Aprendizado incremental e classe-incremental por meio da atualização de árvores geradoras em florestas de caminhos ótimos

Mateus Riva*, Teófilo de Campos†‡, Moacir Ponti*†
*ICMC/Universidade de São Paulo — São Carlos/SP, Brazil
†CVSSP/University of Surrey — Guildford, United Kingdom
‡FGA/Universidade de Brasília — Gama/DF, Brazil

***Resumo*—Algoritmos com capacidade classe-incremental, em que é preciso manter modelos de classificação atualizados a partir de dados que aparecem ao longo do tempo, são importantes em diversas aplicações. Apresentamos neste artigo um algoritmo para atualização de florestas de caminhos ótimos capaz de incluir uma nova instância mantendo as propriedades das árvores de caminhos ótimos. Além de uma prova demonstrando que o algoritmo mantém a estrutura das árvores de caminhos ótimos em tempo linear, as evidências experimentais demonstram a aplicabilidade do método, que partindo de um modelo limitado, e após a inclusão de múltiplas instâncias, é capaz de alcançar a mesma acurácia de um classificador treinado considerando o conjunto completo de treinamento.**

***Abstract*—Class-incremental algorithms, where there is the need to update classification models with data that emerges over time, are important in many applications. We report an algorithm for updating optimum-path forests capable of including a new instance, maintaining the properties of the optimum-path trees. In addition to a proof demonstrating that the algorithm maintains the structure of the trees in linear time, the experimental evidence shows the applicability of the method, which starting from a limited model, and after the inclusion of multiple instances, is able to achieve the same accuracy of a classifier trained with the full training set.**

***Keywords*-aprendizado incremental; minimum spanning trees; optimum-path forests**

I. Introdução

Em diversas aplicações de reconhecimento de padrões novos dados se tornam disponíveis ao longo do tempo, sejam observações individuais ou pequenos lotes, tornando necessário atualizar constantemente o modelo inicial [1]. Esse cenário é comumente conhecido como aprendizado incremental, o qual exige um método para atualização rápida do modelo atual, visto que retreinar o modelo não é uma solução viável do ponto de vista de tempo de execução [2]. Nesse artigo, apresentamos um conjunto de soluções com particular interesse em aprendizado classe-incremental [3], isso é, um cenário em que os novos dados a serem incluídos no modelo podem conter rótulos de classes ainda não vistas no modelo atual.

A Floresta de Caminhos Ótimos (em inglês OPF, Optimum-Path Forest) [4] é um método de aprendizado que utiliza teoria dos grafos para construir um modelo baseado em árvores de caminhos ótimos (OPTs, Optimum-Path Trees) para representar classes ou grupos num espaço de características. O classificador derivado deste método tem sido utilizado em uma variedade de aplicações [5], [6]. O seu algoritmo de treinamento possui complexidade quadrática no número de instâncias; assim, nos últimos anos foram desenvolvidas abordagens para aumentar a velocidade do treinamento e classificação [7].

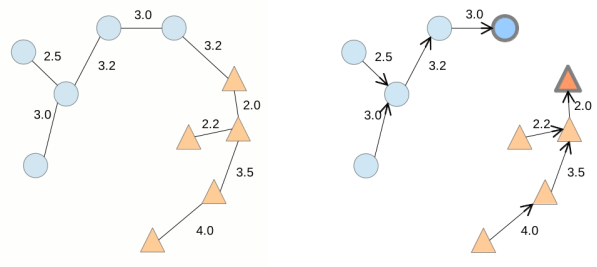
Apresentamos um conjunto de algoritmos para atualização de florestas de caminhos ótimos de maneira incremental [8]. Reportamos resultados considerando o surgimento de novas classes e demonstramos o tempo de execução reduzido do algoritmo para a manutenção do modelo, com complexidade linear. Apesar do modelo criado não ser idêntico ao OPF original devido ao processo de inclusão de novas classes, este mantem as propriedades das florestas de caminhos ótimos inserindo novas instâncias em tempo linear. Comparamos a performance desse método com os classificadores Support Vector Machine (SVM) e k-NN, de maneira a validar sua efetividade e aplicabilidade em diversas bases de dados.

