One Class Classification to Detect *PIWI-interacting* RNAs

Rafaela Vansan Ferreira[1], **Adriano Poupo de Campos**[1], **Ricardo Cerri**[2]

[1]Departamento de Estatística – Universidade Federal São Carlos
Rod. Washington Luís km 235 – São Carlos – SP – Brasil

[2]Departamento de Computação – Universidade Federal São Carlos
Rod. Washington Luís km 235 – São Carlos – SP – Brasil

rafaelavansan@gmail.com, cerri@ufscar.br, polpo@ufscar.br

Abstract. *The prediction of PIWI-Interacting RNAs (piRNAs) is a topic of interest related to small non-code RNAs, providing clues to understanding the mechanism of gametes generation. Several machine learning approaches have been proposed for the prediction of piRNAs, but improvements are still being sought. Due to the wide variety of non-coding RNAs, choosing which of them will be used as negative examples can be a hard task. In such scenarios, one class classifiers can be an option. Thus, this paper investigates the predictive power of one class classifiers in comparison with binary classifiers to predict piRNAs.*

Resumo. *A predição de PIWI-Interacting RNAs (piRNAs) é um tópico de interesse relacionado a pequenos RNAs não-codificantes, fornecendo pistas para a compreensão do mecanismo de geração de gametas. Várias abordagens de aprendizado de máquina foram propostas para a predição de piRNAs, mas ainda busca-se melhorias. Devido à grande variedade de RNAs não-codificantes, a escolha de quais deles serão utilizados como exemplos negativos pode ser uma tarefa trabalhosa. Em tais cenários, classificadores de uma classe podem ser uma opção. Assim, este trabalho investiga o poder de predição de classificadores de uma única classe em comparação aos classificadores binários para predição de piRNAs.*

1. Introdução

De acordo com o conceito molecular clássico, um gene é um segmento de ácido desoxirribonucleico (DNA). O DNA armazena a informação genética da grande maioria dos seres vivos. Assim como o DNA, o RNA (ácido ribonucleico) é um ácido nucleico. Essa molécula é essencial na síntese de proteínas, pois é capaz de expressar as informações presentes no DNA. De acordo com [Luo et al. 2016], os RNAs não codificantes (ncRNAs) são importantes moléculas de RNA funcionais, que não são traduzidas em proteínas [Claverie 2005, Mattick 2005]. RNAs não-codificantes são classificados como ncRNAs longos e ncRNAs curtos, de acordo com seus comprimentos. Os ncRNAs longos são geralmente maiores que 200 nucleotídeos [Huang et al. 2012, Xie et al. 2013]. Entre os ncRNAs curtos, aqueles com 20 a 32 nucleotídeos de comprimento são definidos como pequenos ncRNAs, como microRNAs (miRNAs) e piRNAs [Carmen et al. 2009].

Os piRNAs desempenham um papel importante no silenciamento dos elementos transponíveis (sequências de DNA capazes de se locomover de uma região a outra no

genoma de uma célula), envolvendo a formação de células germinativas, manutenção de células-tronco germinativas, espermatogênese e oogênese [Cox et al. 1998]. Os piRNAs trabalham no silenciamento epigenético dos elementos transponíveis, sendo um importante mecanismo de defesa do genoma para manter a integridade do mesmo. Por essa razão, os piRNAs desempenham um papel indispensável para a fertilidade das espécies [Hirakata and Siomi 2016, Iwasaki et al. 2015].

Devido à sua expressão limitada em gônadas e sua diversidade de sequências, os piRNAs têm sido a classe mais misteriosa de pequenos RNAs não codificadores que regulam o silenciamento do RNA, os quais colocaram dificuldades inesperadas à compreensão usual do conceito de gene. No intuito de explorar e compreender ainda mais sobre esses pequenos RNAs não codificantes, modelos de Aprendizado de Máquina são cada vez mais usados na solução de problemas da Bioinformática, que se caracteriza pelo imenso volume de dados.

