Universidade Federal de São Paulo - UNIFESP
Instituto de Ciência e Tecnologia - ICT

Data Augmentation via aprendizado semissupervisionado baseado em grafos

Orientadora: Prof[a] Dr[a] Lilian Berton
Co-orientador: Dr. Otávio A. B. Penatti
Aluno: Willian Dihanster Gomes de Oliveira

Projeto Iniciação Científica 2017/2018

Relatório referente ao período de 01 de agosto de 2017 a 31 de julho 2018

Assinatura da Pesquisadora Responsável:

_____

Dr[a] Lilian Berton

Resumo

O desempenho dos algoritmos de aprendizado supervisionado melhora com o tamanho do conjunto de dados usado para treinar o classificador. Em aplicações contendo uma quantidade limitada de dados rotulados, mas uma grande quantidade de dados não rotulados, esquemas de aumento de dados (*data augmentation*) são utilizados para criar conjuntos de dados maiores (aumentados) incorporando-se exemplos não rotulados no conjunto de dados rotulado. Este conjunto de dados aumentados é usado para treinar o classificador (por exemplo, uma rede neural), na esperança de que os dados introduzidos possam reduzir o erro de classificação do classificador construído. Em geral, a construção de esquemas de aumento de dados que resultam em algoritmos simples e rápidos é uma tarefa não trivial, uma vez que as estratégias de sucesso variam muito com a natureza dos dados presentes em diferentes aplicações. Nesse trabalho, objetivamos estudar duas abordagens para o aumento dos dados: 1) a primeira delas gera amostras através de transformações aplicadas no espaço de dados; 2) a segunda faz uso do aprendizado semissupervisionado baseado em grafos. Os *datasets* considerados serão conjunto de imagens e os algoritmos supervisionados testados serão: *k-Nearest Neighbors* (kNN), *Naive Bayes* (NB), *Decision Tree* (J48), *Multilayer Perceptron with Backpropagation* (MLP) e *Support Vector Machine* (SMO).

1 Introdução

O Aprendizado de Máquina (AM) é uma subárea da Inteligência Artificial que busca construir e modelar sistemas capazes de adquirir conhecimento de forma automática. O sistema de aprendizado é um software computacional que toma decisões baseadas nas experiências acumuladas de soluções de problemas sucedidos anteriormente. O AM se divide em supervisionado, não supervisionado e semissupervisionado.

No aprendizado supervisionado a quantidade de exemplos rotulados disponíveis e a acurácia da classificação estão relacionados. Quanto maior a quantidade de exemplos, maior a capacidade de predição dos algoritmos. No entanto, em situações da vida real possuir bases de dados com uma grande quantidade de exemplos rotulados não é uma tarefa fácil e muitas vezes, custosa. Dessa forma, criou-se o conceito de *Data Augmentation* (DA) que utiliza técnicas computacionais a fim de aumentar a quantidade de exemplos rotulados e assim, tentar melhorar a acurácia da classificação.

Esse projeto busca estudar duas técnicas diferentes de DA, aplicadas na classificação de imagens. Uma delas, expande o conjunto de dados rotulados por meio de transformações em imagens, como rotações, efeitos *blur*, etc. E a outra, expande o conjunto de dados rotulados por meio do aprendizado semissupervisionado baseado em grafos. Dado um pequeno conjunto de dados não-rotulados e uma grande quantidade de dados não rotulados o aprendizado semissupervisionado classifica esses dados por meio de um algoritmo de propagação de rótulos, nesse caso, utilizou-se o *Global and Local Consistency*, obtém-se assim mais dados rotulados que poderão ser utilizados no aprendizado supervisionado.

Os experimentos de classificação supervisionada foram realizados no software livre WEKA, utilizando-se 5 classificadores: kNN, Naive Bayes, Árvores de Decisão - J48, *Suport Vector Machine* - SVM e *Multi Layer Perceptron* - MLP. Ademais, utilizou-se o software livre R e a biblioteca igraph, para um estudo mais detalhado da abordagem semissupervisionada baseada em grafos. A fim de tentar entender e descobrir qual a melhor técnica de DA e se possível, melhorá-la.

2 Aprendizado de Máquina

No Aprendizado de Máquina (AM) usa-se técnicas de indução, para obter uma conclusão sobre um conjunto de exemplos. Ou seja, o computador recebe uma amostra de dados e usa inferência indutiva sobre os elementos. Enquanto que os

humanos usam raciocínio dedutivo para deduzir uma nova informação baseado em informações relacionadas logicamente. [1]

2.1 Hierarquia do Aprendizado

Em AM, os métodos aprendizado podem ser dividido em 4 técnicas. Sendo elas: supervisionado, semissupervisionado, não supervisionado e por reforço, detalhadas nos próximo paragráfos [2]:

No aprendizado supervisionado, sendo o mais comum a classificação, os algoritmos recebem exemplos já rotulados com um atributo classe e aprendem comparando a saída gerada pelo algoritmo com a saída correta para encontrar erros, e então, modificam o modelo para se ajustar as saídas corretas. Ou seja, identificam padrões para classificar ou rotular novos elementos que não faziam parte da amostra inicial.

No aprendizado não supervisionado, muito usado na clusterização, os algoritmos recebem exemplos sem nenhum rótulo e buscam identificar características em comum para tentar prever classes ou cluster (agrupamento) entre os dados.

No aprendizado semissupervisionado os algoritmos fazem uso de dados não rotulado e dados rotulados. Em geral, maioria é não rotulado e por isso, útil quando se exige muito processamento ou é muito custoso adquirir os rótulos dos dados. Funciona de maneira semelhante ao aprendizado supervisionado, sendo comum usar técnicas como classificação, regressão e previsão.

Já o aprendizado por reforço, é amplamente usado na robótica, jogos e navegação. É baseado em tentativa e erro, onde recebe-se recompensas e punições de acordo com as ações tomadas e busca-se aprender as melhores escolhas para maximizar as recompensas em menor tempo. É necessário um agente (tomador de decisões), ambiente (o lugar no qual o agente interage) e ações (coisas que o agente pode fazer no ambiente).

2.2 Conceitos de Aprendizado de Máquina

A seguir são apresentados os principais conceitos relacionados com AM. [3]

- Exemplo: É um dado, registro ou modelo que possui características que vão ser usadas e/ou observadas pelos algoritmos.
- Atributo: É uma característica do exemplo, como cor, tamanho, etc.

- Classe: É uma espécie de atributo especial que rotula o exemplo.
- Conjunto de exemplos: É um conjunto de dados que contém os atributos e seus respectivos valores e em alguns casos, a classe a qual cada exemplo pertence.
- Indutor: É o algoritmo ou programa que usa indução a fim de extrair e definir um classificador a partir de um conjunto de exemplos.
- Bias: Ou viés, é a tendência do classificador de cometer erros de generalização. [4]
- Modo de Aprendizado:
  - Incremental: Quando é possível adicionar novos exemplos ao conjunto de treinamento.
  - Não incremental: Não é possível adicionar novos exemplos.
- Erro: Mede o desempenho de um determinado classificador de acordo com os seus acertos.

$$err(h) = \frac{1}{n} \sum_{i=1}^{n} \|y_i \neq h(x_i)\|$$

  sendo y o rótulo do exemplo e $h(x_i)$ o rótulo predito pelo indutor.
- Acurácia: Mede a precisão do classificador. É dado por:

$$acc(h) = 1 - err(h)$$

- Distribuição de classes: Dada uma classe c_j, a distribuição é dada pela soma de vezes que essa classe faz parte dos exemplos, dividido pelo número total de exemplos do conjunto.

$$distr(c_j) = \frac{1}{n} \sum_{i=1}^{n} \|y_i = h(x_i)\|$$

- Classe Majoritária: É classe que mais aparece no conjunto de dados.
- Classe Minoritária: É classe que menos aparece no conjunto de dados.

