
Universidade Federal de Pernambuco - CIN

Aprendizagem de Máquina

# Relatório da Lista de Exercícios 3

*Feito por :*
Arnaldo Rafael Morais Andrade
arma

# Conteúdo

# 1 Objetivos

Propor e implementar um sistema que detecta "bugs"em software, para isso, deve-se supor que há apenas instâncias da classe "sem defeito"para o treino do classificador. A proposta deve ser explicada através de pseudo-código e/ou um fluxograma. Avaliar a máquina usando padrões das classes com e sem efeito, reportando: True Positive, False Positive e F1-measure. Os experimentos devem ser realizados em dois bancos de dados distintos do repositório Promise.

# 2 Proposta

O sistema de detecção de falhas baseia-se em dois métodos. O primeiro se trata de um classificador bayesiano para uma única classe [1] (*NBPC* - Naive Bayes Positive Class) supondo uma distribuição gaussiana. Já o segundo, é um classificador baseado na técnica de clusterização k-Means [2]. De uma maneira geral, uma amostra é classificada como anomalia se uma das máquinas a classificar como tal.

Para a fase de treinamento deste sistema, além do conjunto de treino, também é necessário a escolha de dois parâmetros: o número de clusters ($k$) e um valor limiar ($t \geq 1$) que será usado para decidir a classe de uma instância. Ambos parâmetros são referentes ao método de agrupamento.

Ao final desta fase, são computados: os $k$ centroides, bem como os integrantes dos grupos e o valor da probabilidade da instância menos frequente, que é dado por:

$$bayes_{threshold} = min\left[\forall x \prod_{j}^{attributes} P(A_j = v_i)\right] \tag{1}$$

Em que $x$ é uma instância de treino, $A_j = v_i$ é o valor do atributo para o exemplo $x$ e $p(A_j = v_i)$ é a probabilidade do atributo $j$ sendo $v_i$, que é o seu valor atual. Para dados contínuos, esta probabilidade foi obtida de acordo com a distribuição gaussiana, como falado anteriormente.

Um dos pontos chaves para as etapas do sistema é de que os dados de treinamento são normalizados para o k-Means e não normalizados para o classificador de Bayes por conta do cálculo de probabilidade.

No primeiro momento da fase de predição é necessário calcular a seguinte relação:

$$ratio = \frac{d(x, C_i)}{d(m_i, C_i)} \tag{2}$$

Agora, $x$ é uma instância de teste, $C_i$ é o centroide mais próximo dela que contém ao menos um integrante e $m_i$ é o membro do grupo mais distante do centro $C_i$. A distância Euclideana foi utilizada para o cálculo.

Logo em seguida, é preciso estimar a probabilidade do mesmo exemplo de teste de acordo com:

$$P(x) = \prod_{j}^{attributes} P(A_j = v_i) \tag{3}$$

Com os valores fornecidos e os calculados é possível definir a classe da instância de teste:

$$class(x) = \begin{cases} \text{neg}, & \textbf{if } ratio > t \textbf{ or } P(x) < bayes_{threshold} \\ \text{pos} & otherwise \end{cases} \tag{4}$$

Portanto, dentro do espaço de atributos, uma instância deverá estar contida em uma hiperesfera para ser rotulada como classe positiva. Esta hiperesfera é determinada pelo parâmetro $t$, que é interpretado como o quão distante o ponto poderá ser do seu centroide mais próximo.

Entretanto, há casos em que o ponto está localizado dentro da hiperesfera e se trata de uma anomalia. Para corrigir isso, entra-se a parcela do classificador bayesiano, ou seja, caso o exemplo de teste tenha uma probabilidade inferior à um valor limite, $bayes_{threshold}$, ele será considerado como classe negativa.

O pseudocódigo 1 sumariza a proposta estabelecida.

**Algorithm 1** K-means Bayes

**Input:** $D$, the test set, $D_n$, the normalized test set, parameter $t$, parameter $bayes_{threshold}$, set of cluster representatives $C$ and cluster membership vector $m$.

Assuming that the means ($\mu$) and variances ($\sigma^2$) for each attribute are known.

