Boosting

Victor Freguglia Souza
RA: 137784
and
Leonardo Uchoa Pedreira
RA: 156231

5 de Dezembro de 2018

Resumo

The text of your abstract. 200 or fewer words.

Keywords: Quando for a versão final., Tirar esse campo na .tex;

1 Introdução

Boosting é o nome dado a um tipo de algoritmo que, assim como outros métodos em Machine Learning como Bagging e Florestas Aleatórias, busca combinar um grande número de preditores com baixo poder de predição (isto é, um pouco mais eficientes do que a escolha ao acaso) para compor um bom preditor. O conceito é fundamentado nas ideias apresentadas em Kearns (1988) e Schapire (1990).

Diferentemente dos outros métodos citados, onde os preditores fracos que serão combinados são criados de maneira independente (e aleatória devido ao processo de bootstrap), no método de Boosting os preditores fracos são criados de maneira a melhorar o desempenho em regiões com altas taxas de erro. Isto é, se pensarmos que cada preditor tem um "voto" na decisão final, o método nos fornece um comitê em que aqueles que tem grande convicção têm mais poder na decisão. Estes de grande convicção, sabem muito sobre uma parte do espaço amostral (não sei se amostral é a melhor palavra).

O algoritmo AdaBoost (de *Adaptative Boosting*), apresentado pela primeira vez em Freund & Schapire (1997), é um dos exemplos mais clássicos de algoritmo de boosting para o problema de classificação binária. Nele, a cada passo m, um novo classificador G_m é ajustado com base em uma versão ponderada do conjunto de dados original, na qual o peso de cada observação depende do desempenho do classificador anterior: pontos classificados de maneira errada recebem peso maior, e assim, têm uma chance maior de serem corrigidos pelos classificadores ajustados na próxima iteração. O Algoritmo 1 apresenta a descrição completa do AdaBoost. Note que os classificadores G_m a serem usados não precisam ser de nenhum tipo específico e, portanto, se comporta como um parâmetro a ser escolhido por um processo de regulagem (tuning) do problema. Apesar disso, o mais comum, pela grande flexibilidade, é a árvore de classificação e regressão (CART).

Embora o AdaBoost seja um bom ponto de início para entender o conceito de Boosting, ele foi projetado para resolver problemas de classificação binária, e por isso, não apresenta resultados tão bons quando diretamente adaptado a problemas de classificação múltipla e regressão, então diferentes algoritmos são necessários para esses casos. Hastie et al. (2009) propõe uma modificação do AdaBoost para o caso de classificação múltipla, por exemplo.

A principal diferença entre os diferentes algoritmos de Boosting propostos está na ma-

início

$(y_i, x_i), i = 1, \ldots, N$; $y_i \in \{-1, 1\}$; $x_i \in \mathbb{R}^p$;

Inicie todos pesos $w_i = 1/N$;

para $m = 1, \ldots, M$ **faça**

1. Ajuste um classificador $G_m(x)$;
2. Calcular

$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i};$$

3. Calcular $\alpha_m = \log\left(\frac{1-\text{err}_m}{\text{err}_m}\right)$;
4. Atualizar $w_i \leftarrow w_i \exp\left(\alpha_m I(y_i \neq G_m(x_i))\right), i = 1, \ldots, N$;

fim

Defina o classificador final como

$$G(x) = \text{sign}(\sum_{m=1}^{M} \alpha_m I(G_m(x) = k)).$$

fim

Algoritmo 1: Algoritmo AdaBoost apresentado em Friedman et al. (2001). Aqui, as classes do problema de classificação são representadas pelos valores -1 e 1.

neira com que os a ponderação dos dados é feita em cada passo. Nesse trabalho, por apresentar uma forma mais geral, será considerado o algoritmo Gradient Boosting, que pode lidar com todos os tipos de problemas de predição com a devida escolha da função perda a ser minimizada. A teoria que o fundamenta e sua descrição detalhada são apresentadas na Seção 2. O método é então aplicado ao conjunto de dados MNIST, descrito na Seção 3 e os resultados são mostrados na Seção 4.