- Erro Majoritário: Usado para definir um limite para o erro de um classificador. Dado por:

$$majErr(T) = 1 - \max_{i=1,\ldots,n} dist(c_i)$$

- *Overfitting*: Acontece quando um algoritmo não é capaz de generalizar para amostras diferentes, sendo muito específico e ajustado somente para o conjunto de exemplos inicial.

- *Underfitting*: Quando mesmo com o conjunto de testes e treinamento a taxa de erro ainda é muito alta. [1]

- Ruídos: São inconsistências de dados, que podem fazer parte do conjunto de treinamento .

- *Outlier*: Quando um dado possui dados muito discrepantes dos outros do conjunto de exemplos.

- Matriz de Confusão: É uma medida que serve para mostrar a quantidade de classificações corretas e as classificações preditivas para cada classe. Na diagonal principal ($i = j$) estão o número de dados classificados corretamente e os demais elementos da matriz ($i \neq j$), representam os erros de classificação. Então, um bom classificador possui todos ou maior quantidade de zeros nas diagonais, pois assim, significa que há menos erros. Para entender outros resultados em uma matriz de confusão, faz-se necessário entender os conceitos apresentados a seguir [5] :

- Verdadeiro Positivo: O dado foi corretamente classificado como positivo.

- Verdadeiro Negativo: O dado negativo foi corretamente classificado como negativo.

- Falso Positivo: O dado negativo foi incorretamente classificado como positivo.

- Falso Negativo: O dado positivo foi incorretamente classificado como negativo.

2.3 Classificação de Dados

Uma das maiores aplicações de AM é a classificação de dados, a qual busca encontrar características e padrões em exemplos de uma determinada base e em seguida,

predizer o rótulo de novos exemplos. Nesse projeto, fizemos uso dos seguintes classificadores: *k-Nearest Neighbors* - kNN, Naive Bayes, Árvore de Decisão, *Suport Vector Machine* - SVM, Redes Neurais, que serão detalhados a seguir.

2.3.1 *k-Nearest Neighbors* - kNN

Um dos classificadores mais simples e utilizados em AM, é o kNN [6]. Sendo um método lazy que tem como ideia principal, analisar a vizinhança de um exemplo, e então, tentar predizar a classe desse elemento.

Isto é, dado um elemento x, ainda não classificado e um k inteiro positivo, o algoritmo calcula a distância, geralmente Euclidiana, de seus atributos para todos atributos dos outros elementos da base de treinamento e salva-os numa matriz de similaridade.

Assim, seja a_i os atributos do elemento 1, e b_i os atributos do elemento 2. A Distância Euclidiana dos atributos será dado por:

$$DistEucl = \sqrt{\sum (a_i - b_i)^2} \tag{1}$$

Após isso, procura-se os k-vizinhos mais próximos de x e dentre eles, há uma nova análise: A classe mais frequente dentre os k-vizinhos será a classe de x. Caso não seja possível determinar qual a classe mais frequente, isto é, há um empate, é preciso diminuir o valor de k. Ou seja, calcular os $k-1$ vizinhos mais próximos, até que seja possível definir a classe de x.

Apesar de ser amplamente utilizado, o algoritmo não é um dos melhores em eficiência. Pois, os dados calculados são guardados em uma matriz $M \times N$ tal que M é o número de elementos da base de treinamento e N o número de atributos dos elementos, o que consome tempo $O(M * N)$ ou $O(n^2)$.

Além disso, para cada nova inclusão de um elemento na base, deve-se recalcular todos os dados. E a distância eucliadiana é uma medida sensível a escala de dados, que pode gerar uma inconsistência na análise, se a escala dos dados for muito discrepante. Felizmente, isso pode ser contornado se os dados forem normalizados ou ainda, fazendo o uso de outras Medidas de Similaridade, como a Medida de Mahalanobis, que não é sensível a escala de dados.

2.3.2 Medidas de Similaridade

Mahalanobis
Criada pelo matemático Prasanta Chandra Mahalanobis em 1936. Usa a correlação entre variáveis para a calcular a similaridade entre as variáveis. Uma das principais vantagens dessa medida é não ser sensível a escala dos dados, como a Distância Euclidiana. Para isso, os dois elementos devem possuir o mesmo número de atributos. Dado $\mu = (\mu_1, \mu_2, \mu_3, \ldots, \mu_p)^t$ a média de um grupo de valores e M uma matriz de covariância para um vetor multivalorado $x = (x_1, x_2, x_3, \ldots, x_p)^t$ [7]

A distância de Mahalanobis é calculada como:

$$DistMah(x) = \sqrt{(x-\mu)^t M^-1 (x-\mu)} \tag{2}$$

Temos algumas propriedades para a fórmula: se a matriz de covariância é a matriz identidade, então a Distância Mahalanobis é igual a Distância Euclidiana [8] e se a matriz de covariância é diagonal, então a medida de distância resultante é chamada de Distância Euclidiana Normalizada. [9]

Manhattan
Criada por Hermann Minkowski no século XIX, é uma das medidas mais simples, em que a distância é dada pela soma das diferenças absolutas dos atributos. [10] Ou seja:

$$DistMan(x) = \sum_{k=1}^{n} |x_{ik} - y_{ik}| \tag{3}$$

Similaridade Cosseno
Similaridade do Cosseno [11], muito usada para classificação de textos, a medida usa o ângulo entre dois vetores como medida de similaridade. Quanto mais próximo ao ângulo de 0^o, os vetores apontam na mesma direção, o que significa que os elementos possuem maior similaridade. Quanto mais próximo de 90^o, mais diferentes os elementos são. Seja então v_1 e v_2 vetores de valores, a distância do Cosseno será dado pelo produto escalar entre eles, divido pelo produtos de seus módulos.

$$Cos(v_1, v_2) = \frac{v_1 \bullet v_2}{\|v_1\| \, \|v_2\|} \tag{4}$$

Casamento Simples
A medida de Casamento Simples [11], mais usada para conjuntos de dados com atributos binários, é uma medida que simplesmente conta o número de 0's e 1's , e que deve-se calcular $M00$, $M01$, $M10$ e $M11$.

- $M00$ número de atributos tal que $a_i = 0$ e $b_i = 0$
- $M01$ número de atributos tal que $a_i = 0$ e $b_i = 1$
- $M10$ número de atributos tal que $a_i = 1$ e $b_i = 0$
- $M11$ número de atributos tal que $a_i = 1$ e $b_i = 1$

Após isso, o coeficiente de casamento simples é dado por:

$$CasSimples = \frac{M11 + M00}{M01 + M10 + M11 + M00} \tag{5}$$

2.3.3 Naive Bayes

Da categoria de classificadores probabilísticos, temos o Naive Bayes [12] , que usa o Teorema de Bayes para classificar novos dados. Com o diferencial que, para a classificação, o algoritmo parte do pressuposto que todos atributos são condicionalmente independente. Isto é, a informação de um dado não afeta outra e por isso o "Naive" em seu nome.
O Teorema de Bayes é dado pela fórmula a seguir:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \tag{6}$$

Onde X é um vetor de tamanho n e $X = (x1, x2, ..., xn)$.