**Output:** $y$, predicted classes vector

$y \leftarrow$ empty_vector()
**for** $i = 1, 2, \ldots,$ len($D$) **do**
  $x \leftarrow D[i],\ x_n \leftarrow D_n[i]$
  $C_i \leftarrow$ closest_non_empty_centroid($C, x$)
  $m_i \leftarrow$ furthest_member($C_i, m$)
  $ratio \leftarrow \frac{d(x,C_i)}{d(m_i,C_i)}$ ▷ Euclidean distance
  $prob \leftarrow 1$
  **for** $j = 1, 2, \ldots,$ len($x_n$) **do** ▷ Computing probability of $x_n$
    $att \leftarrow x_n[j]$
    $probs \leftarrow probs *$ probability($att, \mu_j, \sigma_j^2$) ▷ Gaussian distribution
  **end for**
  **if** $ratio > t$ **or** $probs < bayes_{threshold}$ **then** $y_i =$ neg ▷ Computing class of $x$
  **else** $y_i =$ pos
  **end if**
  $y$.append($y_i$)
**end for**
**return** y

# 3 Metodologia dos Experimentos

As implementações foram feitas utilizando a linguagem Python.

## 3.1 Bases de Dados

As duas bases retiradas do repositório Promise são relacionadas à detecção de erros em software. São elas: datatrieve.arff e cm1.arff. A base DATATRIEVE possui 9 atributos e 130 instâncias que podem ser classificadas como "0"(91.54% - classe "sem defeito") ou "1"(8.46%). Enquanto que a outra, CM1, dispõe de 22 *features* e 498 registros, dos quais podem ter os valores "false"(90.16% - classe "sem defeito") ou "true"(9.83%).

Utilizou-se uma versão não normalizada e outra normalizada do conjunto de dados

como entradas para o algoritmo proposto. O método de normalização pode ser visto na equação abaixo:

$$x_{norm} = a * \frac{x - x_{min}}{x_{max} - x_{min}} + b \tag{5}$$

### 3.2 Treino

Na fase de treino, foram utilizados 30%, 40% e 50% de dados da classe "sem defeito", escolhidos aleatoriamente.

Para o método k-Means, a seleção inicial dos centros dos clusters foi feita de uma maneira inteligente a fim de acelerar o processo de convergência, de acordo com a biblioteca *scikit-learn*. Além disso, as sementes de centroides foram definidas após 10 execuções do método, escolhendo o conjunto que obteve a menor soma do erro quadrático (SSE). Com a finalidade de agilizar a etapa de predição, as distâncias dos membros mais distantes em relação aos respectivos centros foram pré-computadas em formato de um dicionário.

Já para o outro método, bayesiano, com o intuito de encontrar rapidamente a menor probabilidade global, $bayes_{threshold}$, foram calculadas as médias e variâncias de cada atributo não normalizado do conjunto de treino. Dessa forma, foi possível reduzir o tempo de processamento tanto na etapa de treino quanto na de testes.

### 3.3 Experimentos

Os experimentos foram realizados em uma máquina virtual Ubuntu 18.04.5 LTS, com 7.8 GB de memória e 3 núcleos de CPU virtuais.

## 4 Resultados e Discussões

Nesta seção, há 2 figuras para cada base de dados. As figuras 1 e 2 pertencem ao conjunto de dados DATATRIEVE e as outras referem-se ao CM1.

Como o algoritmo proposto possui dois principais argumentos de entrada, uma busca exaustiva foi realizada com o objetivo de encontrar os melhores parâmetros dado o conjunto de treino. Os melhores foram aqueles que maximizaram a área sob a curva ROC (AUC). Portanto, no gráfico do espaço ROC, um ponto representa a configuração mais razoável para aquele conjunto de treinamento.

