Documentos

ISSN 1677-9274
Dezembro, 2015
135

Utilizando técnicas preditivas que combinam modelagem e seleção de atributos

Embrapa

ISSN 1677-9274
Dezembro, 2015

Empresa Brasileira de Pesquisa Agropecuária
Embrapa Informática Agropecuária
Ministério da Agricultura, Pecuária e Abastecimento

Documentos 135

Utilizando técnicas preditivas que combinam modelagem e seleção de atributos

Fábio Danilo Vieira
Stanley Robson de Medeiros Oliveira

Embrapa Informática Agropecuária
Campinas, SP
2015

Embrapa Informática Agropecuária
Av. André Tosello, 209 - Barão Geraldo
Caixa Postal 6041 - 13083-886 - Campinas, SP
Fone: (19) 3211-5700
www.embrapa.br/informatica-agropecuaria
SAC: www.embrapa.br/fale-conosco/sac/

1ª edição
publicação digitalizada 2015

Dados Internacionais de Catalogação na Publicação (CIP)
Embrapa Informática Agropecuária

Vieira, Fábio Danilo.

Utilizando técnicas preditivas que combinam modelagem e seleção de atributos / Fábio Danilo Vieira, Stanley Robson de Medeiros Oliveira. - Campinas : Embrapa Informática Agropecuária, 2015.

22 p. : il. - (Documentos / Embrapa Informática Agropecuária, ISSN 1677-9274 ; 135).

1. Mineração de dados. 2. Boosting. 3. Lasso. 4. Random forest. I. Oliveira, Stanley Robson de Medeiros. II. Título. III. Embrapa informática Agropecuária. IV. Série.

CDD (21. ed.) 006.33

Autores

Fábio Danilo Vieira
Tecnólogo em Processamento de dados, mestre em Engenharia Agrícola, analista da Embrapa Informática Agropecuária, Campinas, SP

Stanley Robson de Medeiros Oliveira
Cientista da computação, Ph.D. em Ciência da Computação, pesquisador da Embrapa Informática Agropecuária, Campinas, SP

Apresentação

Devido ao expressivo avanço tecnológico atingido nos últimos anos, enormes volumes de dados surgiram de forma acelerada em diversas áreas de conhecimento, sejam em universidades ou indústrias. Em decorrência disso, apareceu a necessidade do uso de ferramentas e técnicas automatizadas para trabalhar com esses dados, as quais pudessem auxiliar o analista a transformá-los em conhecimento potencialmente útil.

A maior parte dessas ferramentas e técnicas pode ser encontrada dentro do processo de mineração de dados, formalmente denominado de Descoberta de Conhecimento em Bases de Dados (do inglês Knowledge Discovery in Databases, ou KDD), que tem por objetivo encontrar padrões úteis que ajudem no entendimento de um problema e na tomada de decisão.

Entretanto, determinados conjuntos de dados são gerados com um número de atributos muito maior que o número de registros (ou observações), o que limita, e muito, o número de técnicas do processo KDD que podem ser aplicadas. Dentre exemplos de conjuntos de dados que possuem um número de atributos muito maior que o número de observações ($p >> n$) pode-se citar os obtidos por meio de tecnologias de genotipagem por marcadores moleculares SNP (do inglês Single Nucleotide Polimorphism), que geram dados com um elevado número desses marcadores (algumas vezes, centenas de milhares por observação) relacionados a um número bem menor de observações (algumas vezes, apenas algumas dezenas). Alguns outros exemplos de conjuntos de dados com um alto número de atributos (e que em grande parte das vezes supera o número de observações) que podem ser citados são: a) dados de imagens; b) textos; c)séries temporais; d) pesquisas médicas.

Para a obtenção de conhecimento desses tipos de conjuntos de dados, existem técnicas de mineração específicas para manipulá-los. Entre as técnicas mais utilizadas atualmente em trabalhos envolvendo modelagem para solução de problemas $p>>n$ estão: a) Lasso; b) Random Forest; c) Boosting. Essas técnicas têm a capacidade de combinar geração de modelos preditivos com seleção de atributos mais importantes para esses modelos. O objetivo deste trabalho é descrever as características mais importantes dessas técnicas e mostrar alguns exemplos de aplicação utilizando o software estatístico R.

