Um Sistema de Fusão de Classificadores Aplicado à Fenologia

Victor Lúcio, Fabio A. Faria e Jurandy Almeida
Instituto de Ciência e Tecnologia, Universidade Federal de São Paulo – UNIFESP
12247-014, São José dos Campos, SP – Brazil
Email: {victor.lucio, ffaria, jurandy.almeida}@unifesp.br

***Resumo*—Fenologia, o estudo de eventos recorrentes de ciclos de vida de plantas e sua relação com o meio ambiente, é uma disciplina chave em pesquisas sobre mudanças climáticas. O monitoramento da fenologia das plantas exige uma abordagem multidisciplinar, combinando pesquisas em computação e biologia. Nesse contexto, câmeras digitais podem ser usadas como sensores de imagem de múltiplos canais para monitorar vegetações e prover medidas precisas das mudanças do ciclo de vida das plantas ao longo do tempo. Um requisito fundamental nessa tarefa é o reconhecimento de padrões na imagem. Neste trabalho, foi desenvolvido um sistema de fusão de classificadores para auxiliar especialistas na área de fenologia a identificar padrões de diferentes espécies de plantas em imagens.**

***Abstract*—Phenology, the study of recurrent plant life cicles events and its relation to the environment is a key discipline for climate change research. The plant phenology monitoring requires a multidisciplinary approach, combining research on computer science and biology. In this context, digital cameras can be used as multi-channel imaging sensors to monitor vegetations and provide accurate measures of plant life cycle changes over time. A fundamental requirement in this task is pattern recognition in images. In this work, it was developed a classifier fusion system for supporting experts in the phenology area to identify patterns from different plant species in images.**

I. Introdução

A observação direta dos ciclos das plantas impõe uma série de dificuldades para os observadores, especialmente nos ambientes tropicais, em que não há nenhuma mudança abrupta ou marcante entre estações. Além disso, a observação em campo é geralmente conduzida por mais de um observador, o que leva a um viés em relação à percepção da mudança da planta [1], [2].

Para aumentar a gama locais e espécies de estudo e a precisão das observações fenológicas, câmeras digitais têm sido aplicadas com sucesso, como sensores de imagem de múltiplos canais, fornecendo medidas para estimar mudanças em eventos fenológicos, como brotamento e senescência [3].

Nesse contexto, a computação é uma grande aliada dos especialistas em fenologia, fornecendo-lhes ferramentas para auxiliar nessa tarefa [4], [5]. Na literatura existem diversos trabalhos que mostram a grande sinergia do uso de técnicas de processamento de imagens e aprendizado de máquina para resolver problemas em diferentes área de aplicações [6]–[8].

Segundo o teorema "*No Free Lunch*" [9], não existe uma técnica de aprendizagem ótima capaz de obter os melhores resultados para todo e qualquer domínio de aplicação. Portanto, uma alternativa para esse grande desafio é a utilização de métodos para fusão de informação, seja em nível de características (*early fusion*), ou fusão em nível de decisão (*late fusion*). Estas duas formas de fusão não são as únicas existentes, porém, as mais conhecidas na literatura [10].

A proposta desse trabalho foi implementar um sistema de fusão de classificadores para reconhecimento de padrões em estudos de fenologia.

II. Conceitos Básicos

A. Tipos de Fusão

A fusão de classificadores pode ser caracterizada de algumas maneiras diferentes, que estão descritas abaixo. O sistema implementado e documentado por meio deste artigo no entanto baseia-se no modelo *"Hybrid Fusion"*, que leva em consideração os outros dois tipos de fusão.

1) Early Fusion: Também conhecida como *feature-level fusion*, se refere ao proceso de fusão que ocorre antes da etapa de aprendizagem (*learning technique*). Dada uma imagem, realiza-se a extração de características com diferentes descritores de imagens e, então, combina-se as diferentes características resultando uma única característica final (concatenação de vetores de características).