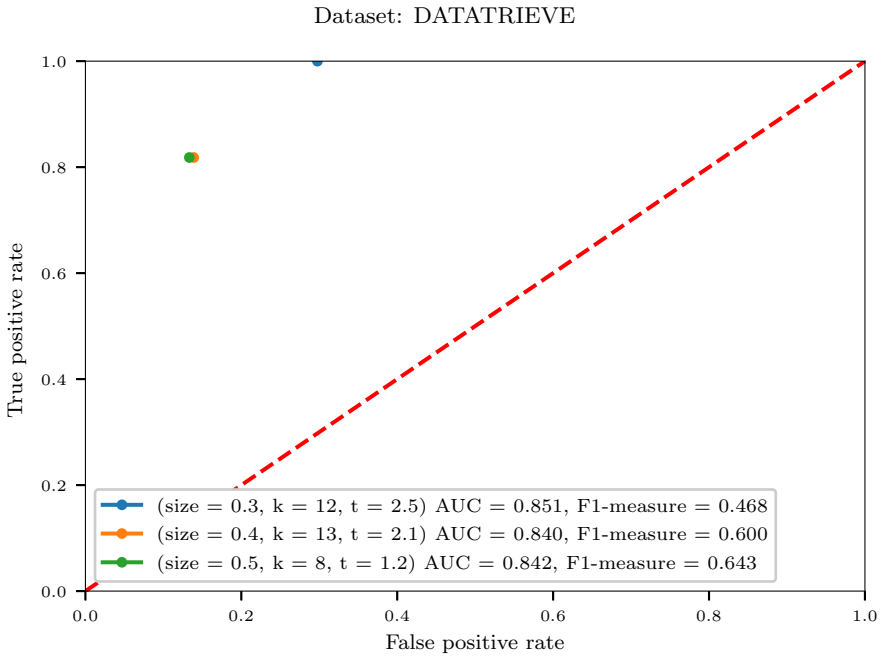


Figura 1: Espaço ROC - Melhores máquinas

A figura acima tem um caráter mais expositivo a comparativo, visto que as máquinas dependem inteiramente do conjunto de ensaio. Como na legenda, *size* representa a taxa referente ao tamanho da coleção de treino, $k$ é o número de clusters e $t$ é o valor limiar usado para definir a classe da instância, conforme mostrado anteriormente.

Embora as máquinas sejam representadas como pontos, o valor AUC pode ser calculado a partir da área do trapézio formado pelas coordenadas: (0,0), (1,0), (0,1) e (FP rate, TP rate) da respectiva predição.

Nota-se um comportamento associado aos parâmetros $k$ e $t$. Quanto menor o número de clusters, maior será o raio da hiperesfera gerada pelo elemento mais distante do grupo e, portanto, menor será o valor de $t$ (sendo maior que 1) para atingir bons resultados.

Diferente da imagem anterior, é mostrado abaixo uma matriz de confusão não normalizada para cada uma das predições mensuradas, que são as mesmas representadas no espaço ROC. Vale ressaltar que a diagonal das matrizes estão invertidas, resultando no elemento $M_{0,0}$ como o TN e o elemento $M_{1,1}$ como o TP.

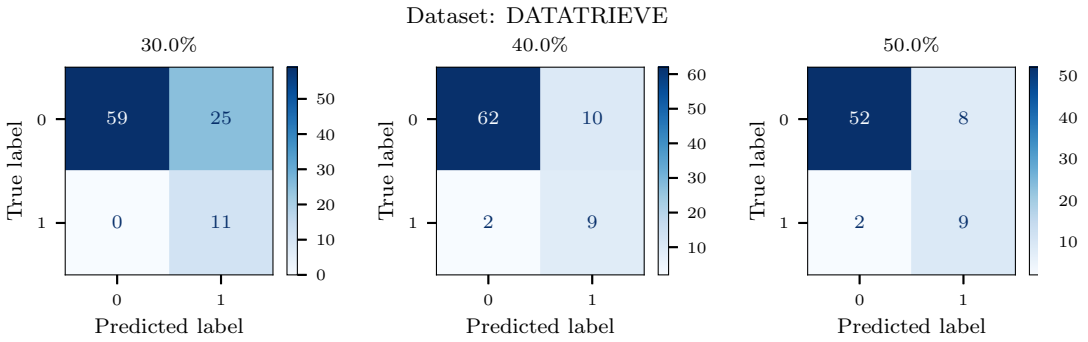


Figura 2: Matriz de confusão das melhores máquinas

Assim como na figura 1, a porcentagem diz respeito ao tamanho do conjunto de treinamento. Enfatizando, os valores True Positive, False Positive e F1-measure de cada matriz são: 30%(11, 25, 0.468), 40%(9, 10, 0.600) e 50%(9, 8, 0.643).

De uma maneira geral as máquinas apresentaram uma boa relação entre a $FP\ rate$ e $TP\ rate$, destacando a configuração que fez uso de 50% dos padrões de treinamento, obtendo o maior balanço entre os valores *precision* e *recall*.

Similarmente à figura 1, é apresentado a seguir, os resultados no espaço ROC referentes à base de dados CM1.