Silvia Maria Fonseca Silveira Massruhá
Chefe-geral
Embrapa Informática Agropecuária

Sumário

Utilizando técnicas preditivas que combinam modelagem e seleção de atributos

Fábio Danilo Vieira
Stanley Robson de Medeiros Oliveira

Introdução

Percebe-se nos últimos anos, devido principalmente ao grande avanço da área de tecnologia de informação, que um enorme volume de dados cresce de forma acelerada em diversos campos de conhecimento, o que dificulta sua interpretação, pois o volume destes dados é maior que o poder de interpretá-los. Com isso, surgiu também a demanda por ferramentas e técnicas automatizadas para contornar esta situação, as quais pudessem ajudar o analista a transformar os dados em conhecimento (HAN et al., 2011).

Um bom número dessas técnicas e ferramentas pode ser encontrado no processo de Descoberta de Conhecimento em Bases de Dados, ou somente KDD, cuja sigla em inglês significa Knowledge Discovery in Databases. A descoberta de conhecimento em bancos de dados é definida como um processo que busca identificar padrões novos, potencialmente úteis e compreensíveis em um conjunto de dados, com o objetivo de facilitar o entendimento de um problema ou um procedimento de tomada de decisão (FAYYAD et al.,1996).

As aplicações de técnicas de mineração estão presentes em praticamente todos os setores do conhecimento humano. Na área de negócios, por exemplo, utilizam-se técnicas em detecção de fraudes em cartões, na

criação de perfis de clientes de acordo com suas compras, entre outros. Na agricultura, podem ser utilizadas em previsão de geadas, sistemas de alerta para a ferrugem do cafeeiro, sistemas de alerta para a ferrugem asiática da soja, etc. Na bioinformática, para se buscar padrões em sequências de DNA, entre muitas outras possibilidades.

Na etapa de mineração de dados propriamente dita, deve ser feita a escolha da tarefa a ser aplicada, assim como a definição do algoritmo. Esta escolha deve ser baseada nos objetivos que se deseja atingir com a solução a ser encontrada. As possíveis tarefas de um algoritmo para se extrair padrões podem ser agrupadas em preditivas e descritivas (HAN et al., 2011).

As tarefas preditivas têm como objetivo principal a construção de modelos que possam predizer a variável resposta de um novo exemplo a partir de exemplos ou experiências passadas com respostas já conhecidas. As tarefas descritivas procuram identificar padrões intrínsecos a um conjunto de dados que não possui uma variável resposta determinada. A escolha de uma ou mais tarefas dependerá do problema a ser solucionado. As tarefas tradicionais de mineração de dados são brevemente descritas a seguir.

- Classificação: consiste na predição do valor de um atributo alvo do tipo discreto ou categórico por meio da construção de modelos e regras a partir de um conjunto de exemplos pré-classificados corretamente, para posterior classificação de exemplos novos e desconhecidos (HAN et al., 2011).

- Regressão: constitui uma técnica estatística muito empregada para se realizar predições (HILL et al., 2003). Essas predições procuram encontrar tendências de variações no conjunto de dados analisado em função dos atributos existentes. Possui um conceito semelhante à classificação, porém se aplica na predição de um valor alvo do tipo contínuo.

- Associação: determinam o quanto a presença de um certo conjunto de atributos nos exemplos de uma base de dados implica na presença de algum outro conjunto de atributos nos mesmos exemplos (AGRAWAL; SRIKANT, 1994).

- Clusterização: é uma tarefa descritiva que procura identificar agrupamentos (*clusters*) finitos de objetos similares entre si e dissimilares entre os grupos no conjunto de dados. De forma diferente da classificação, onde as denominações de classes são conhecidas, a clusterização analisa os dados onde as denominações de classes não estão definidas.

Cada tarefa de mineração de dados possui diferentes técnicas associadas. Dentre as mais populares estão (HAN et al., 2011): a) árvores de decisão; b) redes neurais; c) regressão linear ou não linear; d) k-vizinhos mais próximos. Existem também as abordagens híbridas, que utilizam duas ou mais técnicas em conjunto. Não existe a técnica ideal, pois cada uma delas possui suas vantagens e desvantagens. Assim, ao se escolher uma técnica, deve ser realizada uma análise bem apurada do problema em questão, levando em consideração o formato dos dados e como o conhecimento descoberto pode ser representado.