II. Método

A. Classificador Class-Incremental Optimum-Path Forest

O algoritmo Class-Incremental Optimum-Path Forest (OPF-CI) é uma extensão do classificador OPF Incremental [9], que por sua vez é uma extensão do classificador Optimum-Path Forest (OPF) [7]. Em problemas supervisionados, o OPF interpreta cada instância de treinamento como um vértice em um grafo completo, em que os pesos das arestas são dados por alguma função de distância ou dissimilaridade entre os vetores de características. O procedimento inicia obtendo uma árvore geradora mínima (MST – minimum spanning tree) a partir do grafo completo. Então, cada aresta da árvore conectando instâncias com diferentes rótulos são removidas do grafo, e os vértices incidentes nessas arestas se tornam *protótipos*, ou raízes de uma árvore. Cada instância não-protótipo no grafo é conquistada pelo protótipo que oferece o caminho ótimo até a raiz (ou seja, a soma dos pesos da instância até o protótipo), formando assim uma floresta de caminhos ótimos. Esse conceito é demonstrado na Figura 1. O procedimento de treinamento completo possui complexidade $O(n^2)$, onde n é o número de instâncias de treinamento. Maiores informações

sobre esse método em sua versão rápida e conforme utilizada nesse artigo podem ser encontradas em [7].



(a) Construindo a MST (b) Formando as árvores

Figura 1. Treinamento OPF

O classificador incremental OPF-I adiciona uma nova instância rotulada a um modelo OPF pré-existente. Essa inclusão é feita em tempo linear, isto é, $O(n+1)$, sendo n o número de instâncias no modelo pré-existente. O procedimento inicia classificando a nova instância utilizando o modelo atual. Esse passo determina o predecessor da nova instância, ou seja, o vértice do sub-grafo ao qual essa instância foi conectada; então, a inclusão considera um dos seguintes casos:

- Se o predecessor não é um protótipo e esse possui o mesmo rótulo da nova instância, então a nova instância será um **novo vértice em uma árvore existente**; como a árvore é uma MST, utilizamos uma adaptação do método de Chin e Houck para inserir esse novo vértice na árvore em tempo linear [10];
- Se o predecessor é um protótipo e pertence à mesma classe da nova instância, então é possível que essa possa ser um novo protótipo. Um algoritmo de reconquista é executado de forma a verificar se a nova instância irá se tornar um **novo protótipo em uma árvore existente** ou apenas ser incluída conforme o item 1;
- Se o predecessor é de classe diferente da nova instância, então o elemento sendo inserido passa a ser o **protótipo de uma nova árvore**, e caso o predecessor não seja um protótipo ele também se torna um protótipo, causando o particionamento da árvore atual em 2 sub-árvores, para as quais é executado o algoritmo de reconquista.

O terceiro item acima é o responsável pelo aspecto classe incremental do algoritmo, pois permite que um elemento de uma classe ainda não vista forme uma nova árvore no modelo. Para a criação do modelo inicial temos duas opções: caso tenhamos disponíveis exemplos de apenas uma classe, criamos uma floresta de caminhos ótimos de uma classe utilizando o método não-supervisionado OPF Clustering [11] que contém múltiplas árvores de caminhos ótimos porém todas de uma classe. Caso tenhamos exemplos disponíveis de mais de uma classe, então o OPF supervisionado é utilizado para a criação do modelo inicial [7].

Observamos no entanto, que o modelo atualização não é igual ao modelo retreinado. Isso porque no OPF tradicional uma dada instância pode ser conquistada por uma árvore de rótulo diferente. No nosso algoritmo, forçamos que essa instância seja uma nova árvore para permitir que o modelo atualizado contenha a informação da nova classe, ainda não vista. No entanto, demonstraremos a seguir que essa diferença no modelo não afeta a performance da classificação.

Todos os algoritmos utilizados estão descritos com mais detalhes em [9]. A função de inserção em árvores mínimas utilizada é uma versão adaptada do algoritmo de atualização de MSTs proposto por Chin e Houck [10]. Esta adaptação é descrita no Algoritmo 1.

Algorithm 1 Inserção em MST - OPF-I

Require: T é o grafo; z é a nova instância; r é qualquer vértice na árvore; t é uma variável global e é a maior aresta no caminho entre w e z, enquanto m é a maior aresta entre r e z.

```
1: mark r "old"
2: m ← (r, z)
3: for each vertex w adjacent to r do
4:    if w marked "new" then
5:       insertOPT(w, z, T)  // chamada recursiva
6:       k ← maximum between t and (w, r)
7:       h ← mininum between t and (w, r)
8:       T gets the edge h
9:       if cost of k < cost of m then
10:         m ← k
11:      end if
12:   end if
13: end for
14: t ← m
15: return T
```

III. Análise de Complexidade

A complexidade de inserir um novo elemento no modelo utilizando o OPF-CI é $\mathcal{O}(n)$ para n exemplos no modelo atual, conforme prova descrita em [9]. Seja z um novo exemplo conquistado pelo vértice r por uma das árvores de caminhos ótimos do modelo atual; então temos que:

Proof: Após executar a linha 5 do Algoritmo 1, m e t são as maiores arestas no caminho desde r até z, e desde w (primeiro vértice adjacente a r) a z, respectivamente. Considerando que os vértices estão numerados na ordem em que eles completam a chamada ao algoritmo de inserção, então podemos demonstrar por indução que a propriedade da OPT é mantida em tempo linear.