2 Metodologia

2.1 Contexto e Visão Geral

Em problemas de regressão, escolher o melhor preditor significa estimar uma função f^* que minimiza o risco, definido como o valor esperado da função de perda $L(\cdot, \cdot)$, ou seja,

$$f^* = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{Y|x}\big[L(Y, f(x))|x\big]. \tag{1}$$

Note que o risco para um preditor específico f é desconhecido, uma vez que não assumimos nenhuma distribuição, o que impossibilita o cálculo do valor esperado da função perda para esse preditor. Uma boa estratégia para criação da função de predição $f \in \mathcal{F}$ consiste em escolher preditores de forma a minimizar o risco empírico. Isto é,

$$f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} L(y_i, f(x_i)) \tag{2}$$

Uma forma de resolver o problema 2 é via utilização de algorítmos numéricos. Gradiente descendente (citar watkins), por exemplo, é uma estratégia habitualmente utilizada, que levaria ao algorítmo

$$f_m = f_{m-1} - \rho_k \sum_{i=1}^{N} \nabla_f L(y_i, f(x_i)) \tag{3}$$

Entretanto (Friedman (2001)) cita que simplesmente utilizar isto teria efeitos catastróficos no modelo preditivo. Primeiro porquê só são levados em consideração as observações que temos (i.e, não existe nenhuma intenção de extrapolar poder preditivo, como por exemplo a separação entre teste e treinamento ou validação cruzada faria), segundo que não é levado em consideração à relação entre as covariáveis. Para contornar este problema, o autor sugere sequencialmente (de forma aditiva) ajustar uma quantidade m de árvores de decisão aos pseudo-resíduos, obtidos de predições subsequentes. Assim, o problema de otimização 2 se torna

$$f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} L(y_i, f(x_i)) \tag{4}$$

sujeito à que f seja árvore,

e que oferece como preditor $f_{boost}(x) = \sum_{m=1}^{M} \text{Árvore}_m$.

2.2 Árvores de decisão

Árvore de decisão é uma técnica que busca criar partições disjuntas R_j, $j = 1, \ldots, J$ em que γ_j é uma constante associada à cada partição terminal (Friedman et al. (2001)). Neste caso, se uma observação x pertence à região j, a predição para ela será γ_j. Formalmente, tal árvore pode ser escrita como

$$T(x; \Theta) = \sum_{j=1}^{J} \gamma_j I(x \in R_j) \tag{5}$$

onde $\Theta = \{R_j, \gamma_j\}$, $j = 1, \ldots, J$. Bem, se nosso objetivo é usar várias árvores em 4, precisamos de uma maneira de estimar Θ. Ou seja, na ótica da Teoria de Decisão, para a m-ésima árvore (ou seja, m-ésimo passo) queremos

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x; \Theta)), \tag{6}$$

pois como citado anteriormente (2.1) o ajuste é feito de maneira aditiva e nos resíduos. Para isto existem métodos que fornecem as regiões R_j, como o Particionamento Recursivo (Friedman et al. (2001)). Mas agora que conhecemos as regiões R_j da árvore m, considere que nos encontramos em uma partição terminal j dela, isto é, $I(x \in R_j) = 1$, $\forall x \in R_j$. Neste caso, o que nos falta é

$$\hat{\gamma_{jm}} = \arg\min_{\gamma_{jm}} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \gamma_{jm}), \tag{7}$$

que dependerá da função de perda escolhida. Friedman et al. (2001) fornece uma tabela de soluções analíticas para γ, de acordo com algumas perdas. Assim, conseguimos obter Θ_m, $\forall m = 1, \ldots, M$.