A maior parte dessas técnicas, principalmente as técnicas preditivas, fornecem ótimos resultados (seja em termos de interpretação e/ou medidas de desempenho) utilizando conjuntos de dados em que o número de observações é normalmente muito maior que o número de atributos. Porém, para resolução de problemas envolvendo conjuntos de dados em que o número de atributos é muito maior que o número de registros (observações), poucas são as técnicas que conseguem gerar modelos com resultados satisfatórios.

Para problemas desse tipo, existem técnicas inovadoras com poder predição muito alto, combinando geração de modelos e seleção dos atributos mais relevantes, com taxas de acerto bem satisfatórias. Algumas dessas técnicas são: a) Lasso; b) Random Forest; c) Boosting. Essas técnicas são indicadas, principalmente, para bases de dados em que o número de atributos (p) é muito maior que o número de observações (n) ($p >> n$), utilizando procedimentos internos para controlar a seleção de atributos redundantes (JAMES et al., 2013).

Lasso

Lasso (TIBSHIRANI, 1997) é um método de regressão penalizada comumente aplicado em análises estatísticas, e ultimamente, também vem se constituindo uma alternativa bastante atraente para identificação de atributos relevantes em análise de dados de DNA (AYERS; CORDELL, 2010; WU et al., 2009).

A técnica Lasso possui um algoritmo próprio para reduzir os efeitos dos atributos que não estão ligados à classe, aproximando seus coeficientes para zero ou próximo de zero. Estimar o efeito de um atributo como zero é o mesmo que excluí-lo do modelo.

Em problemas de regressão, o método é usado para estimar os parâmetros $\beta = (\beta_1, \beta_2, ..., \beta_p)$ conforme o modelo da Equação 1:

$$\gamma_i = \mu + \sum_{j=1}^{p} x_{ij}\beta_j + e_i = \mu + X_i\beta + e_i \quad (1)$$

onde γ_i é o atributo meta (classe) do i-ésimo registro (observação) ($i = 1,2,..., n$); μ é o coeficiente denominado intercepto, cujo valor é comum a todos os registros; x_{ij} é o valor do atributo j ($j = 1,2,..., p$) do registro i; o coeficiente β_j representa o efeito do atributo j no atributo meta; e_i é o erro residual.

Lasso foi, inicialmente, formulado para os modelos de regressão linear, nos quais o atributo meta seja contínuo. Para regressão linear, a técnica Lasso baseia-se na minimização da Soma de Quadrados Residuais (SQR) para obter os valores dos coeficientes β_j (JAMES et al., 2013). No entanto, Lasso também pode ser utilizado para resolver um problema de classificação, onde o atributo alvo é do tipo texto (nominal ou ordinal), como uma extensão da regressão logística, onde se desenvolvem modelos a partir de dados cujo atributo alvo contém valores do tipo categórico.

Em problemas de classificação, Lasso estima os coeficientes β_j do modelo por meio da maximização do logaritmo da função de verossimilhança, impondo a restrição de que a soma dos valores dos coeficientes absolutos seja limitada por uma constante (HASTIE et al., 2011). A ideia básica por trás da maximização do logaritmo de verossimilhança é obter os coefi-

cientes de μ e β de forma que a probabilidade $\hat{p}(x)$ de predição da classe correta de uma observação, dado uma ou mais variáveis x, corresponda, em grande maioria das vezes, à classe observada (JAMES et al., 2013).

A função logística utilizada para fornecer a probabilidade de uma observação Y pertencer a uma classe l, dado um ou mais atributos x, é exibida na Equação 2, a qual fornece valores de saída entre 0 e 1 (quanto mais próximo de 1, maior a probabilidade do atributo ser importante para o modelo).

$$p(x) = Pr(Y = 1|x) = \frac{e^{\mu + \sum_{j=1}^{p} x_{ij}\beta_j}}{1 + e^{\mu + \sum_{j=1}^{p} x_{ij}\beta_j}} \quad (2)$$

Para desenvolver o modelo da Equação 2, utiliza-se a função de máxima verossimilhança (com penalização), descrita na Equação 5.

