. *Repita para* $m = 1, 2, \ldots, M$ *:*
   (a) *Ajuste o classificador para obter uma estimativa de probabilidade de classe* $p_m(\mathbf{x}) = \hat{P}_w(y = 1|\mathbf{x}) \in [0, 1]$ *usando os pesos* w_i *nos dados de treinamento*
   (b) *Faça* $f_m =\leftarrow \frac{1}{2} \log p_m(\mathbf{x})/(1 - p_m(\mathbf{x})) \in \Re$
   (c) *Faça* $w_i \leftarrow w_i \exp\left[-y_i f_m(\mathbf{x}_i)\right], i = 1, 2, \ldots, N$ *e renormalize para que* $\sum_i w_i = 1$
3. *Saia com o classificador final sign* $\left(\sum_{m=1}^{M} f_m(\mathbf{x})\right)$

A seguir apresentamos o algoritmo LogitBoost, um algoritmo de *boosting* desenvolvido por [Friedman, Hastie e Tibshirani 2000], o qual é baseado num modelo logístico aditivo. Neste algoritmo, a resposta $y^* = \frac{y+1}{2}$ é um valor em $\{0, 1\}$.

Algoritmo 3.3 (LogitBoost)

1. *Inicialize os pesos* $w_i = 1/N, i = 1, 2, \ldots, N$,$F(\mathbf{x}) = 0$ *e estimativas de probabilidades* $p(\mathbf{x}_i) = \frac{1}{2}$.

2. *Repita para* $m = 1, 2, \ldots, M$ *:*

   (a) *Calcule*

   $$\begin{aligned} z_i &= \frac{y_i^* - p(\mathbf{x}_i)}{p(\mathbf{x}_i)(1 - p(\mathbf{x}_i))} \\ w_i &= p(\mathbf{x}_i)(1 - p(\mathbf{x}_i)) \end{aligned}$$

   (b) *Ajuste a função* $f_m(\mathbf{x})$ *fazendo a regressão de* z_i *em* $\mathbf{x}_i$ *por mínimos quadrados ponderados usando os pesos* w_i

   (c) *Atualize* $F(\mathbf{x}) \leftarrow F(\mathbf{x}) + \frac{1}{2} f_m(\mathbf{x})$ *e* $p(\mathbf{x}) \leftarrow (e^{F(\mathbf{x})})/(e^{F(\mathbf{x})} + e^{-F(\mathbf{x})})$

3. *Saia com o classificador final* $sign(F(\mathbf{x})) = sign\left(\sum_{m=1}^{M} c_m f_m(\mathbf{x})\right)$

3.2 Caso de J classes

No caso de J classes, considere J respostas y_j para o problema de J classes, cada uma assumindo valores em $\{-1, 1\}$. Ou seja, para cada objeto i, há J respostas y_{ij} que identificam se o objeto pertence a classe j ($y_{ij} = 1$) ou não ($y_{ij} = -1$). As classes são consideradas auto-exclusivas.

As versões para J classes do AdaBoost (AdaBoost.MH) e LogitBoost (LogitBoost.MH) são apresentadas em [Friedman, Hastie e Tibshirani 2000].

Algoritmo 3.4 (AdaBoost.MH)

1. *Expanda as N observações originais em $N \times J$ pares dados por :* $((\mathbf{x}_i, 1), y_{i1}), ((\mathbf{x}_i, 2), y_{i2}), \ldots, ((\mathbf{x}_i, J), y_{iJ}), i = 1, \ldots, N$. *A resposta y_{ij} é a resposta para a classe j, observação i.*
2. *Execute o algoritmo AdaBoost Real no conjunto aumentado, produzindo uma função* $F : \mathcal{X} \times (1, \ldots, J) \to \Re; F(\mathbf{x}, j) = \sum_m f_m(\mathbf{x}, j)$.
3. *Saia com o classificador* $\arg\max_j F(\mathbf{x}, j)$.

Na prática, a implementação deste algoritmo envolve executar o algoritmo AdaBoost Real em J conjuntos de dados. No $j-$ésimo conjunto de dados, as respostas usadas devem ser $y_{ij}, i = 1, \ldots, N$. Isso é equivalente a executar J processos de classificação, onde em cada um deles ajusta-se um modelo de uma classe contra as outras. Os classificadores resultantes são $F(\mathbf{x}, j)$, para $j = 1, 2, ..., J$. Para classificar um objeto representado pelo vetor $\mathbf{x}$, toma-se a classe cujo classificador assume o maior valor, ou seja, o argumento que maximiza as $F(\mathbf{x}, j)$ em j.