2.3 Otimização por Gradiente

Gradiente Descendente (ou seu irmão, Mais Íngrime Descida) é um método de otimização numérica que busca obter o mínimo de uma função. Sua proposta é começar em um ponto

inicial e incrementar a função avaliada naquele na direção oposta ao gradiente, em uma certa "velocidade" ρ_k, como em 3.

Entretanto, ao considerar que temos um problema de minimização com restrição o algorítmo torna-se (Friedman et al. (2001))

$$f_m = f_{m-1} - \rho_k \sum_{i=1}^{N} \left[\nabla_f L(y_i, f(x_i))\right]_{f(x_i)=f_{m-1}(x_i)} \tag{8}$$

onde

$$\rho_k = \arg\min_{\rho} L(f_{m-1}(x_i) - \rho\left[\nabla_f L(y_i, f(x_i))\right]_{f(x_i)=f_{m-1}(x_i)}) \tag{9}$$

2.4 Gradient Boosting

A direção fornecida da solução pelo método do Gradiente e sua conexão com os resíduos fornecem a intuição e a engrenagem por detrás de Boosting. Boosting é baseado em modelos aditivos, onde sequencialmente se ajustam modelos lineares generalizados nos resíduos. Ou seja, no passo m, temos o problema

$$\min_{\beta_m, \gamma_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta_m b(x_i; \gamma_m)) \tag{10}$$

onde b é uma função de base. Se, por exemplo, usamos a perda quadrática o problema torna-se

$$\min_{\beta_m, \gamma_m} \sum_{i=1}^{N} (r_{m-1}(x_i) - \beta_m b(x_i; \gamma_m))^2, \tag{11}$$

em que $r_{m-1}(x_i) = y_i - f_{m-1}(x_i)$. Ao compararmos a equação acima ao problema de regressão linear simples, $r_{m-1}(x_i)$ toma o papel de y_i e $\beta_m b(x_i; \gamma_m)$, o papel de βx_i. Portanto, a analogia leva à conclusão de se ajustar uma função de base aos resíduos. Boosting funciona de forma similar, onde γ_m torna-se Θ_m.

Se olharmos a equação equação 6, a predição da árvore $T(x_i; \Theta)$ é justamente o argumento que fornece o mínimo. Este é o mesmo papel que o valor negativo do gradiente desempenha em 2, o que os tornam similares, em certo sentido. Além disso, também existe

a semelhança entre 7 e 9 (onde a diferença é que, no caso das árvores, a busca pela solução ótima é restrita ao nó terminal).

Para perceber onde tudo se encaixa, vamos voltar ao exemplo da perda quadrática. Na equação 8,

$$\left[\nabla_f L(y_i, f(x_i))\right]_{f(x_i)=f_{m-1}(x_i)} = -2(y_i - f_{m-1}(x_i)) := -g_{im}.$$

Isto fornece

$$\begin{aligned}
\hat{\Theta}_m &= \arg\min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x;\Theta)) \\
&= \arg\min_{\Theta_m} \sum_{i=1}^{N} (y_i - f_{m-1}(x_i) - T(x;\Theta))^2 \\
&= \arg\min_{\Theta_m} \sum_{i=1}^{N} (-gim - T(x;\Theta))^2.
\end{aligned}$$

De acordo com a analogia anterior para o caso de modelos aditivos, g_{im} aqui tem uma forte conexão com os resíduos (para este exemplo, ele de fato é). Na verdade, ele é chamado de *pseudo-resíduo*, pois para problemas de classificação, os "resíduos" são, na verdade, a margem de classificação (Friedman et al. (2001) cita como exemplo a perda exponencial e sua interpretação para o algorítmo AdaBoost). Esta é o argumento chave que está por detrás da idéia do Boosting, a conexão entre os pseudo-resíduos e a direção do gradiente.