Com uma pequena manipulação na Equação 2, encontra-se a seguinte Equação 3:

$$\frac{p(x)}{1 - p(x)} = e^{\mu + \sum_{j=1}^{p} x_{ij}\beta_j} \quad (3)$$

A relação $p(x)/[1-p(x)]$ é denominada odds, a qual indica a chance (ou probabilidade) de um evento ocorrer dividida pela probabilidade da não ocorrência do mesmo evento. Aplicando-se o logaritmo nos dois lados, obtém-se a Equação 4 a seguir, denominada logit:

$$\log\left(\frac{p(x)}{1 - p(x)}\right) = \mu + \sum_{j=1}^{p} x_{ij}\beta_j \quad (4)$$

Sendo $\hat{\beta} = (\hat{\beta}_1, \hat{\beta}_2, ..., \hat{\beta}_p)^T$, a estimativa Lasso $(\hat{\mu},\hat{\beta})$ para problemas de classificação é definida pela função de máxima verossimilhança penalizada descrita na Equação 5:

$$\hat{l}(\mu,\beta) = \frac{1}{n} \sum_{i=1}^{n} [y_i(\mu + \sum_{j=1}^{p} x_{ij}\beta_j) - \log(1 + e^{\mu + \sum_{j=1}^{p} x_{ij}\beta_j})] \quad (5)$$

sujeito à restrição $\sum_{j=1}^{p} |\beta_j| \leq t$ para $t \geq 0$,

onde t (ou λ (lambda) em outras formulações) é um parâmetro de penalização que deve ser determinado separadamente. Normalmente, os algoritmos de implementação do Lasso fornecem o valor ótimo para tal parâmetro, utilizando uma análise por validação cruzada de um intervalo de n possíveis valores. Essa restrição permite que algumas estimativas dos coeficientes de regressão sejam exatamente zero, realizando simultaneamente um procedimento de encolhimento e seleção de modelos.

Exemplo:

Para ilustrar um exemplo de aplicação de Lasso, utilizou-se o software livre R (versão 3.0.1) e se instalou o pacote glmnet (FRIEDMAN et al., 2010) para a execução do algoritmo. A Figura 1 demonstra a utilização do pacote glmnet:

```
> lasso.out = cv.glmnet(x[train,], y[train], alpha=1)
> plot (lasso.out)
> bestlam = lasso.out$lambda.min
> newX <- as.matrix(x[train,])
> lasso.pred = predict (lasso.out, s=bestlam, newx=newX)
```

Figura 1. Exemplo de aplicação de Lasso utilizando R com o pacote glmnet.

No exemplo acima, a variável Lasso.out recebe o modelo Lasso gerado pelo comando cv.glmnet que, por validação cruzada, determina o valor ideal de λ (lambda) para os melhores resultados desse modelo. Os parâmetros para o comando são: a) uma matriz com os valores dos atributos preditores (x[train,]); b) um vetor com os valores do atributo meta (y[train]) e alpha=1 indicando que deve ser usado a restrição Lasso para geração do modelo (em vez de outras restrições, como Ridge Regression). Com o comando plot é possível gerar um gráfico representando os valores de coeficientes obtidos dos atributos mais relevantes para o modelo. A variável bestlam recebe o valor do melhor coeficiente λ do modelo Lasso.out. Já a variável newX, que foi usada para conjunto de teste no comando seguinte, foi criada a partir da mesma matriz para treinamento do modelo. Essas duas variáveis foram parâmetros do comando predict para a listagem dos valores dos coeficientes dos atributos do modelo. Aqueles atributos com coeficientes reduzidos a zero não possuem correlação suficiente com atributo meta.

Random Forest

Random Forest é uma técnica de predição (tanto para classificação quanto para regressão) desenvolvida por Breiman (2001), que consiste num comitê (ou conjunto) de árvores de decisão combinadas para solucionar problemas de classificação (ou regressão). Cada uma dessas árvores de decisão é construída utilizando uma amostra aleatória inicial dos dados e, a cada divisão desses dados, um subconjunto aleatório de m atributos é utilizado para escolha do atributo mais informativo. Para tanto, Random Forest utiliza a estratégia denominada bagging (Booststrap Aggregating) (BREIMAN, 1996), que consiste numa abordagem para criar classificadores em amostras bootstrap dos dados na montagem da floresta, e a seleção aleatória de atributos para a construção de cada árvore de decisão. O modelo final de Random Forest produz uma lista dos atributos mais importantes no desenvolvimento da floresta, que são determinados pela importância acumulada do atributo nas divisões dos nós de cada árvore da floresta (JAMES et al., 2013).

O funcionamento básico do algoritmo Random Forest, em problemas de classificação, segue os seguintes passos da Figura 2.

Dado um conjunto de dados $X = x_1, x_2, \ldots, x_j$ e $Y = y_1, y_2, \ldots y_k$.