Assumindo a condição de independência, o Teorema de Bayes, pode ser reduzido a seguinte fórmula [13]:

$$y = argmaxyP(y)\prod_{i=1}^{n} P(xi|y) \tag{7}$$

Sendo assim, o algoritmo é descrito como: Primeiro, calcula-se a probabilidade de todos eventos acontecerem usando o Teorema de Bayes assumindo a condição de independência para cada categoria, dado uma amostra, e a classe da amostra será dada pela classe do evento de maior probabilidade.

As principais vantagens deste algoritmo são sua simplicidade e eficiência [14]. Principalmente se levarmos em conta que não é necessário calcular as probabilidades de todos os elementos para todos os outros. Para incluir um novo elemento, não é

necessário recalcular todos os dados, como no classificador kNN, o que economiza custos computacionais.

Mas, se no conjunto de treinamento temos algum evento com probabilidade 0 (*Zero Frequency*), ao usar a fórmula do Teorema de Bayes, o resultado será 0 e com isso, não será possível determinar a classe do elemento. Além disso, o algoritmo parte do pressuposto que todos eventos são independentes, o que não acontece com muita frequência na vida real. Felizmente, o *Zero Frequency* pode ser contornado com a estimativa de Laplace, onde é adicionado 1 a todos elementos e ainda que a suposição de independência seja ingênua, o algoritmo ainda funciona e é muito eficiente para muitos casos.

2.3.4 Árvores de Decisão

Árvore de Decisão [1] é um classificador baseado no modelo de estrutura de árvore. Em que os nós internos simbolizam os testes dos atributos, nos ramos estão os resultados do teste anterior, e nos nós folhas estão as classes. Para gerar uma árvore de decisão, existem duas fases: na primeira, a construção da árvore, em que ocorre o particionamento de atributos; e a segunda fase de poda da árvore, em que são verificadas e excluídas as ocorrências de ramos que possuem ruídos ou *outliers*.

A fase da poda é dividida em duas outras fases: pré-poda e pós-poda. A pré-poda é obtida na inicialização do algoritmo em que alguns exemplos do conjunto de treinamento são ignorados. Na pós-poda, de imediato, é gerado um classificador para os exemplos e então alguns ramos são excluídos para generalizar ainda mais a árvore.

Para que o algoritmo funcione da melhor forma possível, é necessário saber qual atributo escolher para o *split*, pois precisamos encontrar o atributo que melhor divida os dados. Para isso, temos os conceitos de entropia e ganho.

A entropia mede quanto um certo atributo interfere no conjunto, o que é equivalente a medir a impureza dos exemplos. Dado S um conjunto de exemplos e C um conjunto de categorias, a entropia de S é dada pela fórmula:

$$Entropia(S) = \sum_{i=0}^{c} -pi \log_2(pi) \tag{8}$$

O melhor atributo a ser usado como *split* é aquele que possuir menor entropia, pois quanto menor a entropia, há uma menor dúvida, isso significa que esse atributo traz a maior informação sobre o conjunto de classes.

Se um atributo x pode assumir n valores, serão formados n subconjuntos, em que cada conjunto terá seu valor de entropia. Nesse caso, o cálculo de entropia envolverá a média ponderada da entropia dos subconjuntos de x, sendo calculada dessa maneira:

$$Entropia(x, S) = \sum_{i=0}^{c} \frac{|Si|}{|S|} \tag{9}$$

Assim, o ganho de informação, ou redução de entropia, em virtude da escolha do atributo x é dado por:

$$GanhoInformacao(x, S) = Entropia(S) - Entropia(x, S) \tag{10}$$

Uma das principais vantagens do algoritmo é sua eficiência computacional. Pois é muito rápido para classificar dados, é fácil de interpretar uma árvore de decisão de tamanho pequeno e a precisão é similar aos outros algoritmos de classificação, para conjuntos pequenos. Mas, tem como desvantagem que se houver um "overfitting" no conjunto de dados, o resultado será uma árvore mais complexa. E é sensível a ruídos, com isso, pode não produzir uma boa estimativa para outras amostras.

2.3.5 *Support Vector Machine* - SVM

SVM é um método de aprendizado proposto por Vladimir Vapnik e usado idealmente para classificação binária $0, 1$. Para a separação dos dados, há infinitas retas, a ideia do algoritmo é encontrar a reta com maior margem de separação. Isto é, encontrar um hiperplano ótimo. Para isso, usa-se os chamados vetores de suporte definidos como os vetores x_i tal que $y_i(w.x_i+b) = 1$ e serão usados para determinar a margem.

Um hiperplano pode ser definido como:

$$w_0.x + b_0 = 0 \tag{11}$$

A equação divide o espaço em duas regiões $w_0.x + b_0 > 0$ e $w_0.x + b_0 < 0$ e a classificação é dada por:

$$y(x) = \begin{cases} +1 & \text{se } w_0 \cdot x + b > 0 \\ -1 & \text{se } w_0 \cdot x + b < 0 \end{cases} \tag{12}$$

E um hiperplano é ótimo se possui grande margem ρ. Essa margem ρ é calculada por $\rho = \frac{2}{||w_0||}$ e sendo assim, teremos uma maior margem, tendo um $\|w_0\|$ menor.

Como esse método é linear, é sensível a problemas como *outliers* e logo, não será possível separar os dados de forma que não haja nenhum erro. Sendo assim, para fazer a separação com o minímo de erros possível, a solução é usar a ideia das variáveis de folga, usando um parâmetro C. Ainda, em muitas situações os dados não são linermente separáveis, portanto há uma outra abordagem chamada SVM Não-linear, em que os dados são mapeados para um dimensão maior que a original. Ou seja, se os dados fazem parte do $\Re^n$, esses dados são mapeados para $\Re^{n+1}$ e assim, os dados serão linearmente separáveis. Esses mapeamentos seguem as funções de Kernel, definidas pelas Tabela 1.

Kernel	Função $\phi(x_i, xj_)$
Polinomial	$(\delta(x_i \bullet x_j) + k)^d$
Gaussiano	$\exp(-\sigma \left\|x_i - x_j\right\|^2)$
Sigmoidal	$\tanh(\delta(x_i \bullet x_j) + k)$

Tabela 1: Tabela de funções de kernel.

Idealmente, o SVM é utilizado para problemas de clasificação binárias. Mas problemas da vida real, exigem trabalhar com múltiplas classes. Dessa forma, há algumas técnicas para contornar a situação como as abordagens *Um contra o resto* e a abordagem *Pairwise*. Na primeira, são construídos um SVM por classe em que são usado as amostras de uma classe comparando com os exemplos de todas as outras classes e para decidir a classe, é utilizado uma estratégia de votos. Na segunda, as amostras são classificadas em pares.