Passo base: seja w' o primeiro nó a completar a chamada recursiva, esse deverá ser um nó folha. Então t será a aresta (w', z) que é a única conectando w' a z. Se r' incide em w', é possível ver que $m = (r', z)$ tanto antes quanto depois da chamada a insertOPT(w').

Passo indutivo: ao executar a linha 5, i.e. insertOPT(w), se w é uma folha, novamente as linhas 5 e 9 serão puladas, e $t = (w, z)$. Do contrário, seja x o vértice incidente a w o qual é o último a ser considerado na chamada insertOPT(w). Por hipótese indutiva, após executar insertOPT(x), temos que m e t são as maiores arestas nos caminhos desde w e x até z, respectivamente. Podemos mostrar que em todos os casos t será a maior aresta no caminho entre w e z. De forma similar, m é a maior aresta no caminho entre r e z.

Tabela I
DATASETS UTILIZADOS

Dataset	Tipo	Atributos	# Classes	# Observ.
Cone-Torus	Sintético	2	3	400
L3	Sintético	2	3	1000
NTL	Real	8	2	4952
SpamBase	Real	56	2	4601
MPEG7-BAS	Real	180	70	1400
Produce	Real	64	14	1400

Ainda, nas linhas 6 a 9, a maior aresta entre m, (w, r) e t é excluída, então m e T (i.e. a MST) são atualizados. Como no máximo $n-1$ arestas são excluídas, sendo cada uma delas a maior aresta em um ciclo T permanecerá uma MST.

Note que insertOPT(r) realiza $(n-1)$ chamadas recursivas na linha 5, e por isso as linhas $1, 2$, 6–10 e 14 são executadas n vezes. As linhas 3 e 4 contam cada aresta no máximo duas vezes e como essa contagem é proporcional ao tamanho da lista de adjacências, executam então $2(n-1)$ vezes. Assim, o Algoritmo 1 tem complexidade $\mathcal{O}(n)$. ■

IV. EXPERIMENTOS

A. Datasets e Reproducibilidade

O código e datasets não disponíveis publicamente podem ser encontrados em uma página web[1]. Os experimentos foram conduzidos utilizando 6 datasets: 2 sintéticos e 4 reais conforme detalhes na Tabela I.

B. Configuração experimental

Os experimentos foram projetados para testar os seguintes conceitos: primeiro o de que o OPF-CI é capaz de aprender novas classes incrementalmente; segundo de que este é robusto à diferentes ordenações em que os novos dados aparecem e são incluídos no modelo; e, finalmente, para comparar sua acurácia e tempo de execução com os classificadores SVM e kNN. Acurácias balanceadas foram computadas de forma a melhor comportar datasets com classes desbalanceadas.

Cada experimento utiliza como modelo inicial dados de apenas uma classe. A partir desse modelo de uma classe, executamos o algoritmo OPF-CI. A partição em conjuntos de treinamento e teste foi repetida 10 vezes de maneira aleatória com distribuição uniforme e mantendo a proporção entre as classes, em modelo hold-out.

Os seguintes passos são seguidos para os experimentos com cada base de dados:

1) **Partição treinamento/teste** com 50% de instâncias cada: S é o conjunto de treinamento (esse será utilizado para criar o conjunto inicial e os incrementos), e T o conjunto de testes.
2) **Criação do conjunto inicial e incrementos a partir de** S: S_0 é o conjunto de treinamento inicial contendo 50% dos exemplos da classe inicial j em S (as outras classes não aparecem no conjunto inicial), e I contém as demais instâncias;