Para $b = 1, 2, 3, \ldots B$, repita:

(a) Cria uma amostra *bootstrap* (X_b, Y_b) com n exemplos de (X, Y).

(b) Ajusta uma árvore de decisão f^b para o conjunto de treinamento (X_b, Y_b), utilizando m atributos para a escolha de cada nó.

Fim de repetição.

Gera o modelo final: $\hat{f}(x) = \sum_{b=1}^{B} f^b(x)$, que calcula os votos obtidos por cada modelo f^b, resultando uma classificação final de acordo com a votação majoritária.

Figura 2. Algoritmo básico da técnica Random Forest.

Fonte: Breiman (2001).

Na Figura 3, a seguir, é exibido um exemplo de um conjunto de árvores de decisão desenvolvidas utilizando amostras bootstrap e seleção aleatória de atributos.

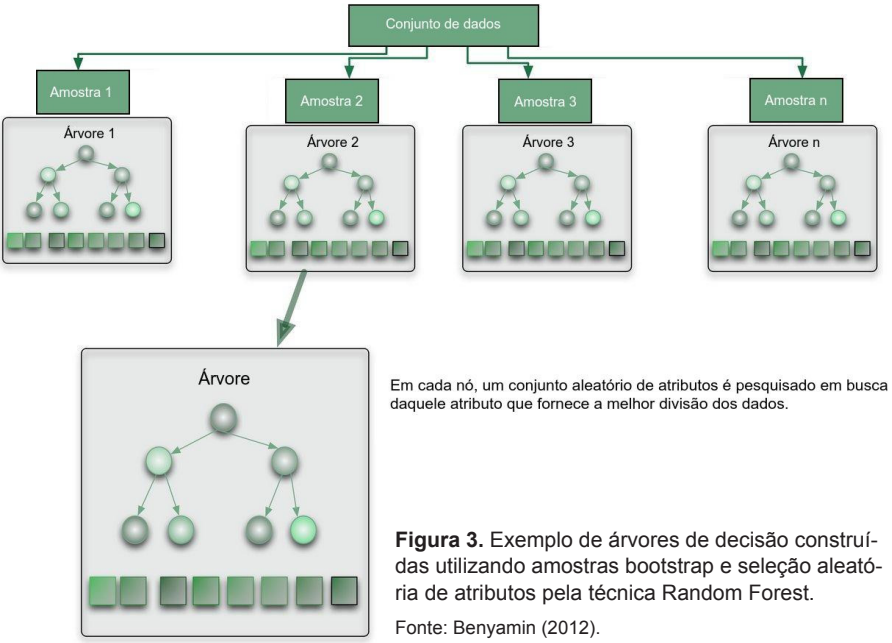

Figura 3. Exemplo de árvores de decisão construídas utilizando amostras bootstrap e seleção aleatória de atributos pela técnica Random Forest.

Fonte: Benyamin (2012).

A técnica Random Forest possui um bom desempenho na tarefa de classificação, muitas vezes comparável às Máquinas de Vetores de Suporte.. Embora seja menos utilizado que outros classificadores mais tradicionais, várias características favorecem o uso de Random Forest em grandes conjuntos de dados, tais como (BREIMAN, 2001):

- Aplicável em problemas que envolvam mais atributos do que amostras (instâncias).
- Possibilidade de utilização tanto em problemas que possuam duas classes (binários) quanto naqueles que possuem mais que duas classes (multiclasses).
- Fornece um bom desempenho nas predições nas quais os conjuntos de dados possuem muitos ruídos.
- Evita *overfitting*, ou seja, seu método de funcionamento direciona a produção de um classificador que não seja superajustado apenas aos dados treinados.

- Fornece medidas de importância de cada atributo.

Ainda de acordo com Breiman (2001), dada uma floresta construída de árvores simples e seus respectivos subconjuntos de atributos, Random Forest define uma função que mede a extensão em que o número de votos para uma dada classe excede a votação para qualquer outra classe.

Exemplo:

Para demonstrar o uso de Random Forest, a Figura 4 mostra um exemplo simples, também com o software R, utilizando o pacote randomForest (LIAW; WIENER, 2002):

```
> rf.out = randomForest(x=matriz_dados, y=class.dados, ntree=100, mtry=30, importance=TRUE)
> varImpPlot(rf.out, n.var=10)
```

Figura 4. Exemplo de aplicação de Random Forest utilizando R com o pacote randomForest.