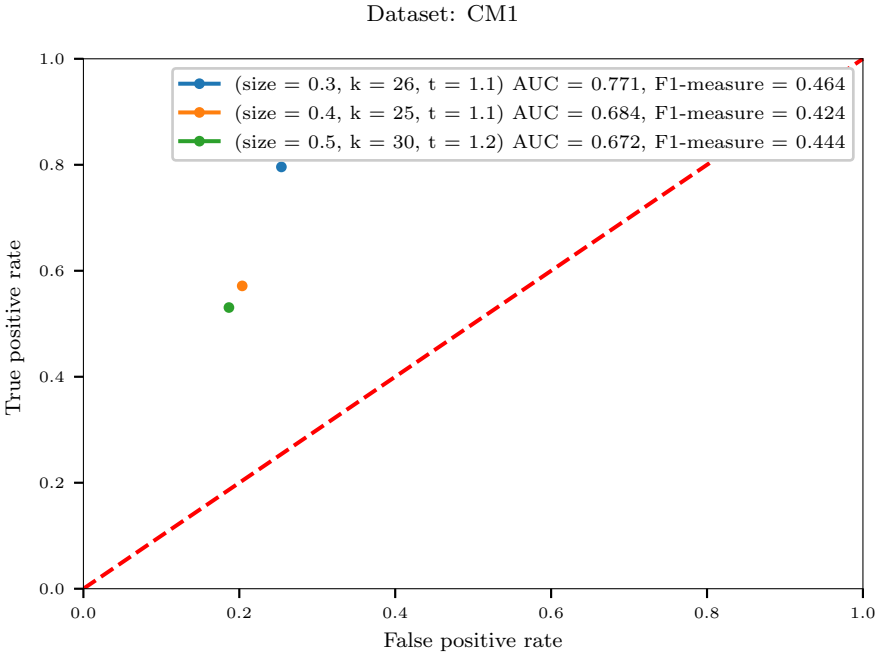


Figura 3: Espaço ROC - Melhores máquinas

Percebe-se que bons resultados foram obtidos quando os valores de $k$ permaneceram altos e os de $t$ próximos à 1. Ainda assim, se comparados às predições passadas, obteve-se métricas inferiores neste experimento.

Por fim, a figura 4 exibe os erros e acertos por classe através das matrizes de confusão para o *dataset* CM1.

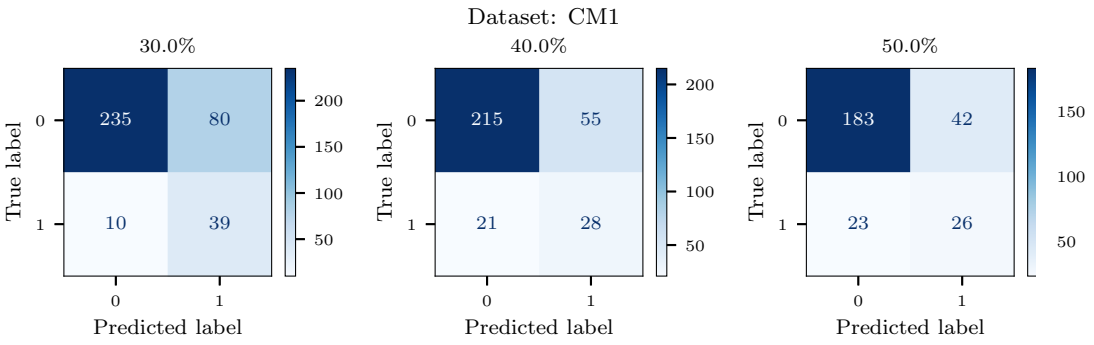


Figura 4: Matriz de confusão das melhores máquinas

Destacando, os valores True Positive, False Positive e F1-measure de cada matriz são: 30%(39, 80, 0.464), 40%(28, 55, 0.424) e 50%(26, 42, 0.444).

Dentro dos pares de parâmetros exaustivamente testados, aquele que resultou na maior área sob a curva ROC (AUC) teve o valor de *size* sendo 0.3, o menor entre as taxas de divisão dos padrões de treino.

# 5 Conclusões

Baseado nos resultados anteriores, o sistema proposto obteve números satisfatórios ao classificar instâncias da classe "com defeito", as anomalias. Vale destacar que para seu funcionamento desejável, há a necessidade de utilizar bons parâmetros de entrada, que como pôde ser visto, foram diferentes em cada base de dados. Outras análises requerem uma melhor divisão do conjunto de treinamento, visto que há várias maneiras de se obter as porcentagens previamente definidas.

# Bibliografia

[1] P. Datta. Characteristic Concept Representations, Tese de doutorado, Universidade da California, Irvine, 1997

[2] Pang-Ning Tan, Michael Steinbach e Vipin Kumar. Introduction to Data Mining, Pearson Education, 2006. Seção 10.5.