Para apresentar o algoritmo LogitBoost para J classes, definiremos a transformação logística simétrica múltipla.

Definição 3.1 *Para um problema de classificação em J classes, seja* $p_j(\mathbf{x}) = P(y_j = 1|\mathbf{x})$. *Definimos a transformação logística simétrica múltipla por*

$$F_j(\mathbf{x}) = \log p_j(\mathbf{x}) - \frac{1}{J} \sum_{k=1}^{J} \log p_k(\mathbf{x}). \tag{7}$$

Equivalentemente,

$$p_j(\mathbf{x}) = \frac{e^{F_j(\mathbf{x})}}{\sum_{k=1}^{J} e^{F_k(\mathbf{x})}}, \ \sum_{k=1}^{J} F_k(\mathbf{x}) = 0. \tag{8}$$

A seguir apresentamos o algoritmo LogitBoost para J classes, que é uma generalização natural do LogitBoost para duas classes.

Algoritmo 3.5 (LogitBoost (J classes))

1. *Inicialize os pesos* $w_{ij} = 1/N, i = 1, 2, \ldots, N,\ j = 1, \ldots, J,\ F_j(\mathbf{x}) = 0$ *e estimativas de probabilidades* $p_j(\mathbf{x}_i) = \frac{1}{J}\ \forall j$.

2. *Repita para* $m = 1, 2, \ldots, M$ *:*

   (a) *Repita para* $j = 1, \ldots, J$ *:*

      i. *Calcule*

$$\begin{aligned} z_{ij} &= \frac{y^*_{ij} - p_j(\mathbf{x}_i)}{p_j(\mathbf{x}_i)(1 - p_j(\mathbf{x}_i))} \\ w_{ij} &= p_j(\mathbf{x}_i)(1 - p_j(\mathbf{x}_i)) \end{aligned}$$

      ii. *Ajuste a função* $f_{mj}(\mathbf{x})$ *fazendo a regressão de* z_{ij} *em* $\mathbf{x}_i$ *por mínimos quadrados ponderados usando os pesos* w_{ij}

   (b) *Faça* $f_{mj}(\mathbf{x}) \leftarrow \frac{J-1}{J}\left(f_{mj}(\mathbf{x}) - \frac{1}{J}\sum_{k=1}^{J} f_{mk}(\mathbf{x})\right)$, *e atualize* $F_j(\mathbf{x}) \leftarrow F_j(\mathbf{x}) + f_{mj}(\mathbf{x})$.

   (c) *Atualize* $p_j(\mathbf{x})$ *via 8*

3. *Saia com o classificador final* $\arg\max_j F_j(\mathbf{x})$

3.3 O que Boosting Faz

Os algoritmos de *boosting* ajustam modelos aditivos através da otimização de um critério, ou funcional de custo. Cada base da expansão é um classificador simples que pode ser escolhido pelo usuário. Comparando com um modelo de regressão logística, por exemplo, o critério utilizado é a verossimilhança da binomial, e o modelo aditivo é linear. No algoritmo AdaBoost discreto, pode-se mostrar que o critério que está sendo otimizado é $e^{-yF(\mathbf{x})}$. A quantidade $yF(\mathbf{x})$ é negativa se, e somente se, o

classificador F errou a predição de y. O critério $e^{-yF(\mathbf{x})}$ é, portanto, um funcional de custo suave baseado na medida $yF(\mathbf{x})$. A razão para utilizar um funcional deste tipo é a tratabilidade matemática. Ainda é possível mostrar que esse critério é uma aproximação da log-verossimilhança da binomial.

Existem muitos algoritmos de *boosting* disponíveis, que utilizam funcionais de custo diferentes e métodos diferentes de otimização. O algoritmo LogitBoost, por exemplo, otimiza a log-verossimilhança da binomial através de um algoritmo do tipo Newton-Rhapson. A interpretação desses algoritmos é clara : eles são combinações lineares de classificadores simples, os quais são construídos de maneira a dar mais peso aos erros cometidos. Métodos lineares, contudo, não são beneficiados por *boosting* [Breiman 1998]. Isso ocorre porque os métodos lineares são estáveis, no sentido de que pequenas perturbações nos dados não produzem grandes perturbações nos classificadores gerados, e portanto o método de *boosting*, quando aplicado a esses métodos, apenas combina vários classificadores muito parecidos, produzindo um classificador final muito parecido ao original.

4 Simulação

Nesta seção damos um exemplo de aplicação do algoritmo AdaBoost Real em um conjunto de dados simulados, descrito em [Breiman 1998] (conjunto denominado *three-norm*).