[1]http://www.icmc.usp.br/~moacir/paper/16wuwopfci.html

Tabela II
ACURÁCIA E TEMPO DE EXECUÇÃO PARA A BASE DE DADOS NTL

		Iterações					
	Classe inicial	1st	25%	50%	75%	100%	Alvo
OPF-CI	1	52.2	68.6	74.7	79.7	80.9	81.5
	2	80.7	81.5	81.4	82.3	80.9	
SVM	1	51.5	63.4	73.4	80.4	80.0	80.0
	2	79.8	77.4	77.4	80.1	80.0	
3NN	1	50.6	51.5	64.4	70.7	73.9	73.9
	2	71.6	72.6	71.1	73.6	73.9	

3) **Modelo inicial usando** S_0 construindo uma floresta de caminhos ótimos com árvores de uma classe apenas utilizando o algoritmo OPF clustering [11];
4) **Criação dos incrementos**: I é particionado em 20 incrementos S_i, $i = 1..20$, todos uniformemente distribuídos, mantendo a proporção entre as classes; uma exceção é o dataset MPEG7 dataset, particionado em 5 incrementos por ser um dataset pequeno;
5) **Atualização incremental**: o modelo é atualizado, incluindo cada conjunto S_i com $i = 1..20$ e avaliando no conjunto de testes T.

V. RESULTADOS E DISCUSSÃO

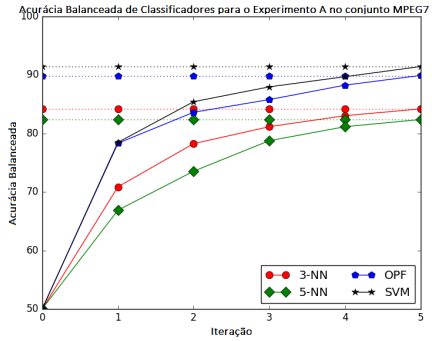
Reportamos a acurácia obtida e o tempo de execução para o treinamento e classificação utilizando cada método. Por limitações de espaço, apenas os resultados mais relevantes são mostrados em detalhes. O classificador SVM com kernel RBF teve parâmetros definidos por um grid search no conjunto S e o tempo de execução para essa busca não foi incluído nos resultados. Isso foi necessário pois seria muito custoso computar parâmetros para cada incremento. Assim, devemos interpretar os resultados do algoritmo SVM como um limite superior.

Na Figura 2 e Figura 3 mostramos os resultados para os datasets MPEG7 e Cone-Torus, respectivamente, comparando a acurácia por incremento como uma linha sólida com a acurácia do treinamento com o conjunto completo S, a qual chamamos de acurácia alvo. Nas Tabelas II e III, a acurácia sobre a porcentagem de incrementos é reportada de forma a demonstrar a robustez do método à variação na ordem das classes como classes iniciais.

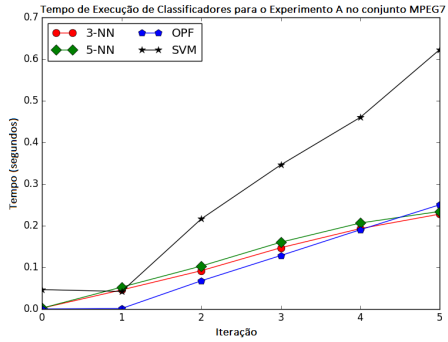
Os resultados apontam que o OPF-CI alcança a acurácia alvo, demonstrando sua capacidade classe-incremental. Ambas acurácia e tempo total utilizando o OPF-CI são comparáveis com os demais algoritmos, ainda que ligeiramente mais lenta do que o kNN (note, no entanto que o algoritmo kNN não possui treinamento). O tempo para treinamento com o SVM variou entre os datasets. Nas Tabelas II e III, notamos que o OPF-CI alcança de forma consistente a acurácia alvo, independente da classe inicial, confirmando sua robustez com relação à ordem.

VI. CONCLUSÕES

Os algoritmos propostos foram capazes de atualizar com sucesso modelos OPF para os diversos datasets estudados,