No exemplo de uso de Random Forest, a variável rf.out recebe o modelo gerado pelo pacote randomForest, cujos parâmetros são, respectivamente, uma matriz x com os valores dos atributos preditores, um vetor y com os valores do atributo alvo, número de árvores a serem construídas na floresta (ntree), número de atributos de cada subconjunto aleatório utilizado na construção de cada uma das árvores, que normalmente é obtido pela raiz quadrada do número total de atributos do conjunto de dados ($mtry = \sqrt{n}$), e, por fim, o parâmetro importance, com valor TRUE, para indicar que o modelo deve fornecer a classificação dos atributos mais relevantes em relação à classe. Após isso, é chamada a função varImpPlot para exibição de um gráfico de barras ilustrando os 10 atributos (determinado por n.var=10) mais importantes (ou relevantes) para o modelo rf.out.

Boosting

A técnica Boosting (SCHAPIRE, 1990) também funciona como um comitê de classificadores, tendo como ideia central transformar múltiplos classificadores ruins em um único muito bom. Entretanto, de forma diferente que Random Forest, os métodos desta abordagem funcionam aplicando-se

sequencialmente um algoritmo de classificação a versões reponderadas do conjunto de dados de treinamento, dando maior peso aos registros classificados erroneamente no passo anterior.

Em outras palavras, da mesma forma que Random Forest, a técnica Boosting funciona modificando a amostra de treinamento. Mas, enquanto Random Forest modifica essa amostra aleatoriamente, através de re-amostragem, Boosting produz, em cada passo, uma distribuição dando maior peso às observações classificadas erroneamente no passo anterior (HASTIE et al., 2011).

O algoritmo que mostra a execução básica da técnica Boosting é descrito na Figura 5.

Dado um conjunto de dados de treinamento $X = x_1, x_2, \ldots, x_j$ e $Y = y_1, y_2, \ldots y_k$.

Define $\hat{f}(x) = 0$ e $residuos_i = y_i$ para todos os registros do treinamento.

Para $b = 1, 2, 3, \ldots B$, repita:

(a) Ajusta um modelo f^b para o conjunto de treinamento $(X, residuos)$.

(b) Atualiza $\hat{f}$ com o novo modelo:

$$\hat{f}(x) = \hat{f}(x) + f^b(x).$$

(c) Atualiza os resíduos (erros na classificação):

$$residuos_i = residuos_i - f^b(x_i).$$

Fim de repetição.

Gera o modelo final: $\hat{f}(x) = \sum_{b=1}^{B} f^b(x)$, que calcula os votos obtidos por cada modelo f^b, resultando uma classificação final de acordo com a votação majoritária.

Figura 5. Algoritmo básico da técnica Boosting.

Fonte: James et al. (2013).

Considere o exemplo seguinte, ilustrado na Figura 6, onde há uma distribuição de duas classes representadas pelos sinais de positivo (+) e negativo (-). Na rodada 1, os pontos classificados erroneamente (sinais positivos circulados) têm seus pesos aumentados para a próxima etapa. Na rodada 2, devido aos pesos dos erros, a nova classificação desloca a reta vertical, gerando novos erros (sinais negativos circulados). Na terceira rodada, uma reta horizontal foi traçada, dessa vez com menos erros (sinal negativo circulado). No final, a combinação dos classificadores desenvolvidos resulta