Os dados possuem 2 classes e são gerados da seguinte maneira. Os dados da classe 1 são gerados com probabilidade $\frac{1}{2}$ de uma distribuição normal multivariada com média $(a, a, ...a)$, e com probabilidade $\frac{1}{2}$ de uma distribuição normal com média $(-a, -a, ..., -a)$, com matrizes de variância identidades nos dois casos, com $a = 2/(20)^{1/2}$. Os dados da classe 2 são gerados de uma distribuição normal mul-

tivariada com média $(a, -a, a, -a, ..., a)$ e matriz de variância identidade. O vetor de variáveis explanatórias tem 20 variáveis. Foram simulados 1000 valores para amostra de treinamento e 1000 valores para amostra de teste. Executamos os algoritmos AdaBoost Real e LogitBoost, com o número de iterações variando de 1 a 500, de 25 em 25. Todo o procedimento (simulação dos dados e execução dos algoritmos) foi executado 100 vezes, e foram calculadas as estimativas das taxas de erro para cada número de iterações como uma média das taxas de erro no conjunto de teste nas 100 repetições. Os modelos baseados no algoritmo AdaBoost Real foram ajustados usando-se o *software* R versão 1.7.1, com o pacote *gbm*. Os classificadores base utilizados nesta implementação são árvores de classificação. Os modelos baseados no algoritmo LogitBoost foram implementados no mesmo *software*. Para efeito de comparação, foram ajustados modelos estatísticos tradicionais de classificação, a saber, regressão logística e análise discriminante linear. Os modelos foram ajustados sem seleção de variáveis. Para o ajuste desses modelos foi utilizado o *software* SAS versão V8. Ambos os modelos foram ajustados em cada uma das amostras, em cada uma das 100 iterações. As taxas de erro em amostras teste foram calculadas em cada iteração, e foi utilizada a taxa de erro média para comparação.

O Gráfico 1 sumariza os resultados obtidos. A linha pontilhada mais abaixo, paralela ao eixo das abscissas, representa o erro do classificador de Bayes, que neste exemplo é de $0,105$. Esta é, a princípio, a menor taxa de erro atingível. Todas as outras linhas representam taxas de erro médias estimadas usando amostras teste. A linha tracejada representa a taxa de erro média dos modelos de regressão logística. A linha de traços e pontos representa a taxa de erro média dos modelos de análise discriminante linear. A linha sólida representa a taxa de erro média dos modelos contruídos pelo algoritmo AdaBoost, de acordo com o número de funções. Note que esta taxa de erro atinge a estabilidade após aproximadamente 75 iterações (nº de

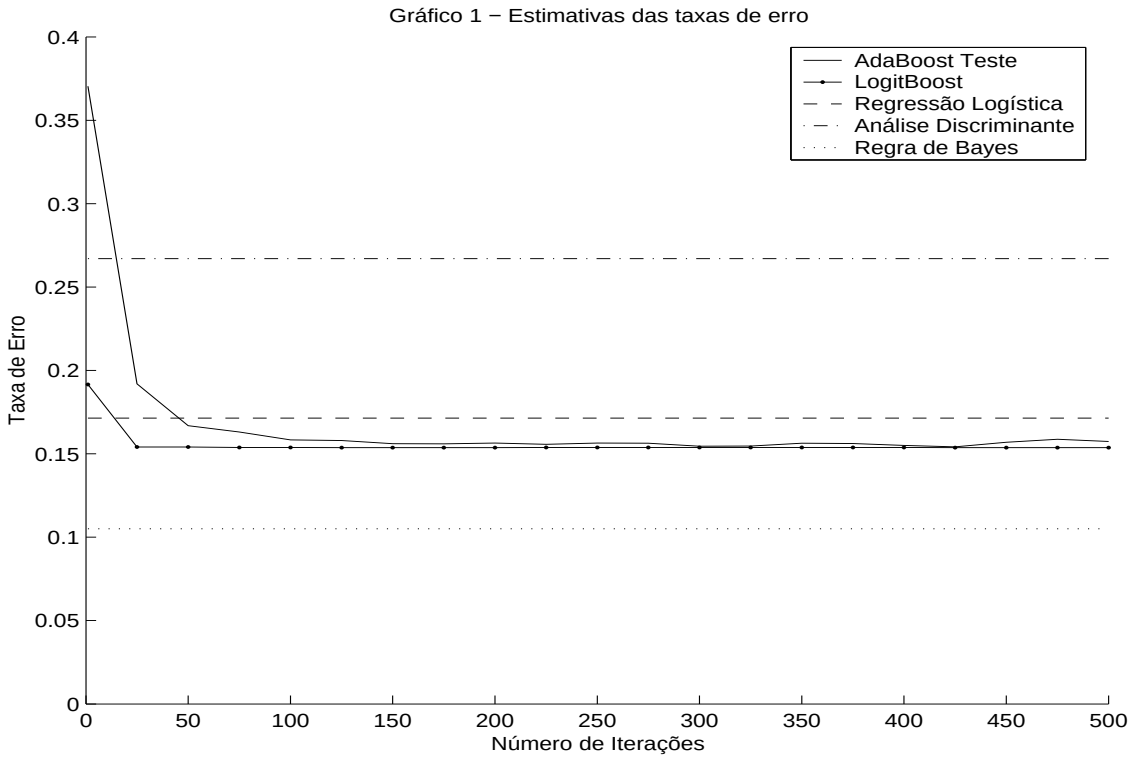
Gráfico 1 − Estimativas das taxas de erro
AdaBoost Teste
LogitBoost
Regressão Logística
Análise Discriminante
Regra de Bayes
Taxa de Erro
Número de Iterações
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
50
100
150
200
250
300
350
400
450
500