Segundo [Alashwal et al. 2006], o método de classificação de uma classe é um caso especial do método de classificação de duas classes, no qual apenas dados de uma classe estão disponíveis e são bem amostrados. Essa classe é chamada de classe alvo. A outra classe é chamada de classe *outlier* e, em geral, os seus dados são escassos ou podem estar totalmente ausentes. Pode ser que seja muito difícil ou caro realizar medições para obtenção de dados da classe *outlier*.

Desenvolver ferramentas para a classificação de piRNAs é um trabalho complexo. Isso porque os piRNAs aparentam não dispor de conservação de estruturas secundárias, o que torna a utilização de métodos baseados em estruturas duvidosos [Zhang et al. 2011]. Como existem diferentes variações de RNAs não-codificantes, um pesquisador pode optar por considerar como exemplos negativos todos os possíveis RNAs não-codificantes que não são piRNAs. Isso pode gerar problemas, como por exemplo a dificuldade de se extrair características considerando diferentes comprimentos de sequências utilizadas como exemplos negativos. Outra alternativa de escolha de exemplos negativos pode ser a escolha de RNAs não codificantes com tamanhos semelhantes, porém exigindo um pré-processamento mais trabalhoso. Assim, a escolha de exemplos negativos é uma tarefa não trivial, exigindo tomadas de decisões na montagem dos conjuntos de dados, o que pode afetar os desempenhos dos classificadores. Nesse sentido, a utilização de métodos de classificação de uma classe se torna mais interessante.

O objetivo deste trabalho é investigar o desempenho dos classificadores de uma só classe, além de comparar com os classificadores binários. O restante desse documento está organizado da seguinte maneira: A Seção 2 apresenta a metodologia de desenvolvimento, com os algoritmos e os conjuntos de dados investigados. A Seção 3 mostra o desempenho dos métodos discutidos, através de gráfios e tabelas. A Seção 4 contém as conclusões e comentários finais.

2. Classificadores de Uma Classe

Há diversos algoritmos de aprendizado de máquina para classificação com uma única classe. Este trabalho tem foco principal nos seguintes algoritmos de classificação: K-vizinhos mais próximos, Máquinas de Vetores de Suporte e Autoencoder.

2.1. K-Vizinhos Mais Próximos

O algoritmo dos vizinhos mais próximos de uma classe (*One Class Nearest Neighbour* - OCNN) detecta exemplos não vistos da classe negativa, quando elas não estão presentes durante a fase de treino [Khan and Madden 2014]. O algoritmo OCNN localiza as regiões de alta e baixa densidade com base na vizinhança local de um exemplo de teste. Usando um limite de decisão, o algoritmo aceita ou rejeita um exemplo de teste como um membro da classe de destino, a qual, em geral, é a classe positiva. Em sua forma mais simples, o OCNN encontra o primeiro vizinho mais próximo de um exemplo de teste na classe da classe de destino e, em seguida, localiza o primeiro vizinho mais próximo desse vizinho na classe de destino. Se a proporção dessas distâncias for menor do que um limite definido pelo usuário, o exemplo será aceito como um membro da classe de destino. Esse método é simples e eficaz para detectar exemplos da classe negativa invisível. O OCNN não requer treinamento do classificador, possui poucos parâmetros, e pode ser facilmente usado com diferentes medidas de similaridade. Essas propriedades tornam o OCNN um método útil para problemas de classificação de uma classe.

2.2. Máquinas de Vetores de Suporte

As Máquinas de Vetores de Suporte *Support Vector Machines - SVMs* de uma classe surgiram como um método para adaptar as SVMs ao problema de classificação com uma classe [Schölkopf et al. 2001]. Essencialmente, depois de transformar os atributos via *kernel*, a origem é tratada como o único membro da segunda classe. Em seguida, usando os "parâmetros de relaxamento", o algoritmo separa a imagem de uma classe da origem. Em seguida, as técnicas convencionais de SVMs de duas classes são empregadas.

As SVMs de uma classe tratam o problema da seguinte maneira: suponha que um conjunto de dados tenha uma distribuição de probabilidade P no espaço de atributos. Encontre um subconjunto "simples" S do espaço de atributos de forma que a probabilidade de um ponto de teste de P ficar fora de S seja limitada por algum valor especificado a priori. Supondo que há um conjunto de dados extraído de uma distribuição de probabilidade subjacente P, é necessário estimar um subconjunto "simples" S do espaço de entrada, de forma que a probabilidade de um ponto de teste de P ficar fora de S seja limitada por alguma priori especificada V $\in$ (0, 1).