2) Late Fusion: Também conhecida como *decision-level fusion* refere-se ao processo de fusão que ocorre depois da etapa de aprendizagem (e.g., Adaboost [11] e Bagging [12]). Dado uma imagem, usa-se diferentes técnicas de aprendizagem (*learning techniques*) resultando em resultados individuais dessas técnicas e, então, combina-se essas diferentes decisões utilizando algum critério (e.g., votação majoritária).

3) Hybrid Fusion: Refere-se ao processo de fusão envolvendo ambos processos de fusão explicados anteriormente (*early* e *late*).

B. Arcabouço de Fusão e Seleção de Classificadores

A Figura 1 mostra o arcabouço de fusão e seleção de classificadores proposto por Faria et al. [13], que foi utilizado como base deste trabalho.

Sejam $\mathcal{L}$ um conjunto de técnicas de aprendizagem (e.g., *Decision Tree*, *Naive Bayes*, e k-nearest neighbors - kNN) e $\mathcal{F}$ um conjunto de descritores de imagens (e.g., Histograma de cor). Suponha que classificadores são criados pela combinação de cada técnica de aprendizagem com um descritor de imagem.

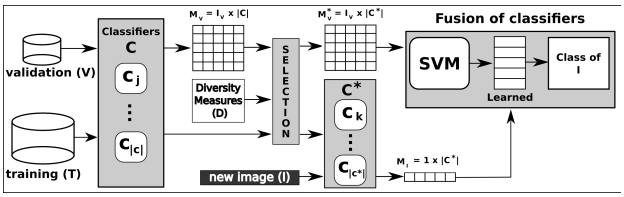

Figura 1. Arcabouço para seleção e fusão proposto por Faria et al. [13]. O arcabouço seleciona os mais discriminantes classificadores para ser combinados utilizando uma técnica de fusão.

Por exemplo, três classificadores podem ser criados combinando as técnicas de aprendizagem Árvore de decisão, Naive Bayes e kNN com o descritor Histograma de cor. Seja $\mathcal{C}$ o conjunto de classificadores criados da combinação, em que $|\mathcal{C}| = |\mathcal{L}| \times |\mathcal{F}|$.

Seja $\mathcal{S}$ ser um conjunto de dados (e.g., imagens), em que a classe de $s_i \in \mathcal{S}$ $(1 < i \leq |\mathcal{S}|)$ é conhecida. O conjunto $\mathcal{S}$ é usado para construir os conjunto de treinamento (T) e validação (V), em que $T \cup V = \mathcal{S}$ e $T \cap V = \emptyset$.

Inicialmente, todos os classificadores $c_j \in \mathcal{C}$ $(1 < j \leq |\mathcal{C}|)$ são treinados no conjunto T. Depois, realiza-se a classificação utilizando-se cada um dos classificadores treinados no conjunto de validação V, resultando em uma matriz M_V, em que $|M_V| = |V| \times |\mathcal{C}|$.

Em seguida, M_V é usada para selecionar o conjunto $\mathcal{C}^* \subset \mathcal{C}$ de classificadores que são os melhores candidatos para serem combinados. Em nossa abordagem, medidas de diversidade [14] são utilizadas para determinar $\mathcal{C}^*$. Note que a nova matriz $M_V^* \subset M_V$ pode ser criada pelos classificadores selecionados $\mathcal{C}^*$.

Dado um novo dado (e.g., imagem) I, é utilizado cada classificador $c_k \in \mathcal{C}^*$ $(1 < k \leq |\mathcal{C}^*|)$ para determinar a classe de I. Estas k saídas são usadas como entrada para a técnica de fusão (e.g., votação majoritária, *support vector machine* – SVM) que definirá a classe I. No caso de se usar uma técnica de fusão na qual seja necessário etapa de treinamento (e.g., SVM), a matriz M_V^* é utilizada.

III. Implementação

A implementação do sistema de fusão de classificadores foi desenvolvida usando a linguagem de programação JAVA e a biblioteca de aprendizado de máquina WEKA [15]. Ao longo do processo de desenvolvimento mais de uma versão foi criada, levando sempre em consideração a modularidade e a facilidade de uso do sistema, tornando-o robusto e utilizável não só para imagens, mas para qualquer tipo de instância.