funções) do algoritmo. A linha sólida com pontos sobrepostos representa a taxa de erro média do algoritmo LogitBoost.

Nota-se através do gráfico que o desempenho do algoritmo AdaBoost é superior ao dos métodos regressão logística e análise discriminante linear. No caso da análise discriminante linear, o ganho é bem maior, e no caso da regressão logística, a diferença não é tão grande, mas o AdaBoost ainda é melhor. O algoritmo LogitBoost é ainda ligeiramente superior ao algoritmo AdaBoost.

Uma questão de interesse prático é o critério de parada para o algoritmo. Na maioria das vezes, executar o algoritmo por centenas de iterações não irá causar *overfitting*. Esse problema só surge, em geral, com números de iterações muito maiores, da ordem de milhares. Assim, se for utilizada uma amostra teste, pode-se parar o algoritmo quando o erro na amostra teste atinge um patamar de estabilidade. Se não houver amostra teste disponível, pode-se usar validação cruzada e parar quando a estimativa do erro por validação cruzada deixar de cair. Para evitar este tipo de critério, pode-se usar algoritmos de *boosting* regularizados. Estes algoritmos baseiam-se em otimizar funcionais de custo adicionados de um termo de penalização. Referências para algoritmos deste tipo são [Mason 1999, Jiang 2000b, Evgenious 2002].

5 Aplicação em Dados Reais

A aplicação em dados reais deste trabalho foi feita em dados de uma rede de lojas de varejo. O problema consiste em classificar clientes em grupos que definem seus perfis. Cada cliente pode pertencer a um de 10 grupos. O conjunto de dados possui, para cada cliente, observações em variáveis de interesse, assim como a classe (grupo) correspondente. O conjunto original continha 184 variáveis de vários tipos. Foram

excluídas 14 variáveis que ou não se encaixavam no tipo de variável que pudesse ser utilizada pelo algoritmo de *boosting*, ou que tivessem muitos dados faltantes (na implementação de *boosting* usada, só é possível utilizar variáveis contínuas, de maneira que excluímos as variáveis de qualquer outro tipo). O tamanho da base original era de aproximadamente meio milhão de clientes. Desta base foi selecionada uma amostra aleatória simples de tamanho aproximadamente 50000. Esta amostra foi dividida em duas : uma de tamanho aproximadamente 34000, para treinamento, e outra, de tamanho aproximadamente 16000, para teste.

Utilizando a amostra de treinamento, ajustamos inicialmente modelos de análise discriminante linear e quadrática. A análise discriminante linear foi superior a quadrática, de maneira que só reportamos os resultados para a linear. Com este mesmo conjunto de dados de treinamento, aplicamos o algoritmo 3.4, AdaBoost.MH, variando o número de funções de 25 a 400 e calculando, para cada passo, a estimativa da taxa de erro na amostra teste. A partir de 300 funções não houve melhora nesta taxa. Os modelos foram ajustados seguindo as mesmas considerações da seção 4.

Os resultados obtidos estão dispostos na Tabela 1.