Portanto, para que o classificador funcione corretamente, é necessário analisar o conjunto de dados e pensar em qual função de kernel e qual o valor do parâmetro C de margem de folga utilizar. Dessa forma, usamos as técnicas de *cross-validations*. [15]

Em geral, o SVM é um bom classificador, pois consegue trabalhar com grandes conjunto de dados, com alta dimensão, e a classificação é rápida. Contudo, é necessário fazer um boa escolha de kernel e o tempo de treinamento pode ser demorado de acordo com o número de exemplos e a dimensão dos dados. [15]

2.3.6 Redes Neurais

Redes neurais é uma técnica inspirada na estrutura neural de de organismos inteligentes e que adquirem conhecimento através da experiência. No cérebro, a inte-

ligência é uma propriedade resultante de um grande número de unidades simples, enquanto que o contrário acontece com regras e algoritmos simbólicos. [16]

O sistema nervoso é formado por neurônios, que são células complexas que ligam e desligam em alguns milissegundos, já o hardware que conhecemos atualmente é capaz de fazer a mesma tarefa em nano segundo. No entanto, o cérebro é capaz de realizar outras tarefas cognitivas complexas como a visão, reconhecimento de voz em décimos de segundo. [16]

Essas celulas são compostas por três partes: dentritos, corpo celular ou soma e o axônio. Os neurônios comunicam através de sinapses por seus dendritos. Os sinais de comunicação são acumulados em seu corpo e quando a soma dos sinais passa de um limiar determinado, um sinal é propagado no axônio, que distribui a informação processada para outros neurônios ou partes do corpo. Além disso, as sinapses possuem pesos que podem ser excitatório, ou seja, incrementam a soma dos sinais ou inibidor, que subtrai da soma dos sinais. [16]

O cérebro humano é bom em reconhecer padrões, fazer associações e tolera ruídos. O computador é bom em cálculos, precisão e lógica. Então, pode-se representar uma rede neural com um certa eficiência diante ao cérebro humano. [16]

Assim, esses modelos de redes neurais são sempre acoplados a alguma regra de treinamento para definição dos pesos. Além disso, são organizados em camadas. Sendo elas:

- Camada de entrada: Os padrões de entradas que farão as conexões.
- Camada intermediária ou oculta: Onde é feito grande parte do processamento com as conexões.
- Camada de saída: o resultado.

[17]

Em redes neurais, temos como formas de aprendizado o *Perceptron* e *Backprogation*.

Perceptron

É um método ideal para redes neurais simples onde há apenas uma camada. O método recebe um conjunto de exemplos de treinamento que dão a saída desejada para uma unidade, dado um conjunto de entradas. A ideia é aprender pesos sinápticos, analisando os exemplos, até modelar uma função que possa classificar a saída corretamente. [16]

Na unidade de *Threshold Linear*, os pesos são somados, se essa soma é maior que 0, recebe 1. Senão, recebe 0. [16]

Para que um perceptron possa modelar a função corretamente, é necessário que se faça mudanças no valor dos pesos ajustáveis por uma quantidade proporcional a diferença entre a saída desejada e a atual saída do sistema. Os pesos são dados por:

$$W_i = W_i + \Delta W_i \Delta W_i = \eta(t - o)X_i \tag{13}$$

Em que t é a saída desejada, o a saída atual e Δ a taxa de aprendizado. [16]

Assim, se a saída do perceptron não estiver correta, os pesos devem ser alterados, até que se obtenha novos pesos que produzam a saída mais próximo da saída desejada. Para que o algoritmo convirja para a classificação correta, o conjunto de treinamento deve ser linearmente separável e a taxa de aprendizado, deve ser suficientemente pequena. [16]

Para trabalhar com superfícies de decisão não linear, é necessário fazer uma combinação de perceptrons lineares para gerar superfícies de decisão mais complexas. Para as redes multicamadas são adicionadas as camadas ocultas. Em que o uso de uma permitem que a rede possa gerar uma função de *convex hull* e usando várias, é possível gerar diferentes *convex hull* que separam os dados. [16]

Quando se há funções que não podem ser modeladas por funções lineares, é necessário usarmos funções de ativação, como a sigmoidal e radial (gaussiana). [16]

Backpropagation

É usado para redes multicamadas que tenha um número fixo de unidades e interconexões. Funciona usando a descida do gradiente para minimizar o erro quadrático entre a saída e os alvos de saídas da rede. [16]

A ideia do método do gradiente é encontrar um vetor de pesos que minimiza o erro, assim, começa com um vetor inicial arbitrário e vai sendo alterado na direção que produz a maior queda ao longo da superfície de erro. [16]

As redes neurais podem gerar bons classificadores, pois possuem boa capacidade de generalização, certa imunidade a ruídos, auto-aprendizado e geralmente, possui um grau de eficiência maior que as técnicas estatísticas. Como desvantagem, deve-se trabalhar com um grande volume de dados e estes, devem ser normalizados, além de muitas vezes o treinamento ser demorado e a possibilidade dos resultados contrariarem regras e teorias estabelecidas. [18]

3 *Data Augmentation*

Em AM, o desempenho dos algoritmos no aprendizado supervisionado pode ser estendido de acordo com o aumento no número de instâncias numa base de dados. No entanto, em situações da vida real, nem sempre temos muitos exemplos disponíveis ou muitas vezes, a maioria destes são não rotulados.

Dessa forma, criou-se o conceito de *Data Augmentation* (DA), que em tradução direta, significa aumento de dados. O DA tem como objetivo utilizar técnicas computacionais para aumentar a base de dados original e como consequência, obter uma melhor acurácia na classificação.

Uma abordagem mais simples e comum quando se está trabalhando com aprendizado semissupervisionado e há uma base com poucos elementos rotulados e bastante elementos não-rotulados, é classificar esses elementos não-rotulados e gerar uma nova base de dados aumentada de exemplos classificados.

3.1 Transformações de Imagens

Quando se trabalha com imagens e deseja-se aumentar essa base, uma abordagem simples e muito utilizada é a de transformação de imagens. Nessa técnica, a partir de um conjunto de imagens determinadas são aplicadas edições de imagem como corte, rotação, blur, etc na imagem original, gerando uma nova imagem. Assim, é gerado uma base aumentada das imagens originais mais as imagens editadas.

3.2 Algoritmo *Global and Local Consistency* GLC

Um dos principais conceitos do aprendizado semissupervisionado é a assunção de consistência local e global [19], isto é, pontos próximos (local) e de mesma estrutura (global) são prováveis de possuírem o mesmo rótulo.

A maioria dos algoritmos de aprendizado supervisionado assumem apenas uma das duas opções. Por exemplo, o tradicional algoritmo kNN assume apenas o conceito de consistência local. No entanto, o algoritmo semissupervisionado aqui usado propõe um novo método que utiliza as duas opções, e por isso, seu nome.

Primeiro, dado um grafo ponderado $G(V, E, W)$, onde V representa um conjunto de vértices, E um conjunto de arestas e $W = \{w(e_1), \ldots, w(e_k), \ldots, w(e_M)\}$ um conjunto de pesos. Sendo w uma função $w : E \to \Re$ que atribui para cada aresta $e_{ij} \in E$ um peso $w(e_{ij})$, ou simplesmente w_{ij}.

O algoritmo GLC [19] é descrito como a sequência de passos a seguir:

1º - Calcula-se a matriz de afinidade W, utilizando a seguinte fórmula $W_{ij} = exp(\| x_i - x_j \|^2 / 2\sigma^2)$, se $i \neq j$ e 0, se $i = j$.