A solução para esse problema é obtida estimando uma função f que é positiva em S e negativa no complemento S. Em outras palavras, [Schölkopf et al. 2001] desenvolveram um algoritmo que retorna uma função f que recebe o valor $+1$ em uma região capturando a maioria dos vetores de dados, e -1 em outra região. Essa "pequena" região será a região dos *outliers*, e a maior parte da região é a classe alvo. O algoritmo pode ser resumido como o mapeamento dos dados em um espaço de recurso H usando uma função de *kernel* apropriada e, em seguida, tentando separar os vetores mapeados da origem com a margem máxima, como mostra a figura 1. A função de separação dos vetores mapeados é então:

$$f(x) = \begin{cases} +1, & x \in S \\ -1, & x \in \bar{S} \end{cases}$$

Sejam $x_1, x_2, \ldots, x_l$ os dados de treino pertencentes a uma classe X, onde X é um subconjunto compacto de $\mathbb{R}^N$. Seja $\Phi : X \rightarrow H$ uma função de *kernel* (*kernel map*)

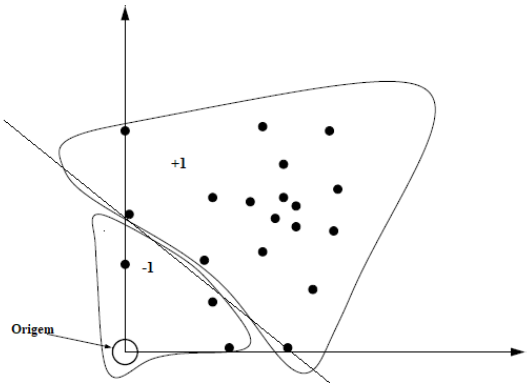

Figura 1. Representação do SVM de uma classe [Manevitz and Yousef 2001]

que mapeia os dados de treinamento para outro espaço. Para separar os dados da origem, é preciso resolver o seguinte problema quadrático:

$$min\frac{1}{2}\left\|w\right\|^2 + \frac{1}{vl}\sum_{i=1}^{l}\epsilon_i - \rho$$

sujeito a:

$$(w.\Phi(x_i)) > \rho - \epsilon_i$$

com $i = 1, 2, \ldots, l$, $\epsilon_i \geq 0$, e $v \in (0, 1)$.

Se w e ρ resolvem esse problema, então a função de decisão para classificação é dado por

$$f(x) = sign((w.\Phi(x)) - \rho),$$

função que será positiva para a maior parte dos exemplos x_i dos dados de treino.

2.3. AutoEncoder

As Redes Neurais Artificiais são modelos computacionais inspirados pelo sistema nervoso central. A estrutura básica de uma rede neural é o neurônio artificial, que consiste em modelo matemático inspirado no neurônio biológico, com a função de receber entradas (vindas de outros neurônios ou dos exemplos de entrada), e computar uma saída (que pode servir de entrada para outros neurônios).

As Redes Neurais Artificiais possuem os seguintes componentes: uma camada de entrada; uma quantidade arbitrária de camadas ocultas; uma camada de saída; um conjunto de pesos e vieses entre cada camada e uma escolha da função de ativação para cada camada oculta.

O algoritmo tenta uma aproximação para a função identidade, de modo a produzir os atributos dos exemplos de treino reconstruídos, $\bar{\mathrm{x}}$, de maneira que sejam semelhante

aos exemplos originais ($\mathbf{x}$). A função identidade parece uma função particularmente trivial de se aprender; mas ao colocar restrições na rede, como limitar o número de unidades ocultas, podemos descobrir uma estrutura interessante sobre os dados.

Para construir um AutoEncoder, são necessárias uma função de codificação, uma função de decodificação, e uma função para medir a quantidade de perda (*loss function*) de informação entre os dados originais e os dados reconstruídos pela rede. A função de perda escolhida neste trabalho foi o erro quadrático médio (EQM).