Se juntarmos as idéias e formularmos um algorítmo, temos o boosting por gradiente (ou

gradient boosting) como feito em Friedman (2001). Isto fornece o algorítmo a seguir.

início

$(y_i, \mathbf{x}_i), i = 1, \ldots, N$;

Inicie com o preditor constante $H_0 = \arg\min_c \sum_{i=1}^{N} L(y_i, c)$;

para $m = 1, \ldots, M$ **faça**

1. Calcular
$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f}\right]_{G=f_{m-1}(x_i)};$$
2. Ajustar uma nova árvore f_m ao conjunto de dados $(r_{im}, \mathbf{x}_i)$ com J regiões terminais $R_{jm}, j = 1, \ldots, J$
3. Para $j = 1, \ldots, J$, obtenha
$$\hat{\gamma_{jm}} = \arg\min_{\gamma_{jm}} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \gamma_{jm});$$
4. Atualize $f_m(x_i) = f_{m-1}(x_i) + \sum_{j=1}^{J} \gamma_{jm} I(x_i \in R_{jm})$;

fim

O preditor final é então $f_{boost}(x) = f_M(x)$.

fim

Algoritmo 2: Algoritmo Gradient Boosting apresentado em Friedman (2001).

Importante: Boosting surgiu para problemas de classificação binária e foi adaptado para regressão e classificação de várias categorias. Para classificação de K classes, para $m = 1, \ldots, M$, ajuste K árvores de classificação binária e use uma função perda apropriada, como a softmax (Friedman (2001)), que é a mais habitual para este tipo de problema.

2.5 PCA Whitening

Talvez isso não seja necessário pois habitualmente as arvores nao precisam desse tipo de tratamento. Svm e redes habitualmente precisam pq usam demais combinacoes lineares.

Mas essa secao pode ser util pra fazer comparacao com os outros metodos em que isso serve e melhorar a secao de discussao.

3 Conjunto de Dados Fashion-MNIST

Para uma ilustração do funcionamento do método de Gradient Boosting com dados reais, será considerado o problema de classificação no conjunto de dados Fashion-MNIST. Similar ao popular conjunto de dados MNIST de classificação de dígitos, o Fashion-MNIST também conta com pequenas imagens de 28 por 28 pixels em escala de cinza para classificação de peças de roupa em 10 categorias. São elas:

- T-Shirt

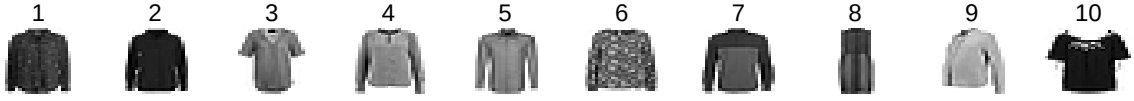
- Trouser

- Pullover

- Dress

- Coat

- Shirt



- Sandal

- Sneaker

- Bag

- Ankle Boot

Estão disponíveis em `https://www.kaggle.com/zalando-research/fashionmnist/home` dois conjuntos de dados: Um conjunto de treinamento, contendo 60000 imagens e um conjunto de teste, com 10000 imagens. Tanto o conjunto de treino quanto o de teste estão balanceados com exatamente 10% (6000 e 1000, respecitavamente) observações de cada categoria. Cada pixel da imagem é considerado uma variável preditora, totalizando $28 \times 28 = 784$. Portanto, temos um total de $n = 60000$ observações e $p = 784$.

A priori, parece existir um grande potencial de confundimento entre algumas categorias. Por exemplo, *Coat, Shirt, Dress* e *T-Shirt* apresentam estruturas similares, com estruturas parecidas para o tronco e as mangas. As diferenças ficam em alguns detalhes, como as diferença de comprimento para a manga e a parte do tronco. A mesma dificuldade também parece aparecer na comparação entre *Sneaker* e *Ankle Boot*, onde a principal diferença é apenas no comprimento da parte do tornozelo. Outra coisa que parece diferencias batante imagens da mesma categoria é o ângulo da foto. Embora a maioria tenha sido tirada de frente para a peça, algumas estão em posições diferentes, podendo gerar um confundimento extra.

