Boosting

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

6 de Dezembro de 2018

Resumo

É apresentada a ideia geral de Boosting e descrita detalhadamente a construção do algorítmo de Gradient Boosting e suas principais características. O algorítmo é aplicado a dados de classificação de peças de roupa através de imagens e seu desempenho é comparado com o de Florestas Aleatórias, tanto ao utilizar Análise de Componentes Principais nas covariáveis como pré-processamento e não usar pré-processamento algum. Como conclusão, o Gradient Boosting apresenta resultados superiores aos das Florestas Aleatórias mesmo com um número menor de iterações, tanto em poder preditivo como custo em computacional reduzido. Além disso, a inclusão de PCA diminui drásticamente o poder preditivo de ambas os algorítmos. Por fim, é feita uma discussão sobre os aspectos teóricos e características do problema que levaram às diferenças.

Keywords: Gradient Boosting, Classificação, Imagens, Componentes Principais, Árvores de Classificação

1 Introdução

Boosting é o nome dado a um tipo de algorítmo que, assim como outros métodos em Aprendizado de Máquina 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 consecutivos preditores fracos são criados de maneira a melhorar o desempenho dos anteriores, 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 certas partes do espaço amostral.

O algorítmo AdaBoost (de *Adaptative Boosting*), apresentado pela primeira vez em Freund & Schapire (1997), é o exemplo mais clássico de algorítmo 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 algorítmo 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 algorítmos são necessários para esses casos. Hastie et al. (2009) propõe uma modificação do AdaBoost para o caso de classificação múltipla, por exemplo.

Nesse trabalho, por apresentar uma forma mais geral, será considerado o algorítmo Gra-

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

dient Boosting, que pode lidar com todos os tipos de problemas de predição com a devida escolha da função perda a ser minimizada. O objetivo do trabalho é apresentar e discutir as principais propriedades do Gradient Boosting, bem como possíveis ajustes e comparar com outro método baseado em árvores (Florestas Aleatórias) através de uma aplicação a dados reais. A parte teórica do método e suas descrição detalhada são apresentadas na Seção 2. O método é então aplicado ao conjunto de dados Fahsion-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, 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 que só são levados em consideração as observações que temos (isto é, não existe nenhuma intenção de extrapolar poder preditivo, como a separação entre teste e treinamento ou validação cruzada faria) e, segundo, não é levada em consideração a 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

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

onde $\mathcal{F}$ é o conjunto de árvores todas as de decisão,

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, $j = 1, \ldots, J$ e associar constantes d_j a cada partição (Friedman et al. (2001)). Neste caso, se uma observação x pertence à região R_j, a predição para ela será d_j. Formalmente, tal árvore pode ser escrita como

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

onde $\Theta = \{R_j, d_j\}$, $j = 1, \ldots, J$. Com o objetivo de usar várias árvores em 4, precisamos de uma maneira de construir Θ. Ou seja, na ótica da Teoria de Decisão, para a m-ésima árvore (ou 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 para definir as regiões R_j^m, como o de Particionamento Recursivo (Friedman et al. 2001). Uma vez conhecidas as regiões R_j^m da m-ésima árvore, é necessário estimar ainda d_j^m

$$\hat{d_j^m} = \arg\min_{d} \sum_{i: x_i \in R_j^m} L(y_i, f_{m-1}(x_i) + d), \tag{7}$$

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

2.3 Otimização por Gradiente

Gradiente Descendente é um método de otimização numérica para obter o valor mínimo de uma função. Sua proposta é começar em um ponto inicial e incrementar a função avaliada neste ponto, na direção oposta ao gradiente, em uma certa "velocidade" ρ_k, como em 3. As iterações são dadas por

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

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 são ajustam modelos lineares generalizados nos resíduos. Ou seja, no passo m, temos o problema

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

onde b é uma função de base (uma função constante por partes, no caso das árvores). Se considerarmos a perda quadrática, o problema torna-se

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

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 $\beta_m b(x_i; \Theta_m)$, o papel de βx_i. Portanto, a analogia leva à conclusão de se ajustar uma função de base aos resíduos.