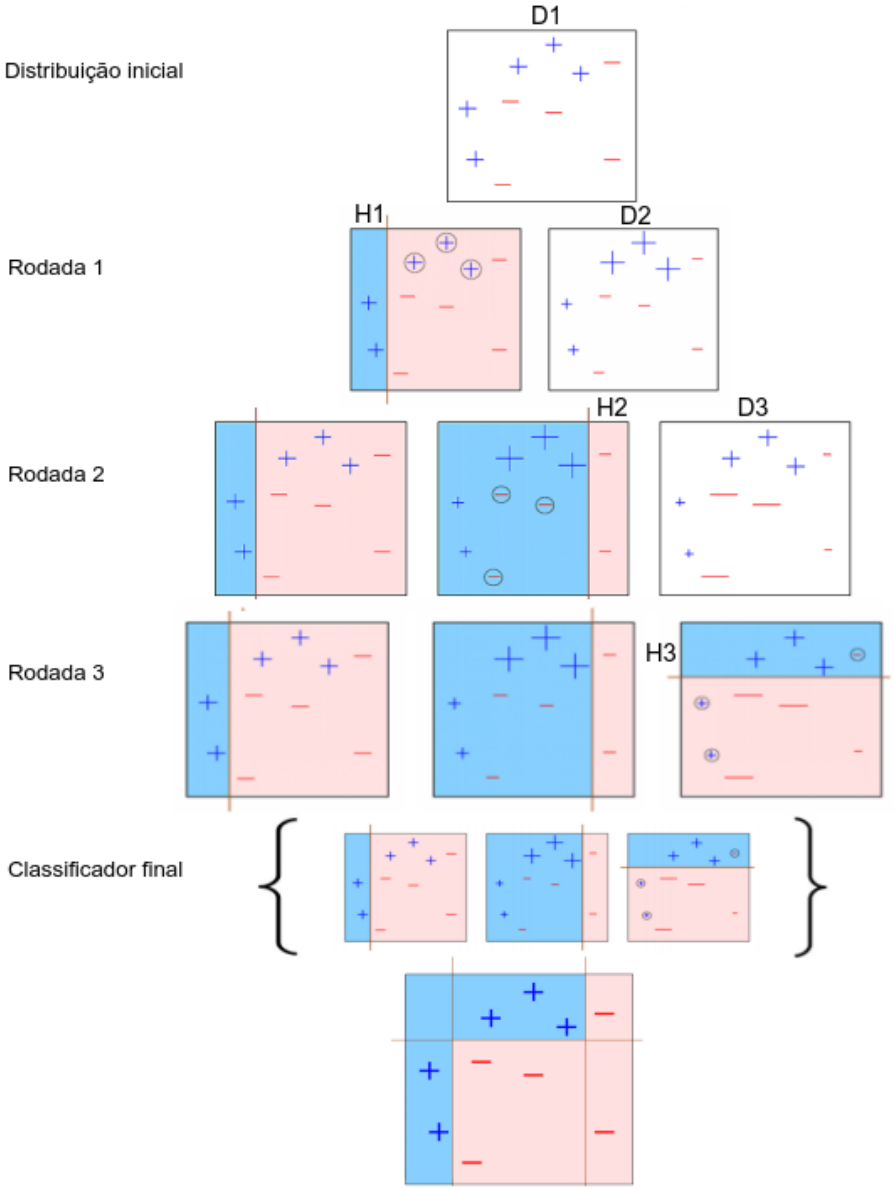

Figura 6. Dinâmica de funcionamento da técnica Boosting.

Fonte: Adaptado de Freund e Schapire (2015).

num classificador global muito superior, sendo que cada classificador tem um peso distinto na votação final do comitê.

Um dos limites teóricos do algoritmo Boosting implica que o erro na amostra de treinamento decai exponencialmente com o número de iterações do algoritmo. Empiricamente, observa-se que, após algumas iterações, o erro na amostra de treinamento é muito menor que na amostra de teste, confirmando o resultado teórico, conforme mostra a Figura 7 (JAMES et al., 2013).

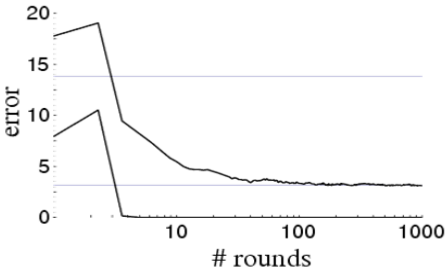

Figura 7. Curvas de erros de treinamento (abaixo) e teste (acima) indicando que quanto maior o número de iterações, menor o erro nos dados de treinamento.

Fonte: Freund e Schapire (1999).

Boosting também fornece uma lista das variáveis mais importantes no desenvolvimento do conjunto de classificadores, que são obtidas pela importância acumulada da variável nas divisões dos nós de cada árvore construída (JAMES et al., 2013).

Exemplo:

Como exemplo para a técnica Boosting, foi utilizado o algoritmo gbm (RIDGEWAY, 2013), também para o software R, demonstrado na Figura 8:

```
> gbm.out = gbm.fit(x=matriz_dados, y=class_dados, distribution="multinomial", n.trees=1000)
> summary(gbm.out, cBars=5)
```

Figura 8. Exemplo de aplicação de Boosting utilizando R com o pacote gbm.