4 Aplicação

Existem diversas ferramentas disponíveis para ajuste de algoritmos de Boosting. Em particular, para a linguagem R, os mais populares são os pacotes *"xgboost"(Extreme Gradient Boosting), "lightGBM"* e no framework "h2o". Para esse trabalho, os ajustes serão feitos usando o pacote *"xgboost"*, por apresentar maior flexibilização dos parâmetros de maneira mais simples e por também estar implementado através da função *train* do pacote *caret*, o que facilita a comparação com outros modelos ajustados.

O conjunto original de treinamento foi separado em dois subconjuntos, um subconjunto para realizar o treinamento de fato (90%) e um outro subconjunto de validação (10%), de maneira aleatória. O conjunto de teste será deixado exclusivamente para a avaliação da performance dos modelos ajustados. Estratégias mais sofisticadas de validação cruzada não foram utilizadas por questões de tempo. Os métodos utilizados tendem a ter ajustes lentos, o que torna o ajuste em um grid de parâmetros muito caro, principalmente se quantidades de árvores M grandes pertencem ao grid.

Os principais objetivos são: Avaliar a qualidade de predição do Gradient Boosting no problema de classificação das peças de roupa, comparar o poder de predição do Boosting com o das Florestas Aleatórias (árvores dependentes versus árvores independentes) e investigar o efeito de pré-processamento para os métodos, por exemplo, através de Análise de Componentes Principais (PCA) nas covariáveis.

Para comparação dos métodos, foram feitos ajustes de diversos tipos de modelos com diferentes configurações de parâmetros. Para resumir as informações, serão reportadas as métricas apenas das configurações de parâmetros que apresentaram melhor desempenho no conjunto de validação, ou seja, do grid escolhido para ajustar cada modelo, apenas a melhor configuração no grid será reportada. Os modelos ajustados para comparação foram:

- Gradient Boosting com árvores de classificação: Taxa de aprendizado $\eta = 0.08$, árvores f_m com profundindade máxima 4, taxa de amostragem de 80% para cada árvore e um total de $M = 500$ passos.
- Florestas Aleatórias: Total de 1000 árvores, com $m = 34 \sim \sqrt{p}$ covariáveis amostradas para cada árvore.
- Gradient Boosting com árvores de classificação + PCA Whitening: PCA utilizada

Tabela 1: Acurácia de cada modelo por conjunto

	Treinamento	Teste	Tempo..em.segundos.
GBM	0.9977167	0.9132	6935
Florestas Aleatórias	1.0000000	0.8881	31794
GBM + PCA	0.6568833	0.5569	9794
Florestas Aleatórias + PCA	1.0000000	0.5016	14559

no pré-processamento, todas as componentes foram utlizadas. Taxa de aprendizado $\eta = 0.08$, árvores f_m com profundindade máxima 4, taxa de amostragem de 80% para cada árvore e um total de $M = 500$ passos.

- Florestas Aleatórias + PCA Whitening: PCA utilizada no pré-processamento, todas as componentes foram utlizadas. Total de 1000 árvores, com $m = 34 \sim \sqrt{p}$ covariáveis amostradas para cada árvore.

Os resultados obtidos por cada método em termos de acurácia são apresentados na Tabela 1. Além disso as Tabelas 2 a **??** apresentam as matrizes de confusão de cada caso.

Tabela 2: Matriz de Confusão para predições com Gradient Boosting no conjunto de teste

	Predito									
	T-Shirt	*Trouser*	*Pullover*	*Dress*	*Coat*	*Sandal*	*Shirt*	*Sneaker*	*Bag*	*Ankle Boot*
T-Shirt	885	1	10	20	1	1	130	0	2	0
Trouser	1	985	0	5	0	0	0	0	1	0
Pullover	13	1	846	9	44	0	61	0	4	0
Dress	12	10	12	928	21	0	21	0	0	0
Coat	1	1	70	19	884	0	54	0	2	0
Sandal	0	1	0	0	0	960	0	4	1	1
Shirt	80	1	56	19	47	1	728	0	8	1
Sneaker	0	0	0	0	0	26	0	966	2	26
Bag	8	0	6	0	3	2	6	0	979	1
Ankle Boot	0	0	0	0	0	10	0	30	1	971