Para perceber onde tudo se encaixa, vamos voltar ao exemplo da perda quadrática,

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

Isto fornece

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

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 a principal motivação 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), obtemos o algorítmo a seguir.

início

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

Inicie com o preditor constante $f_0(x) = c^* = \arg\min_c \sum_{i=1}^{N} L(y_i, c)$;

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 $R_j^m, j = 1, \ldots, J$
3. Para $j = 1, \ldots, J$, obtenha
$$\hat{d}_j^m = \arg\min_d \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + d);$$
4. Atualize $f_m(x_i) = f_{m-1}(x_i) + \sum_{j=1}^{J} \hat{d}_j^m I(x_i \in R_j^m)$;

fim

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

fim

Algoritmo 2: algorítmo 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 Parâmetros de Refinamento

O propósito de adicionar parâmetros de refinamento no modelo é evitar o sobreajuste para melhorar sua predição e diminuir o custo computacional. Em boosting por gradiente, temos dois componentes principais: o número de árvores M e seus respectivos tamanhos, ou profundidades. Além disso, pode-se adicionar vários outros parâmetros, como aqueles implementados nas ferramentas disponíveis. De muitos, destacam-se dois: taxa de amostragem e taxa de aprendizado.

2.5.1 Profundidade das Árvores

Para uma árvore de decisão, a abordagem padrão é crescer uma árvore e, para diminuir sua variância, podar ela Friedman et al. (2001). Entretanto, Friedman et al. (2001) cita que para boosting este procedimento levaria à criação de árvores muito grande no inicio e, consequentemente, seria extremamente custoso ao ponto de talvez tornar o método inviável. Uma alternativa então é controlar o tamanho máximo, ou profundidade, de cada árvore.

A profundidade das árvores tem um papel muito importante no Gradient Boosting, pois ele é parecido com a ordem de interações em modelos de regressão. Se utilizarmos uma decomposição ANOVA para funções, Friedman et al. (2001) diz que

$$f(x) = \sum_{j} f_j(x) + \sum_{jk} f_{jk}(x_j, x_k) + \sum_{jkl} f_{jkl}(x_j, x_k, x_l) + \ldots. \tag{12}$$

Neste caso $f_j(x)$ são todas as funções de um parâmetro que melhor aproximam $f(x)$ e estão associados aos "efeitos principais". $f_{jk}(x_j, x_k)$ são todas as funções de dois parâmetros que melhor aproximam $f(x)$ e estão associados às interações de segunda ordem e assim por diante. Ao fazer conexão com árvores de decisão, o nível de interação é limitado pelo tamanho de cada árvore.

Como visto em disciplinas de delineamento de experimentos, interações de altos níveis são raras. Isto, aliado à experiência dos autores, levou à conclusão de que por mais que um modelo com somente "efeitos principais" seja simplório demais, interações de ordem 10 são desnecessárias. É comentado então que valores habituais estão entre 4 e 8.

2.5.2 Taxa de Aprendizado

A taxa de aprendizado η é, na verdade, uma estratégia de encolhimento (do mesmo tipo associado à Regressão Ridge), cujo propósito é evitar sobreajuste do modelo e melhorar sua predição. Para boosting o passo 4 do algorítmo 2 é levemente modificado de forma que agora

$$f_m(x_i) = f_{m-1}(x_i) + \eta \sum_{j=1}^{J} d_j^m I(x_i \in R_j^m).$$

Note a semelhança com ρ, na equação 3 (por este motivo é chamado de taxa de aprendizado). Em Friedman et al. (2001) é citado uma dependência entre os valores de η e M. Habitualmente valores pequenos de η levam a valores grande de M. Como em Friedman (2001) foi-se descoberto empiricamente que pequenos η fornecem os menores erros de predição, quando avaliados no conjunto de teste, a estratégia para escolha deles é fixar valores pequenos de η e buscar aqueles M que levam à erros de predição satisfatórios.

2.5.3 Taxa de Amostagem

A idéia aqui é simplesmente de que, a cada iteração, ajusta-se a árvore de decisão em uma fração ν (habitualmente $\nu = 1/2$ ou até mesmo menos) dos dados, sorteada aleatoriamente. Com isso Friedman (2001) reporta que o custo computacional cai bastante e que, em muitos casos, o modelo torna-se mais acurado. Porém, é importante citar que sem utilizar encolhimento, o desempenho do modelo costuma cair drasticamente.