2º - Calcula-se a matriz diagonal D, em que a diagonal principal é igual a soma da i-ésima linha de W, como $D_{ii} \leftarrow \sum_0^j W_{ij}$.

3º - Calcula-se a matriz laplaciana normalizada, dada pela fórmula $S = D^{1/2} W D^{1/2}$.

4º - Escolhe-se um parâmetro α no intervalo [0, 1) e realiza-se iterações até atingir a convergência, utilizando a função $F(t+1) = \alpha SF(t) + (1-\alpha)Y$, em que Y é o conjunto de dados rotulados. Ao final, obtém-se F^*.

5º - Por último, rotula-se as amostra xi, com $yi = arg\ max\ _{j \leq c} F^*_{ij}$.

4 Materiais e Métodos

Nessa seção é descrito a metodologia abordada para realização dos testes e as ferramentas utilizadas.

4.1 Base de dados

A base de dados utilizada (Brazilian Coffee Scenes Dataset [20]) é composta por imagens de satélite de plantações de café, em cidades de Minas Gerais pelo sensor SPOT em 2005. Cada imagem possui 64x64 pixels e foram classificadas manualmente por pesquisadores agrícolas como "coffee"se possuem pelo menos 85% dos pixels de café e "non coffee"se possuem menos que 10% dos pixels de café.

Assim, o cojunto de dados é composto por 2876 exemplos, sendo 50% exemplos da classe "coffee"e 50% da classe "non coffee". E a seguir nas figuras 1 e 2, alguns exemplos das imagens capturadas pelo sensor.

Figura 1: Exemplo de imagens rotuladas como "coffee"

Figura 2: Exemplo de imagens rotuladas como "non-coffee"

4.2 Expansão das bases - *Data Augmentation*

Para a expansão dos conjuntos de dados rotulados utilizamos duas abordagens:

1) A primeira delas faz uso de transformações de imagens. Considerando uma porcentagem inicial (ex: 10% de dados com rótulos) aplica-se diversas transformações como rotação, blur, etc nesse subconjunto até expandi-lo para um tamanho N. Na figura 3 temos alguns exemplos das transformações aplicadas nas imagens da base original.

Figura 3: Exemplo de aplicações do efeito das transformações aplicadas nas imagens da base de dados. Na figura a)uma foto de satélite contendo café, b)a figura a com efeito girar, c)a figura a com efeito blur

2) A segunda abordagem considera um conjunto de dados de tamanho N, seleciona uma porcentagem dos dados com rótulo (ex:10%), constrói um grafo usando o algoritmo kNN e aplica um algoritmo semissupervisionado baseado em grafos (consideramos aqui *Global and Local Consistency*) para predizer os rótulos dos outros 90% dos dados.

Em seguida, ambos os conjuntos de dados gerados (compostos de 10% de dados com rótulos original e 90% de dados com rótulos estimados) serão utilizados para treinar um classificador. A função de classificação gerada é testada no conjunto de dados original.

4.3 Classificação dos dados

Para fazer a classificação de dados, fizemos uso do software WEKA - *Waikato Environment for Knowledge Analysis* [21], o qual é um software *open source*, baseado em Java e criado na Universidade de Waikato na Nova Zelândia.

O software conta com uma grande quantidade de algoritmos úteis para AM. Assim, é possível usá-lo para pré-processamento de dados, classificação, regressão, agrupamento, regras de associação e visualização. Além disso, estes podem ser aplicados diretamente a um ou vários conjunto de dados ou utilizando o código Java.

A tela inicial do programa pode ser vista na Figura 4. Nela podemos escolher entre os diferentes modo de utilização. Como o *Explorer*, onde os experimentos são realizados de forma individual e o *Experimenter*, onde é possível trabalhar com várias base de dados e vários algoritmos ao mesmo tempo.

Figura 4: Tela inicial do software WEKA.

Na tela inicial do explorer (Figura 5) é possível carregar uma base, checar resumos estatísticos dessa base e também adicionar filtros, selecionar os melhores atributos e removê-los. Para carregar uma base, é necessário que ela esteja no padrão *.arff* do WEKA, mas também é possível a fazer a conversão do formato da base diretamente pelo software.

No menu de classificação (Figura 6) é possível fazer testes usando 4 técnicas, nesse trabalho usou-se o Supplied test set:

- Training set: Separa algumas instâncias para teste.

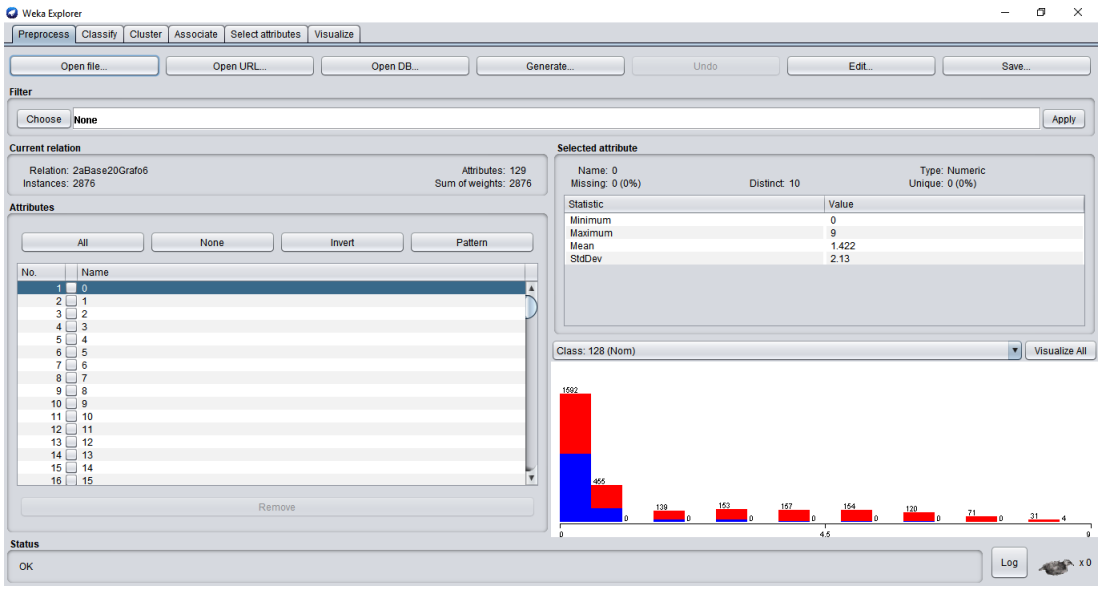


Figura 5: Tela de Pré-processamento dos dados do WEKA.

- Supplied test set: Pode-se carregar um arquivo .arff para os casos de testes.
- Cross-validation: Utiliza o método k-fold para fazer uma validação cruzada.
- Percentage Split: Utiliza uma porcentagem dos dados, definida pelo usuário, para os casos de testes.

No menu superior choose, é possível escolher o classificador a ser usado e clicando na barra ao lado, pode-se alterar algumas propriedades do classificador. Mais baixo, no grande retângulo, temos o resultado do teste. Nele há informações como a porcentagem de acertos, erros, desvio padrão, tempo gastos, matriz de confusão e etc.

Para os experimentos realizados nesse trabalho usamos 5 classificadores disponíveis na ferramenta, sendo eles:

- Naive Bayes: Um classificador probabilístico bayesiano, usando o Teorema de Bayes e assumindo que os eventos são independentes.
- IBK: Um classificador baseado em *Lazy Learning*, simula o algoritmo de classificação kNN.