Tabela 3: Matriz de Confusão para predições com Florestas Aleatórias no conjunto de teste

	Predito									
	T-Shirt	*Trouser*	*Pullover*	*Dress*	*Coat*	*Sandal*	*Shirt*	*Sneaker*	*Bag*	*Ankle Boot*
T-Shirt	865	2	8	18	1	0	163	0	1	0
Trouser	0	972	1	7	0	0	1	0	1	0
Pullover	13	5	803	7	60	0	96	0	8	0
Dress	28	15	11	933	26	0	28	0	0	0
Coat	1	1	116	20	868	0	69	0	3	0
Sandal	1	1	0	0	0	952	0	14	2	7
Shirt	79	4	51	15	42	0	628	0	7	1
Sneaker	0	0	0	0	0	33	0	933	2	39
Bag	13	0	10	0	3	4	15	0	976	2
Ankle Boot	0	0	0	0	0	11	0	53	0	951

Tabela 4: Matriz de Confusão para predições com Gradient Boosting com PCA no conjunto de teste

	Predito									
	T-Shirt	*Trouser*	*Pullover*	*Dress*	*Coat*	*Sandal*	*Shirt*	*Sneaker*	*Bag*	*Ankle Boot*
T-Shirt	686	22	46	208	183	2	249	0	25	0
Trouser	0	803	1	110	0	0	0	0	1	0
Pullover	43	6	427	14	275	11	267	0	94	11
Dress	130	149	3	643	27	1	50	0	9	0
Coat	76	11	306	14	413	3	190	0	24	11
Sandal	4	0	4	0	2	538	13	183	45	36
Shirt	43	7	149	10	72	8	188	0	55	4
Sneaker	0	0	0	0	0	345	0	694	20	101
Bag	17	2	60	1	27	33	39	0	484	144
Ankle Boot	1	0	4	0	1	59	4	123	243	693

Tabela 5: Matriz de Confusão para predições com Gradient Boosting

	Predito									
	T-Shirt	*Trouser*	*Pullover*	*Dress*	*Coat*	*Sandal*	*Shirt*	*Sneaker*	*Bag*	*Ankle Boot*
T-Shirt	518	17	49	177	135	4	187	0	22	1
Trouser	8	805	3	157	4	0	5	0	4	1
Pullover	52	7	359	12	280	10	241	0	85	18
Dress	167	139	10	578	42	2	59	0	12	0
Coat	108	15	263	24	327	4	183	0	40	8
Sandal	3	1	6	0	4	547	12	256	42	49
Shirt	111	14	217	44	167	8	238	1	64	8
Sneaker	0	0	0	0	0	313	0	609	23	106
Bag	30	2	79	8	36	33	68	10	441	215
Ankle Boot	3	0	14	0	5	79	7	124	267	594

5 Discussão

Tanto o algoritmo de Gradient Boosting quanto as Florestas Aleatórias apresentaram um excelente desempenho para a classificação das peças de roupa no conjunto de teste, com uma vantagem de pouco mais de 2% de acurácia para o GBM. Além disso, como esperado, a maioria dos casos de confundimento ocorreram entre as classes *Coat, Shirt, Dress* e *T-Shirt*. A Figura **??** apresenta algumas das peças classificadas de forma errada pelo método de Gradient Boosting.