2.6 PCA Whitening

Árvores de decisão buscam fazer divisões do espaço em regiões perpendiculares a algum eixo e a definição dessas regiões é feita de acordo com características da resposta nos

pontos, mas não dos pontos em si. Isso faz com que as predições feitas sejam invariantes a transformações lineares de cada variável individualmente. Por exemplo, se a melhor divisão em um determinado nó era com a regra $(x_1 > 2)$, os mesmos grupos serão obtidos pelas regra $(2x_1 > 4)$ ou $(x_1 + 1 > 3)$. Portanto, o ajuste de árvores (e consequentemente do Gradient Boosting) são invariantes por translação e escala das covariáveis individualmente, como normalizações. Logo, esse tipo de pré-processamento não traz nenhum tipo de benefício ou prejuízo para métodos baseados em árvores.

Por outro lado, árvores não são capazes de criar regras como $(x_1 + x_2 > 0)$, embora esse tipo de regra possa ser aproximado por várias regras consecutivas envolvendo apenas uma variável por vez. Isso faz com que sejam consideradas possíveis transformações lineares que combinem variáveis, a fim de se conseguir criar regras que dividem o espaço em outras direções. Se uma regra envolvendo uma combinação linear de covariáveis é ótima, então essa regra única seria mais precisa e mais simples do que uma sequência de regras "univariadas".

Uma das maneiras de se combinar variáveis é por meio de Análise de Componentes Principais ou PCA (Hotelling 1933), onde se busca uma transformação ortogonal da matriz X, que faça uma rotação do espaço na direção de maior variabilidade. As componentes (combinações lineares) dificilmente têm qualquer interpretabilidade, mas podem ser úteis, por exemplo para redução de dimensionalidade, uma vez que a maioria da variabilidade usualmente se concentra em um número de componentes relativamente menor que a dimensão original e são não-correlacionadas.

É de interesse investigar os efeitos do uso desse tipo de rotação para o algorítmo de Gradient Boosting, se a rotação pode tornar as divisões do espaço pelas árvores mais eficiente em termos de predição e/ou computacionais e comparar com outros métodos baseados em árvores, como Florestas Aleatórias.

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

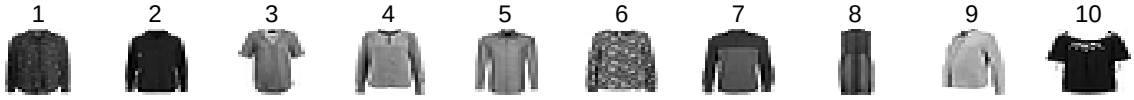
- Trouser

- 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 diferenças de comprimento para a manga e a tronco. A mesma dificuldade também aparece na comparação entre *Sneaker* e *Ankle Boot*, onde a principal diferença é apenas no comprimento do tornozelo. Outra coisa que parece diferenciar bastante 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 algorítmos de Boosting. Em particular, para a linguagem R, os mais populares são os pacotes *"xgboost"* (Chen et al. 2018), *"lightGBM"* (Ke 2018) e no framework "h2o" (LeDell et al. 2018). 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* (Kuhn 2018), 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 estimação da verdadeira performance dos modelos ajustados. Estratégias mais sofisticadas de validação cruzada não foram utilizadas por questões de tempo. Os métodos utilizados costumam ter ajustes lentos, o que torna o ajuste em um grid de parâmetros muito custoso, principalmente se uma grande quantidade de árvores M pertence 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 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: 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

Tabela 1. Além disso as Tabelas 2 a 5 apresentam as matrizes de confusão de cada caso. GBM se refere a "Gradient Boosting Machine".

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 Florestas Aleatórias com PCA

	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 algorítmo de Gradient Boosting quanto das 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 1 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 algorítmos 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 mesmo 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 que 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 porquê 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 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, concluímos que o algorítmo 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 à 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

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y. & Li, Y. (2018), *xgboost: Extreme Gradient Boosting*. R package version 0.71.2.
URL: *https://CRAN.R-project.org/package=xgboost*

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.

Hotelling, H. (1933), 'Analysis of a complex of statistical variables into principal components.', *Journal of educational psychology* **24**(6), 417.

Ke, G. (2018), *lightgbm: Light Gradient Boosting Machine*. R package version 2.2.3.
URL: *https://github.com/Microsoft/LightGBM*

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

Kuhn, M. (2018), *caret: Classification and Regression Training*. R package version 6.0-80.
URL: *https://CRAN.R-project.org/package=caret*

LeDell, E., Gill, N., Aiello, S., Fu, A., Candel, A., Click, C., Kraljevic, T., Nykodym, T., Aboyoun, P., Kurka, M. & Malohlava, M. (2018), *h2o: R Interface for 'H2O'*. R package

version 3.20.0.8.

URL: *https://CRAN.R-project.org/package=h2o*

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