A. Análise da Biblioteca de Classificação WEKA

Ao decidir pelo uso de uma biblioteca de classificação é necessário entender como ela funciona, se a mesma traz os recursos necessários para as intenções desejadas e sua capacidade de modularização para poder aproveitar ao máximo as ferramentas já implementadas, como por exemplo, classificadores tradicionais da literatura e métodos de avaliação e validação de classificadores. Foi analisada a estrutura de classes de classificação do WEKA, que traz "caminhos" para diferentes tipos de classificadores, incluindo meta-classificadores, por exemplo, o **Stacking**. A biblioteca tem grande capacidade de modularização e utiliza interfaces simples que são implementadas por classes abstratas de diferentes níveis. Isso se faz necessário pela diversidade de tipos de classificadores, por exemplo, os meta-classificadores utilizam a classe abstrata **MultipleClassifiersCombiner** ou classes em níveis abaixo (classes "filho"). Apesar de vários níveis de classes abstratas, todos os classificadores são derivados da classe **AbstractClassifier**, a classe que as ferramentas do WEKA recebem como o parâmetro e faz referência a um classificador compatível, ou seja, estendendo esta classe é possível construir um novo classificador compatível com toda a biblioteca.

B. Implementação do Classificador

A classe **MultipleClassifierCombiner** do WEKA foi estendida, sendo criada então a classe **MCSClassifier**, a classe principal do sistema de classificação deste trabalho. O primeiro desafio se deu pela maneira como classificadores **AbstractClassifier** lidam com instâncias de treino. Este trabalho demanda que o treino seja realizado com instancias descritas de diferentes formas, ou seja, diversas maneiras de descrever a mesma imagem (Múltiplas características), mas a entrada de treino da classe abstrata não traz essa facilidade.

C. Lidando com Diferentes Características

Algumas alternativas foram avaliadas nesse ponto:

1) Tranformar imagens em instâncias, extraindo as caracteristicas dentro do classificador.
2) Indicar nas instâncias o local da memória secundária onde se encontram as imagens para extrair características dentro do classificador.
3) Criar uma nova classe que representa instâncias múltiplas e estende a classe que representa instâncias no WEKA.

A primeira alternativa foi uma solução que traria um gasto de memória muito grande, pois baseava-se na representação de imagens, as quais gastam muito espaço. Além disso, imagens podem ter diferentes tamanhos, comprometendo a funcionalidade do sistema. A segunda alternativa corrige o problema de memória da primeira, porém, também priva a capacidade de classificar instâncias de diferentes tipos, tornando o classificador exclusivo a imagens. A terceira alternativa foi a melhor avaliada, pois as características são normalmente mais leves do que a imagens, resolvendo o problema de memória, além disso, o fato de não precisar extrair as características da imagem gera a possibilidade de deixar o usuário escolher livremente a maneira como descrever as características.

D. A Evolução da Classe *MCSClassifier*

Inicialmente, a implementação da classe principal do classificador não envolvia seleção. O foco era apenas lidar com a entrada de classificadores levando em consideração que haveria uma seleção dos mesmos e de características das instâncias,

sendo necessário nesse caso criar uma matriz de validação para treinar um classificador de fusão. As primeiras versões utilizavam classificadores padrão definidos na implementação das mesmas. O usuário deveria utilizar um arranjo com as opções para selecioná-los. O grande defeito desse tipo de implementação é a modularidade, pois o usuário estaria preso a opções pré-existentes, não podendo, por exemplo, criar o próprio classificador e utilizá-lo no **MCSClassifier**. As versões mais recentes do classificador recebem em seu construtor um arranjo de classificadores da classe **AbstractClassifier**, o parâmetro de divisão entre validação e treino, o classificador de fusão (pode ser nulo, definindo a classe da instância testada por votação entre os classificadores selecionados) e o método de seleção. A última versão do **MCSClassifier** foi esquematizada na Figura 2 e o conjunto de validação e treino foram concatenados para seguir o padrão de entrada de treino dos classificadores da biblioteca WEKA.