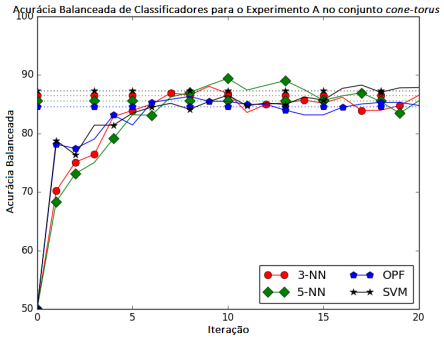


(a) Acurácia

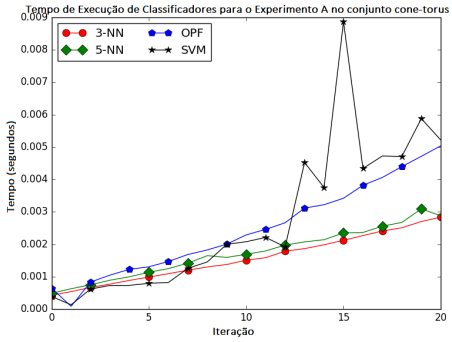


(b) Tempo

Figura 2. Acurácia e tempo de execução MPEG7



(a) Acurácia



(b) Tempo

Figura 3. Acurácia e tempo de execução Cone-Torus

Tabela III
ACURÁCIA E TEMPO DE EXECUÇÃO PARA A BASE DE DADOS PRODUCE (14-CLASSES) PARA UMA AMOSTRA DE CLASSES COMO MODELO INICIAL

		Iterações					
	Classe inicial	1st	25%	50%	75%	100%	Alvo
OPF-CI	1	70.9	88.2	91.1	94.1	94.7	94.4
	2	73.5	89.4	92.6	94.0	94.7	
	13	70.9	89.4	92.2	94.2	94.7	
	14	71.2	89.3	92.5	94.4	94.7	
SVM	1	71.5	91.4	93.8	95.5	96.8	96.6
	2	72.6	90.7	93.5	95.4	96.8	
	13	73.6	90.8	93.6	95.5	96.7	
	14	72.6	90.8	93.8	95.5	96.7	
3NN	1	58.7	83.0	87.5	91.2	93.0	93.0
	2	60.8	84.9	89.7	92.0	93.0	
	13	68.9	84.8	90.1	92.0	93.0	
	14	62.7	83.9	90.8	91.2	93.0	

possuindo prova de manutenção da estrutura das árvores de caminhos ótimos e de inclusão em tempo linear para cada instância. Apesar do modelo atualizado não ser idêntico àquele produzido pelo retreinamento, a abordagem proposta tem apelo do ponto de vista de suas capacidade incrementais, sem perda de acurácia conforme demonstrado nos experimentos.

Trabalhos futuros podem explorar a abordagem proposta em conjunto com outros métodos de aceleração do treinamento e classificação OPF, bem como adaptados para uso com OPF não supervisionado entre outras aplicações.

AGRADECIMENTOS

Os autores gostariam de agradecer às bolsas FAPESP 15/13504-4, 14/04889-5, e 15/24652-2.

REFERÊNCIAS

[1] X. Geng and K. Smith-Miles, "Incremental learning," *Encyclopedia of Biometrics*, pp. 912–917, 2015.

[2] I. Kuzborskij, F. Orabona, and B. Caputo, "From n to n+ 1: Multiclass transfer incremental learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3358–3365.

[3] B.-F. Zhang, J.-S. Su, and X. Xu, "A class-incremental learning method for multi-class support vector machines in text classification," in *IEEE International Conference on Machine Learning and Cybernetics*, 2006, pp. 2581–2585.

[4] J. P. Papa, A. X. Falcão, and C. T. Suzuki, "Supervised pattern classification based on optimum-path forest," *International Journal of Imaging Systems and Technology*, vol. 19, no. 2, pp. 120–131, 2009.

[5] M. Ponti and C. T. Picon, "Color description of low resolution images using fast bitwise quantization and border-interior classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 1399–1403.

[6] M. Ponti Jr and I. Rossi, "Ensembles of optimum-path forest classifiers using input data manipulation and undersampling," in *Multiple Classifier Systems*. Springer, 2013, pp. 236–246.

[7] J. P. Papa, A. X. Falcão, V. H. C. De Albuquerque, and J. M. R. Tavares, "Efficient supervised optimum-path forest classification for large datasets," *Pattern Recognition*, vol. 45, no. 1, pp. 512–520, 2012.

[8] M. Riva, M. Ponti, and T. de Campos, "One-class to multi-class model update using the class-incremental optimum-path forest classifier," in *European Conference on Artificial Intelligence (ECAI)*, 2016.

[9] M. Ponti and M. Riva, "An incremental linear-time learning algorithm for the optimum-path forest classifier," *arxiv Preprint*, 2016.

[10] F. Chin and D. Houck, "Algorithms for updating minimal spanning trees," *Journal of Computer and System Sciences*, vol. 16, no. 3, pp. 333–344, 1978.

[11] L. M. Rocha, F. A. M. Cappabianco, and A. X. Falcão, "Data clustering as an optimum-path forest problem with applications in image analysis," *International Journal of Imaging Systems and Technology*, vol. 19, no. 2, pp. 50–68, 2009.