Sobre o custo computacional, comparado aqui através do tempo de execução (foram utilizadas máquinas do www.kaggle.com para os ajustes), os algoritmos de Florestas Aleatórias demoraram mais que os de Boosting nos 2 casos (com e sem PCA), mesmo que, ao adicionar PCA, o tempo de execução das Florestas Aleatórias tenha reduzido bastante, uma vez que, com menos variáveis importantes, a complexidade das árvores ajustadas tende a diminuir, acelerando o ajuste. Também é importante ressaltar, que se dividirmos o tempo de execução pelo número de árvores ajustadas, no caso com PCA, as árvores ficam mais rápidas que o Boosting pelo menos motivo.

O número de passos $M = 500$ selecionado foi o maior testado no grid de escolha dos

Figura 1: Exemplos de erros de classificação do GBM. Classe predita em vermelho, classe verdadeira em preto.

parâmetros, o que indica que, possivelmente, a classificação poderia ser ainda mais precisa se fossem utilizados mais passos, enquanto com as 1000 árvores consideradas as Florestas Aleatórias já atingiam 100% de acurácia no conjunto de treinamento, indicando pouco potencial de melhora com o investimento de mais árvores.

Com respeito ao efeito do pré-processamento por Análise de Componentes Principais (PCA), utilizando o método apenas como uma rotação (PCA-Whitening), a consequência foi uma diminuição muito grande no poder de predição dos métodos. Isso acontece porque a rotação apenas faz com que variáveis se transformem na direção de maior variabilidade, mas sem levar em consideração sua relação com a resposta. Como consequência, alguns detalhes importantes para predição mas com pouca variabilidade são "diluídos" em muitas componentes de pouca importância e acabam sendo de difícil identificação pelos preditores. Esse problema não ocorreria, por exemplo, se regiões lineares discriminassem bem as respotas, o que geralmente não é caso de dados de imagens.

6 Conclusão

Através da investigação dos resultados da aplicação e da discussão dos pontos apontados nas seções de Discussão e Metodologia, pudemos concluir que o algoritmo de Gradient Boosting produz preditores muito bons por ter sido construído com uma estratégia muito eficiente e que pode ser regulada através da inclusão de diversos parâmetros de ajuste em suas variações para evitar problemas característicos dos problemas. Como consequência, percebemos que, se construído e ajustado de maneira apropriada, ele produz resultados melhores do que a estratégia de Florestas Aleatórias, tanto do ponto de vista de erro de predição quanto em custo computacional. De maneira geral, criar novas árvores simples baseado nas qualidades e defeitos do "comitê" a cada passo parece mais eficiente do que criar novas árvores de maneira aleatória e independente das qualidades e defeitos do preditor.

A rotação da matriz das covariáveis através do uso de Componentes Principais não é apropriada para o tipo de problema considerado, pois a rotação induz a perda de características importantes para o problema de classificação. O fato dos métodos testados serem capazes de criar regiões de separação altamente não-lineares é uma das maiores vantagens de se usar árvores, mas essa vantagem acaba sendo perdida devido à rotação feita. Por

outro lado, caso essa rotação fosse apropriada, uma redução de variáveis poderia ser feita para ajustar os modelos de maneira mais rápida, assim, mais passos do Gradient Boosting poderiam ser feitos com o mesmo tempo, possivelmente obtendo preditores ainda melhores.

Referências

Freund, Y. & Schapire, R. E. (1997), 'A decision-theoretic generalization of on-line learning and an application to boosting', *Journal of computer and system sciences* **55**(1), 119–139.

Friedman, J. H. (2001), 'Greedy function approximation: a gradient boosting machine', *Annals of statistics* pp. 1189–1232.

Friedman, J., Hastie, T. & Tibshirani, R. (2001), *The elements of statistical learning*, Vol. 1, Springer series in statistics New York, NY, USA:.

Hastie, T., Rosset, S., Zhu, J. & Zou, H. (2009), 'Multi-class adaboost', *Statistics and its Interface* **2**(3), 349–360.

Kearns, M. (1988), 'Thoughts on hypothesis boosting', *Unpublished manuscript* **45**, 105.

Schapire, R. E. (1990), 'The strenght of weak learnability', *Machine Learning* **5**, 197–227.