Nesse exemplo de Boosting, a variável gbm.out recebe o modelo gerado pelo algoritmo gbm.fit, cujos parâmetros são, respectivamente: a) matriz x com os dados dos atributos preditores; b) vetor y com os valores do atributo alvo; c) distribution igual a “multinomial” para indicar que o atributo alvo tem mais de 2 classes; e d) n.trees para indicar que o comitê de classifica-

dores deve ser composto por 1000 árvores de decisão para gerar o modelo final. O comando summary produz um gráfico de influência relativa dos cinco melhores atributos (indicado por cBars=5) do modelo e também um relatório com informações estatísticas desses atributos.

Considerações finais

A utilização de técnicas de mineração apropriadas para buscar conhecimento e informações úteis em conjuntos de dados com um número de atributos muito maior que o de observações ($p>>n$), em particular as técnicas Lasso, Random Forest e Boosting, demonstra ser uma potencial solução para problemas desse tipo. Para aplicação dessas técnicas, diversos algoritmos estão disponíveis. O software R, por exemplo, disponibiliza pacotes para cada uma das técnicas apresentadas, sendo que o uso de cada uma delas mostrou ser de simples implementação, com possibilidades diversas de configuração e gerando saídas com listagens dos atributos mais relevantes para o modelo final.

Referências

AGRAWAL, R.; SRIKANT, R. Fast algorithms for mining association rules. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 20., 1994, Santiago. **Proceedings**… San Francisco: Morgan Kaufmann, 1994. p. 487-499.

AYERS, K. L.; CORDELL, H. J. SNP selection in genome-wide and candidate gene studies via penalized logistic regression. **Genetic Epidemiology**, v. 34, n. 8, p. 879-891, Dec. 2010. DOI: 10.1002/gepi.20543.

BENYAMIN, D. **A gentle introduction to random forests, ensembles, and performance metrics in a commercial system**. 2012. <https://citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics/>. Acesso em: 13 ago. 2015.

BREIMAN, L. Bagging predictors. **Machine Learning**, v. 24, n. 1, p. 123-140, 1996.

BREIMAN, L. Random forests. **Machine Learning**, v. 45, n. 1, p. 5-32, Oct. 2001. DOI: 10.1023/A:1010933404324

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery: an overview. In: **Advances in knowledge discovery & data mining**. Menlo Park: American Association for Artificial Intelligence, 1996. p. 1-34.

FREUND, Y.; SCHAPIRE, R. A short introduction to boosting. **Journal of Japanese Society for Artificial Intelligence**, v. 14, n.5, p. 771-780, Sept. 1999.

FREUND, Y.; SCHAPIRE, R. **A tutorial on boosting**. [2015]. Disponível em: <http://www.cc.gatech.edu/~thad/6601-gradAI-fall2013/boosting.pdf>. Acesso em: 13 ago. 2015.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. Regularization paths for generalized linear models via coordinate descent. **Journal of Statistical Software**, v. 33, n. 1, p. 1-22, Aug. 2010.

HAN, J.; KAMBER, M.; PEI, J. **Data mining**: concepts and techniques. 3rd ed. Burlington: Elsevier, 2011. 703 p. ill.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The elements of statistical learning**: data mining, inference, and prediction. London: Springer, 745 p. 2009.

HILL, C. M.; MALONE, L. C.; TROCINE, L. Data Mining and Traditional Regression. In: BOZDOGAN, H. **Statistical data mining and knowledge discovery**. Boca Raton: Chapman & Hall, 2003. p. 17.

JAMES, G.; HASTIE, T.; TIBSHIRANI, R. **An introduction to statistical learning**: with applications in R. New York: Springer, 2013. 426 p. ill.

LIAW, A.; WIENER, M. Classification and regression by randomforest. **R News**, v. 2, n. 3, p.18-22, Dec. 2002.

RIDGEWAY, G. gbm: generalized boosted regression models. R package. Version 2.1. 2013. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.398.7110&rep=rep1&type=pdf>. Acesso em: 13 ago. 2015.

SCHAPIRE, R. The strength of weak learnability. **Machine Learning**, v. 5, p. 197-227, June, 1990.

TIBSHIRANI, R. Regression shrinkage and selection via the Lasso, **Journal of the Royal Statistical Society**. Series B, v. 16, p. 385-395, 1997.

Embrapa

Informática Agropecuária

Ministério da
Agricultura, Pecuária
e Abastecimento

GOVERNO FEDERAL
BRASIL
PÁTRIA EDUCADORA

CGPE 12432