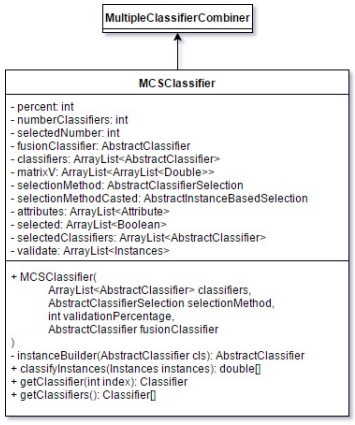

Figura 2. Estrutura da classe **MCSClassifier**.

E. Medidas de Diversidade entre Classificadores

Os valores das medidas de diversidade podem ser ordenados de maneira diferente em relação ao tipo de diversidade. Por isso, foi criada uma classe abstrata para descrever classes de medidas que também serve como comparador (isto é, implementa a interface **Comparator** do pacote "java.util" da linguagem JAVA).

F. Classes para Seleção

Inicialmente, a estrutura de seleção seria exclusiva para a seleção dos classificadores no **MCSClassifier**. Porém, após verificar que vários sistemas dentro do projeto como um todo utilizam seleção, tornou-se interessante criar uma estrutura de classes de seleção genérica, servindo para diversos propósitos deste trabalho.

A interface **Selection Method** é apenas composta por um método de seleção que retorna um arranjo booleano (verdadeiro para instância selecionada ou falso para instância não selecionada). A partir dela, foram desenvolvidas duas classes abstratas, a **AbstractClassifierSelection** e a **AbstractGenericSelection**, com o intuito de criar um nível de especialização de seleção. A primeira (**AbstractClassifierSelection**) é utilizada para implementação de seletores de classificadores genéricos. Foi criada mais uma classe abstrata em um nível abaixo para representar as classes de seleção de classificadores que necessitam de instâncias para selecioná-los, que é o caso do método de seleção **Concensus** [16]. A segunda (**AbstractGenericSelection**) é utilizada para seleção de qualquer tipo de instância comparável, que neste trabalho foi utilizado para o desenvolvimento de classes que selecionam instâncias acima de um limiar ou que ordenam os dados e selecionam os "t" (variável de entrada) melhores. No esquemático UML, há mais uma classe abstrata (**AbstractSelectionMetrics**), a qual é na verdade um classe feita para medidas de um conjunto de classificadores (não uma especialização de seleção) e que apenas implementa a interface para herdar o método de seleção. Esta classe é utilizada pela classe **Concensus**.

G. Exemplo de Implementação e Utilização do MCS

```
1  ArrayList<AbstractDiversityMeasure> dm =
2                  new ArrayList<AbstractDiversityMeasure>();
3  dm.add(new QStatistic());
4  dm.add(new DoubleFaultMeasure());
5  dm.add(new DisagreementMeasure());
6  dm.add(new CorrelationCoefficient());
7  dm.add(new InterraterAgreement());
8
9  AverageAccuracyMean metrics = new AverageAccuracyMean();
10
11 Concensus concensus = new Concensus(dm, 100, 6, metrics);
12 SMO svm = new SMO();
13 MCS = new MCSClassifier(classifiers, concensus, 75, svm);
14 MCS.buildClassifier(treino);
15
16 Evaluation eval;
17 eval = new Evaluation(treino);
18 eval.evaluateModel(MCS, teste);
```