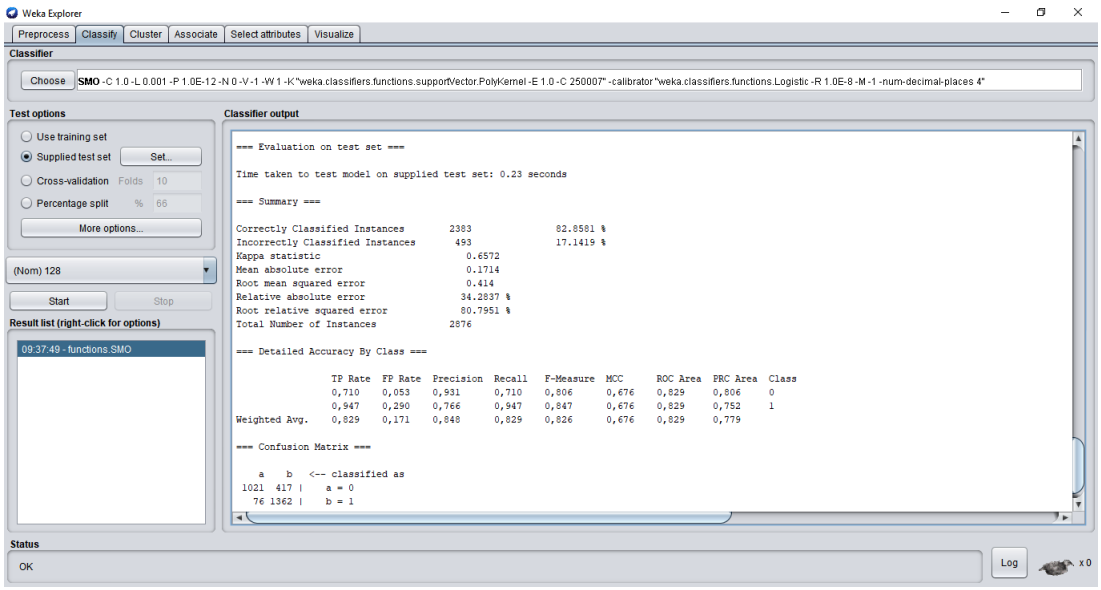


Figura 6: Tela de classificação dos dados do WEKA.

- J48: Um classificador baseado em uma árvore de decisão C4.5, que pode ter poda ou não.
- SMO: Simula o algoritmo *Support Vector Machine* - SVM usando a técnica de otimização mínima de John Platt.
- MLP: Simula uma rede neural, usando *backpropagation* para classificar as instâncias.

Além disso, nas outras abas, temos ferramentas como:

- Cluster: Ideal para agrupamento de dados.
- Associate: Ideal para associação de dados.
- Select Atributes: Disponibiliza várias formas de selecionar os atributos da base, como verificar quais atributos possuem maior informação de ganho, quais tem maior capacidade de predição, grau de redundância e etc.
- Visualize: Há diversas ferramentas de visualização dos dados.

5 Resultados

5.1 Base original

Para os experimentos iniciais foram geradas 5 bases de dados com transformações de imagens e 10 bases de dados com aprendizado semissupervisionado considerando 10% dos dados rotulados. Assim como a base original, cada base gerada para os experimentos contém 2876 instâncias e cada instância possui 128 atributos mais sua classe. Os experimentos foram realizados no software livre WEKA, usando 5 classificadores: Naive Bayes, IBK (com $k = 1$, $k = 3$ e $k = 5$.), MLP, SMO e J48. Mas, primeiramente, foram realizados experimentos de cross-validation com a base original (livre de aumento de dados como a transforamação de imagens e propagação de rótulos que será usados nos próximos experimentos) usando $k = 10$, usando os mesmos classificadores e os resultados obtidos são mostrados na tabela 2 a seguir.

Tabela 2: Tabela de resultados obtidos com o cross-validation da base original

Classificador	Acertos (%)	Erros (%)
Naive Bayes	79.7288	20.2712
kNN, k = 1	83.0320	16.9680
kNN, k = 3	85.7441	15.2559
kNN, k = 5	86.7177	13.2823
SMO	87.4131	12.5869
MLP	84.9444	15.0556
J48	83.0320	16.9680

5.2 Classificação usando bases geradas pelo aprendizado semissupervisionado baseado em grafos

A Tabela 3 mostra a média dos resultados obtidos na classificação a partir dos dados gerados pelo aprendizado semissupervisionado baseado em grafos. O melhor resultado foi obtido usando o classificador SMO (Tabela 4 mostra a acurácia e Tabela 5 a matriz de confusão). E o pior resultado foi obtido usando o classificador Naive Bayes (Tabela 6 mostra a acurácia e Tabela 7 a matriz de confusão).

A partir desses resultados, notamos uma certa discrepância entre os erros na matriz de confusão. A Tabela 7 apresentou um número muito alto de falsos positivos e verdadeiros negativos.

Tabela 3: Média do resultado entre todos os classificadores para dados gerados via aprendizado semissupervisionado baseado em grafos.

	Média	Porcentagem
Acertos	2302	80,04
Erros	574	19,96

Tabela 4: Melhor resultado obtido com o classificador SMO para dados gerados via aprendizado semissupervisionado baseado em grafos.

	Média	Porcentagem
Acertos	2444	85
Erros	432	15

Tabela 5: Matriz de confusão para o classificador SMO.

	a	b
a = 0	1144	294
b = 1	138	1073

Tabela 6: Pior resultado obtido com o classificador Naive Bayes para dados gerados via aprendizado semissupervisionado baseado em grafos.

	Média	Porcentagem
Acertos	2117	73,6
Erros	759	26,4

Tabela 7: Matriz de Confusão para o classificador Naive Bayes.

	a	b
a = 0	1181	257
b = 1	502	936

Porém, houve casos em que o número de falsos positivos foi menor mas os verdadeiros negativos foram bem maiores, como mostra a Tabela 8 usando o classificador MLP. Essa discrepância nos itens classificados erroneamente mostra que o aprendizado semissupervisionado baseado em grafos influencia a base gerada e consequentemente o algoritmo de classificação. Posteriormente, estudamos os grafos gerados a fim de identificar algum padrão entre grafo $\times$ acurácia do classificador.

Tabela 8: Matriz de Confusão para o classificador MLP.

	a	b
a = 0	917	521
b = 1	56	1382

5.3 Classificação usando bases geradas por transformações em imagens

A Tabela 9 mostra a média dos resultados obtidos na classificação a partir dos dados gerados por transformações em imagens. O melhor resultado foi obtido usando o classificador SMO (Tabela 10 mostra a acurácia e Tabela 11 a matriz de confusão). E o pior resultado foi obtido usando o classificador J48 (Tabela 12 mostra a acurácia e Tabela 13 a matriz de confusão).

Tabela 9: Média do resultado entre todos os classificadores para dados gerados via transformações em imagens.

	Média	Porcentagem
Acertos	2262	78,66
Erros	614	21,34

Tabela 10: Melhor resultado obtido com o classificador SMO para dados gerados via transformações em imagens.

	Média	Porcentagem
Acertos	2378	82,7
Erros	498	17,3

Tabela 11: Matriz de confusão para o classificador SMO.

	a	b
a = 0	1147	291
b = 1	207	1073

Tabela 12: Pior resultado obtido com o classificador J48 para dados gerados via transformações em imagens.