Tabela 1 - Taxas de Erro estimadas com amostra Teste

Método	Taxa de Erro (Teste)
Análise Discriminante Linear	0,1549
AdaBoost com 25 função	0,1210
AdaBoost com 50 funções	0,0991
AdaBoost com 75 funções	0,0880
AdaBoost com 100 funções	0,0802
AdaBoost com 200 funções	0,0712
AdaBoost com 300 funções	0,0662
AdaBoost com 400 funções	0,0662

Como a tabela evidencia, o algoritmo AdaBoost foi bastante superior ao método tradicional de análise discriminante linear. Com uma combinação de apenas 25 iterações, essa superioridade já é evidente, e com 300 iterações, a melhora é notável. Aumentar o número de iterações acima de 300 não trouxe melhora, de maneira que um modelo final poderia conter este número de funções. O método de análise discriminante supõe distribuições *a priori* normais, e produz uma fronteira de decisão linear. A superioridade dos métodos de boosting se deve à sua capacidade de aproximar funções mais complexas, e portanto produzir fronteiras de decisão que proporcionam classificadores mais poderosos.

O custo computacional de executar o algoritmo AdaBoost.MH num conjunto de dados razoavelmente grande e complexo como este não foi muito superior ao custo computacional de executar o modelo de análise discriminante. O ganho obtido em precisão, não obstante, foi substancial. De fato, com poucas iterações, o custo computacional do AdaBoost é muito similar ao de Análise Discriminante. Conforme o número de funções cresce, entretanto, o custo computacional aumenta.

6 Considerações Finais

Apresentamos uma metodologia pouco conhecida na comunidade estatística em geral e, em especial, no Brasil. Nas aplicações apresentadas, esta metodologia, chamada *boosting* e apresentada em dois algoritmos diferentes, AdaBoost e LogitBoost, apresentou resultados muito bons em comparação aos métodos estatísticos tradicionais, como regressão logística e análise discriminante. *Boosting* pode ser implementado facilmente a partir de um classificador razoável, transformando-o em um comitê de classificação com desempenho muito superior ao classificador original. Seu custo computacional não é elevado, apesar de ser superior ao dos métodos

tradicionais, e o método pode lidar com grandes quantidades de informação, o que é o caso em muitas aplicações atuais.

A utilização de metodologias modernas e eficientes, como *boosting*, pode trazer muitos benefícios em diversas áreas onde o problema de classificação aparece.

Referências

[Agresti 1996]AGRESTI, A. *An Introduction to Categorical Data Analysis*. [S.l.]: John Wiley and Sons, Inc, 1996.

[Breiman 1984]BREIMAN, L. et al. *Classification and Regression Trees*. [S.l.]: Chapman-Hall/CRC, 1984.

[Breiman 1998]BREIMAN, L. Arcing classifiers. *The Annals of Statistics*, v. 26, n. 3, p. 801–849, 1998.

[Breiman 2000]BREIMAN, L. Some infinity theory for predictor ensembles. *Technical Report 577, Statistics Department, University of California*, 2000.

[Evgenious 2002]EVGENIOUS, T. et al. Regularization and statistical learning theory for data analysis. *Computational Statistics and Data Analysis*, v. 38, p. 421–432, 2002.

[Freund e Schapire 1997]FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, v. 55, p. 119–139, 1997.

[Friedman, Hastie e Tibshirani 2000]FRIEDMAN, J. H.; HASTIE, T.; TIBSHIRANI, R. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, v. 28, p. 337–407, 2000.

[Hosmer e Lemeshow 1989]HOSMER, D.; LEMESHOW, S. *Applied Logistic Regression*. [S.l.]: John Wiley and Sons, Inc, 1989.

[Jiang 2000a]JIANG, W. Process consistency for adaboost. *Technical Report 05, Department of Statistics, Northwestern University*, 2000.

[Jiang 2000b]JIANG, W. Is regularization unnecessary for boosting ? *Technical Report 04, Department of Statistics, Northwestern University*, 2000a.

[Johnson e Wichern 2002]JOHNSON, R. A.; WICHERN, D. W. *Applied Multivariate Statistical Analysis*. [S.l.]: Prentice Hall, New York, NY, 2002.

[Lugosi e Vayatis 2004]LUGOSI, G.; VAYATIS, N. On the bayes-risk consistency of regularized boosting methods. *The Annals of Statistics*, v. 32-1, 2004. To Appear.

[Mason 1999]MASON, L. et al. Functional gradient techniques for combining hypotheses. *Advances in Large Margin Classifiers*, 1999. MIT Press.

[Ripley 1996]RIPLEY, B. D. *Pattern Recognition and Neural Networks*. [S.l.]: Cambridge University Press, 1996.

[Schapire 1990]SCHAPIRE, R. E. The strength of weak learnability. *Machine Learning*, v. 5, p. 197–227, 1990.