IV. Resultados

Os principais testes foram feitos utilizando imagens de 6 classes de espécies de plantas (*A.tomentosum*, *C.brasiliensis*, *M.guianensis*, *M.rubiginosa*, *P.ramiflora* e *P.torta*) em diferentes horas do dia (das 6 às 18 horas) e considerando os três canais de cor (RGB) de cada imagem, totalizando 39 características. O descritor utilizado foi o *RGB chromatic coordinates (RGBcc)* [17], que é considerado um dos mais eficientes para detectar mudanças de cor das folhas. O método de aprendizado utilizado foi o k-NN com parâmetro k=1, como sugerido por Conti et al [18]. Foi realizada a combinação de 39 classificadores treinados com o método k-NN, dentre os quais, $\mathcal{C}^*$ (o valor pode ir de 0 até o número de classificadores gerados pelo **MCSClassifier**, neste caso, 39) foram selecionados e utilizados como dados de entrada para um classificador de fusão SVM (utilizando função de kernel polinomial) em um protocolo de validação cruzada utilizando 5 rodadas (5-folds). Apesar da variação do $\mathcal{C}^*$, também variou-se o parâmetro de corte t (seleciona os t pares de classificadores com maior diversidade). Para avaliar os resultados foram utilizadas classes da própria biblioteca WEKA. O gráfico de acurácia variando $\mathcal{C}^*$ para $t = 100$ pode ser observado na Figura 4. Um teste de destaque foi o de $t = 100$ e $\mathcal{C}^* = 7$, com 86.62% de acurácia média. O MCS se saiu melhor comparado a

abordagens de fusão de classificadores da literatura (*Majority Vote e Bagg*) em testes comparativos para problemas utilizando plantas. No quesito acurácia média balanceada o segundo melhor classificador (*Majority Vote*) teve uma diferença em torno de 5% abaixo em relação ao MCS.

Para analisar a eficácia do método de seleção **Concensus**, foram analisados dois gráficos: (1) o histograma (Figura 5) com votos contabilizados através do ranqueamento de pares de classificadores com maior diversidade e (2) a acurácia média balanceada (Figura 3) de cada classificador utilizando parâmetros $t = 100$ e $\mathcal{C}^* = 6$. As barras de cor vermelha simbolizam a seleção classificador para a fase de fusão.

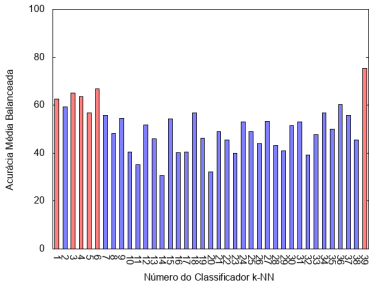

Figura 3. Acurácia Média Balanceada de cada classificador.

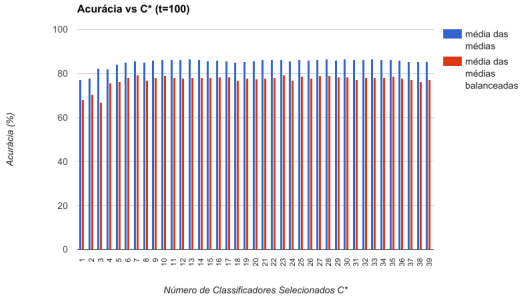

Figura 4. Gráfico: Acurácia versus $\mathcal{C}^*$ para $t = 100$.

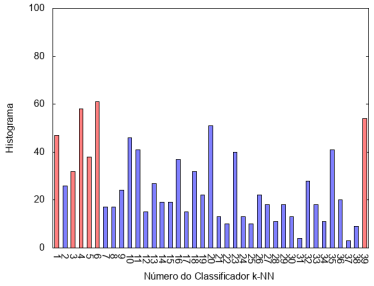

Figura 5. Histograma de votos de diversidade.

V. Conclusões

A implementação do sistema garantiu bons resultados de acurácia para tarefa de reconhecimento de espécies de plantas e está de acordo com a literatura. Além disso, mostrou ser uma implementação modular compatível a qualquer classificador e ferramenta da biblioteca de mineração de dados WEKA, com uma engenharia de software consistente que permite a expansão do trabalho e utilização em diferentes tarefas de aprendizado de máquina.

Agradecimentos

Os autores agradecem aos pesquisadores Dra. Leonor Patrícia Cerdeira Morellato, Dr. Ricardo da Silva Torres e Bruna de Costa Alberton pela assistência com os dados utilizados neste trabalho; e também às agências de fomento FAPESP (processo 2016/06441-7) e CNPq (processo 423228/2016-1) pelo apoio financeiro.