	Média	Porcentagem
Acertos	2041	71,0
Erros	835	29,0

Tabela 13: Matriz de confusão para o classificador J48.

	a	b
a = 0	1181	257
b = 1	502	936

5.4 Discussão

Como fora observado, a classificação usando aprendizado semissupervisionado baseado em grafos teve em média 80,04% de acertos, enquanto que usando transformação de imagens obteve 78,66% de acertos. Levando em consideração a média e os resultados em geral, foi possível concluir que o desempenho do DA baseado em transformações de imagens é levemente inferior ao mesmo processo, usando aprendizado semissupervisionado baseado em grafos.

Além disso, usando aprendizado semissupervisionado baseado em grafos foi possível observar uma particularidade, algumas das bases que obtiveram menor desempenho (78,94% e 79,40%) o número de falsos positivo ou verdadeiros negativo teve muita discrepância. Por isso, realizamos experimentos focando em bases geradas pelo aprendizado semissupervisionado baseado em grafos para identificar se existem diferenças entre os grafos gerados e como estes podem estar influenciando na classificação.

5.5 Análise do método semissupervisionado baseado em grafos

Para os novos testes focando no aprendizado semissupervisionado baseado em grafos foram geradas novas bases de treino, com diferentes porcentagens de exemplos rotulados: 5%, 10%, 15% e 20%. Para cada porcentagem gerou-se 10 grafos variando-se o parâmetro k, correspondente ao número de vizinhos, de 1 a 10 (aqui referidos como Grafo1, Grafo2, etc), totalizando 40 grafos. Como mostra a Figura 7, notou-

se um aumento relativo da média da acurácia na classificação, conforme aumenta-se a porcentagem de exemplos rotulados.

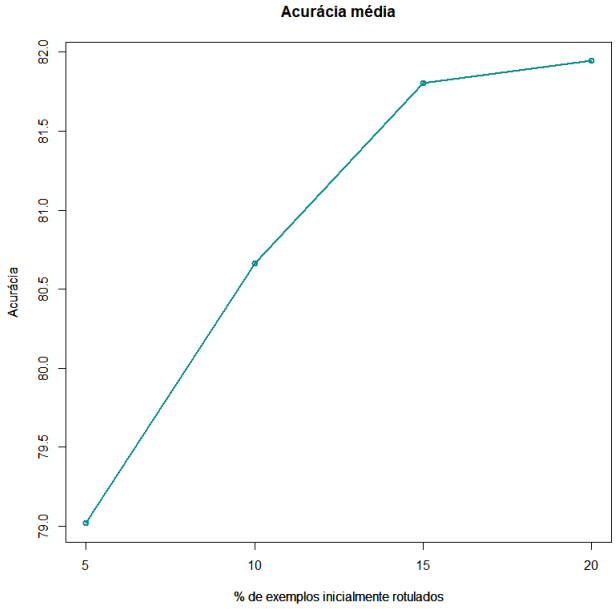


Figura 7: Relação entre porcentagem de exemplos rotulados × acurácia média na classificação.

Também foi possível notar que com o aumento de exemplos rotulados houve uma menor variação entre os maiores e menores resultados obtidos nos experimentos, conforme mostra a Tabela 14.

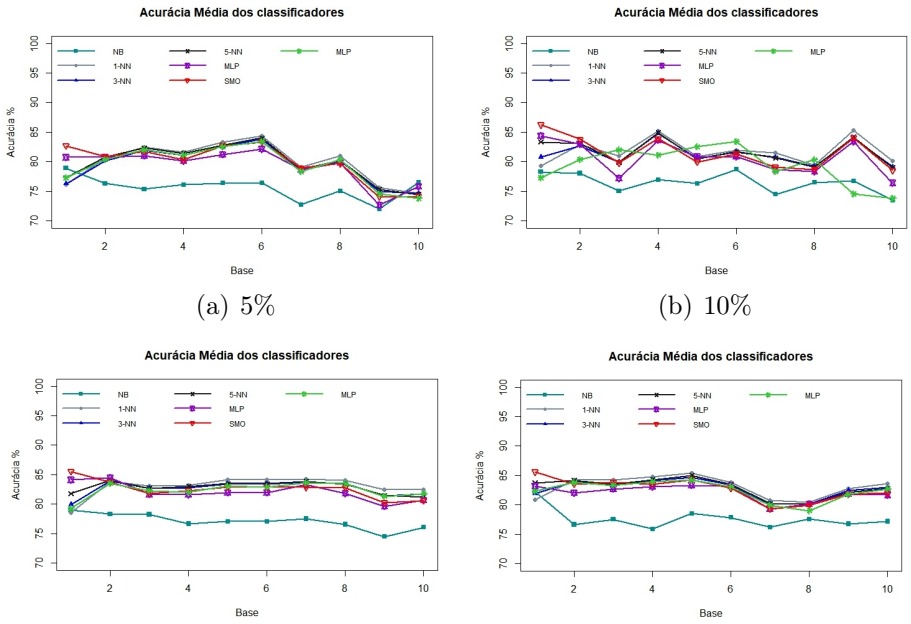
Tabela 14: Máximos e mínimos de acerto na classificação para dados gerados com diferentes porcentagens de exemplos rotulados no aprendizado semissupervisionado baseado em grafos.

	Máximo	Mínimo
5%	82,495	74,170
10%	83,409	77,935
15%	83,126	80,136
20%	83,633	79,420

A acurácia para todos os classificadores testados em todos os grafos é mostrado na Figura 8. O classificador que obteve pior desempenho geral foi Naive Bayes, o

que obteve melhor resultado geral foi kNN, que é um método lazy, seguido de SMO e J48.

Observa-se que os grafos 5, 6, 7, 8, 9, 10 apresentaram uma notável discrepância em sua matriz de confusão. Mas, analisando a média de acertos, não há relação direta entre acurácia $\times$ discrepância na matriz de confusão, pois os grafos 7, 9, 10 possuem menor média de acurácia, enquanto que o grafo 6 possui a maior média.



(a) 5% (b) 10%

(c) 15% (d) 20%

Figura 8: Número de acertos para todos os classificadores considerando diferentes porcentagens de rótulos: a) 5%, b) 10%, c) 15% e d) 20%.

5.6 Análise de Centralidade

Afim de entender os resultados obtidos também analisamos os grafos gerados por meio de medidas de centralidade [22], entre elas:

- Grau: Mede a quantidade de conexões que um vértice possui.

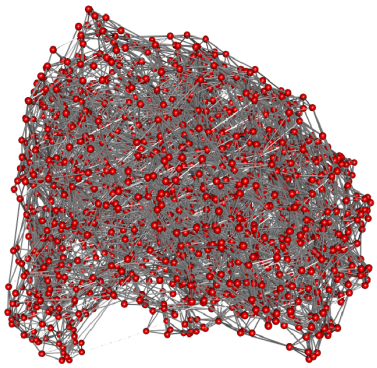
- Closeness: Calcula quanto um vértice está próximo dos outros, em média, usando a menor distância do vértice até os demais.
- Betweenness do vértice: Calcula quantas vezes um vértice age como ponte, isto é, quantas vezes ele faz parte do menor caminho entre os outros vértices dos grafo.
- Betweenness da aresta: Análogo ao betweenness do vértice, mas agora, quantifica quantas vezes uma aresta age como ponte na rede.
- Coeficiente de clustering [23]: Mede o agrupamento do grafo, geralmente contabiliza o número de triângulos para cada vértice.
- PageRank: Quantifica um score baseado na influência do vértice na rede, checando suas conexões e atribuindo maiores valores para conexões com outros vértices que também possuam maior valor (maior grau).

Os experimentos foram realizados pelo software livre R, com a instalação da biblioteca igraph, que permite gerar a visualização dos grafos e cálculo das medidas de centralidades. Os resultados obtidos são mostrados na Tabela 15. A Tabela 15 foi rearanjada da seguinte maneira: Grafo10 obteve o pior resultado na classificação e Grafo6 o melhor, seguido do Grafo9 com segundo pior resultado na classificação e Grafo4 com segundo melhor resultado e por último Grafo7 com terceiro pior resultado na classificação e Grafo2 com terceiro melhor resultado.

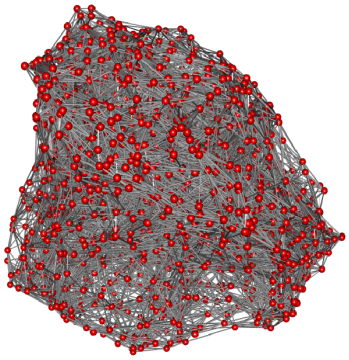
Tabela 15: Medidas de centralidade calculados para diferentes grafos.

Medida	Grafo10	Grafo6	Grafo9	Grafo4	Grafo7	Grafo2
Grau	20	12	18	8	14	4
Closeness	7.284455e-05	6.006604e-05	7.024474e-05	4.938123e-05	6.402957e-05	2.163702e-05
Bet. Vértice	5464.415	6941.54	5721.429	8764.107	6421.338	16018.76
Bet. Aresta	14380.5	8628.5	12942.5	5752.5	10066.5	2876.5
Clustering	0.2870908	0.2542872	0.2801525	0.2205095	0.2647996	0.1505595
PageRank	0.0003477051	0.0003477051	0.0003477051	0.0003477051	0.0003477051	0.0003477051

Fazendo uma comparação direta dos resultados obtidos do k-ésimo grafo com menores resultados com o k-ésimo grafo com maiores resultados, foi possível concluir que os grafos com resultados inferiores, possuem sempre maior grau, maior closeness, maior betweenness da aresta, maior clustering e menor betweenness do vértice em relação as redes com resultados superiores. Enquanto, que o PageRank se manteve praticamente igual para todas os grafos gerados. Conclui-se então que um parâmetro k menor para gerar os grafos leva a resultados melhores de classificação.



(a) Rede com maior média de resultados



(b) Rede com menor média de resultados

Figura 9: Plot das redes geradas após a aplicação do algoritmo kNN

5.7 Redes Neurais Profundas

Uma área crescente em aprendizado de máquina é o Deep Learning, também conhecido como aprendizado profundo. Este método de aprendizado .....

No WEKA, o número de camadas ocultas é limitado a 3 unidades. Por padrão, o MLP é iniciado com uma única camada oculta e com um valor determinado pelo conjunto de dados utilizado. Nas configurações do classificador, além da função de ativação e visualização, podemos definir a quantidade de camadas ocultas e seus respectivos valores. Assim, variando alguns parametros foram feitos novos experimentos utilizando da base 6 aumentada de um conjunto de exemplos com 20% dos exemplos rotulados inicialmente, com as novas alterações. Considere também que

a = número de atributos/2, b = números de atributos e 0 = sem camada oculta. Temos os seguintes resultados conforme a tabela 16.

Tabela 16: Teste com variações de camadas com redes neurais no WEKA.

Camadas	Acertos (%)	Erros (%)
a, 0, 0	83,1711%	16,8289%
a, a, 0	82,8929%	17,1071%
a, a, a	82,3713%	17,6287%
t, 0, 0	79,9722%	20,0278%
t, t, 0	83,2058%	16,7942%
t, t, t	83,0668%	16,9332%
200, 200, 0	83,5883%	16,4117%
248, 248, 248	83,1015%	16,8985%
250, 250, 0	80,9805%	19,0195%
300, 300, 0	83,1363%	16,8637%

Nota-se que utilizando de duas camadas ocultas de valor 200, obtemos %83,5883 de acertos. Maior que o resultado obtido, utilizando das configurações padrões do classificador.

No entanto, o software é limitado quanto a essa técnica. Pois permite o uso de no máximo 3 camadas ocultas.

Referências

[1] S. R. M. Oliveira, "Introdução à aprendizagem de máquina," 2012.

[2] SAS, "Machine learning what it is and why it matters.," 2000.

[3] M. C. Monard and J. A. Baranauskas, "Conceitos sobre aprendizado de máquina," *Sistemas inteligentes-Fundamentos e aplicações*, vol. 1, no. 1, p. 32, 2003.

[4] J. Milgram, M. Cheriet, and R. Sabourin, ""one against one" or "one against all": Which one is better for handwriting recognition with svms?," in *tenth international workshop on Frontiers in handwriting recognition*, SuviSoft, 2006.

[5] E. J. da Silva, "Modelagem e aplicação de técnicas de aprendizado de máquina para negociação em alta frequência em bolsa de valores," 2015.

[6] T. d. C. de Curso, "MÉtodos baseados na regra do vizinho mais prÓximo para reconhecimento de imagens,"

[7] Stephanie, "Mahalanobis distance: Simple definition, examples," 2017.

[8] C. McCormick, "Mahalanobis distance," 2014.

[9] R. Wicklin, "What is mahalanobis distance?," 2012.

[10] P. E. Black, "Manhattan distance.," 2006.

[11] J. Jason, "Medidas de similaridade," 2012.

[12] B. Stecanella, "A practical explanation of a naive bayes classifier," 2017.

[13] N. Kumar, "Naive bayes classifiers," 2000.

[14] S. RAY, "6 passos fáceis para aprender o algoritmo naive bayes (com o código em python)," 2016.

[15] A. A. P. Biscaro, "Redes neurais artificiais mÁquina de vetor de suporte (support vector machines)," 2017.

[16] E. S. de Lima, "Redes neurais.," 2014.

[17] A. P. de Leon F. de Carvalho, "Redes neurais artificias," 2000.

[18] A. C. M. Mattos, "Vantagens e desvantagens das redes neurais," 2000.

[19] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Scholkopf, "Learning with local and global consistency," in *Advances in neural information processing systems*, pp. 321–328, 2004.

[20] J. A. d. S. O. A. B. Penatti, K. Nogueira, "Do deep features generalize from everyday objects to remote sensing and aerial scenes domains?," 2015.

[21] M. L. G. at the University of Waikato, "Weka 3: Data mining software in java)," 2017.

[22] A. Disney, "Keylines faqs: Social network analysis," 2014.

[23] J. Kunegis, "Defining the clustering coefficient," 2014.

6 Apoio

O presente projeto foi apoiado e financiado pela UNIFESP - Universidade Federal de São Paulo.

7 Agradecimentos

Agradeço primeiramente aos meus pais e irmã por todo o apoio até aqui. Aos amigos. A profª Lilian Berton pela oportunidade e todas as orientações e a UNIFESP pelo apoio financeiro.