Referências

[1] L. P. C. Morellato, M. G. G. Camargo, F. F. D'Eça Neves, B. G. Luize, A. Mantovani, and I. L. Hudson, "The influence of sampling method, sample size, and frequency of observations on plant phenological patterns and interpretation in tropical forest trees," in *Phenological Research*, I. L. Hudson and M. R. Keatley, Eds. Springer, 2010, chapter 5, pp. 99–121.

[2] C. A. Polgar and R. B. Primack, "Leaf-out phenology of temperate woody plants: From trees to ecosystems," *New Phytologist*, vol. 191, pp. 926–941, 2011.

[3] B. Alberton, J. Almeida, R. Henneken, R. da S. Torres, A. Menzel, and L. P. C. Morellato, "Using phenological cameras to track the green up in a cerrado savanna and its on-the-ground validation," *Ecological Informatics*, vol. 19, pp. 62–70, 2014.

[4] J. Almeida, J. A. Santos, B. Alberton, L. P. C. Morellato, and R. S. Torres, "Phenological visual rhythms: Compact representations for fine-grained plant species identification," *Pattern Recognition Letters*, vol. 81, pp. 90–100, 2016.

[5] J. Almeida, D. C. G. Pedronette, B. Alberton, L. P. C. Morellato, and R. S. Torres, "Unsupervised distance learning for plant species identification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 12, pp. 5325–5338, 2016.

[6] J. Almeida, J. A. dos Santos, B. Alberton, R. da S. Torres, and L. P. C. Morellato, "Applying machine learning based on multiscale classifiers to detect remote phenology patterns in cerrado savanna trees," *Ecological Informatics*, vol. 23, pp. 49–61, 2014.

[7] F. A. Faria, J. Almeida, B. Alberton, L. P. C. Morellato, A. Rocha, and R. S. Torres, "Time series-based classifier fusion for fine-grained plant species recognition," *Pattern Recognition Letters*, vol. 81, pp. 101–109, 2016.

[8] F. A. Faria, J. Almeida, B. Alberton, L. P. C. Morellato, and R. S. Torres, "Fusion of time series representations for plant recognition in phenology studies," *Pattern Recognition Letters*, vol. 83, Part 2, pp. 205–214, 2016.

[9] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Computation*, vol. 8, no. 7, pp. 1341–1390, 1996.

[10] F. Faria, D. Pedronette, J. dos Santos, A. Rocha, and R. Torres, "Rank aggregation for pattern classifier selection in remote sensing images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 4, pp. 1103–1115, 2014.

[11] R. E. Schapire, "A brief introduction to boosting," in *International Joint Conference on Artificial Intelligence (IJCAI'99)*, 1999, pp. 1401–1406.

[12] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[13] F. A. Faria, J. A. dos Santos, A. Rocha, and R. da S. Torres, "A framework for selection and fusion of pattern classifiers in multimedia recognition," *Pattern Recognition Letters*, vol. 39, no. 0, pp. 52–64, 2014.

[14] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[15] I. Witten, E. Frank, M. Hall, and C. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, 2016.

[16] F. Faria, "A framework for pattern classifier selection and fusion," Ph.D. dissertation, Instituto de Computação, Unicamp, Campinas, Brazil, 2014.

[17] J. Almeida, J. A. Santos, W. O. Miranda, B. Alberton, L. P. C. Morellato, and R. S. Torres, "Deriving vegetation indices for phenology analysis using genetic programming," *Ecological Informatics*, vol. 26, pp. 61–69, Mar. 2015.

[18] J. C. Conti, F. A. Faria, J. Almeida, B. Alberton, L. P. C. Morellato, L. Camolesi Jr., and R. da S. Torres, "Evaluation of time series distance functions in the task of detecting remote phenology patterns," in *IEEE International Conference on Pattern Recognition (ICPR'14)*, 2014, pp. 3126–3131.