Para compreender o funcionamento da classificação através do AutoEncoder, é necessário supor que a distribuição conjunta dos atributos difere entre a classe positiva e a classe negativa. Dessa forma os padrões identificados pelo AutoEncoder nos atributos dos piRNAs não estarão presentes para os pseudo-piRNAs.

O AutoEncoder é treinado apenas com os atributos de uma classe. Vamos supor que seja a classe positiva. Então temos o erro de reconstrução da classe positiva na fase de treino. Uma vez que o algoritmo é treinado apenas com a classe positiva, na fase de teste haverá maior erro de reconstrução no AutoEncoder para os exemplos negativos. A partir do erro de reconstrução, basta determinar um ponto de corte k no qual a partir deste, todos os exemplos com erro de reconstrução maior que k serão classificados como sendo da classe negativa [Falbel 2018]. Em suma, o método pode ser abordado como classificação de uma classe pois define-se uma classe e o resto se classifica como uma segunda categoria. Na implementação, foi utilizada a mediana do erro quadrático médio entre os dados de treino e a reconstrução do AutoEncoder como ponto de corte k. Posteriormente, calculou-se o erro quadrático médio da reconstrução do teste e a classificação é realizada da seguinte forma:

$$Classe_i = \begin{cases} 1, & \text{se } EQM_i \leq k \\ 0, & \text{caso contrário} \end{cases}$$

2.4. Conjunto de Dados

Os conjuntos de dados utilizados são de sequências de RNA não codificantes da espécie *Drosophila*. Os exemplos positivos são piRNAs e os exemplos negativos são pseudo-piRNAs. Para obtenção das sequência negativas, [Luo et al. 2016] propuseram o seguinte procedimento: foi realizado o download de um grande número de sequencias de não-piRNA ncRNAs do NONCODE v. 3.0 [Xie et al. 2013], e foi removido não-piRNA ncRNAs cujos tamanhos são menores do que o tamanho mínimo do tamanho do exemplo positivo. Então, foi aleatoriamente cortado pequenas partes de sequências como candidatos a pseudo-piRNA. Depois de alinhá-los a transposons de Drosophila, outro número de não redundante candidatos a pseudo piRNA foi obtido. Então selecionaram-se aleatoriamente algumas sequências pseudo piRNA candidatas para simular o número e o comprimento distribuição de exemplos positivos. Então, não redundantes pseudo piRNAs foram gerados como os exemplos negativos.

A partir dos dados coletados, atributos foram extraídos utilizando a ferramenta Pse-in-One-2.0 (em sua versão local) [Liu et al. 2017]. Como atributos, foram utilizados k-mers. O termo k-mer tipicamente se refere a todas as subsequências DNA ou RNA de tamanho k que estão contidas em uma sequência. Dada uma sequência de DNA ou RNA,

é calculada a ocorrência de ácidos nucleicos vizinhos de tamanho k que diferem em m posições da sequência original. Foram extraídos 340 atributos considerando as seguintes combinações de m e k: $m = 0$ e $k = 1$, $m = 1$ e $k = 2$, $m = 1$ e $k = 3$, e $m = 1$ e $k = 4$.

Para exemplificar um k-mer, considere a sequência de DNA AGATCGAGTG. Escolhendo $k = 3$, temos os seguintes 3-mers: AGA, GAT, ATC, TCG, CGA, GAG, AGT e GTG. Assim, um atributo dos dados é, por exemplo, o número de subsequências "AGA"presentes na sequência de DNA.

No total, o conjunto de dados possui 18.428 exemplos, divididos em 9.214 exemplos positivos (piRNAs) e 9.214 exemplos negativos (pseudo-piRNAs). Desse total, 60% foi considerado como treino, 15% como validação e 25% como teste. A Tabela 1 apresenta a distribuição de exemplos positivos e negativos nos dados. Para um melhor desempenho do modelo, há um balanceamento entre a quantidade de exemplos positivos e negativos no conjunto de dados.

	Num. exemplos positivos	Num. exemplos negativos
Treino	5.528	5.528
Validação	2.304	2.303
Teste	1.382	1.383

Tabela 1. Distribuição de exemplos positivos e negativos nos dados

2.5. Medidas de Avaliação

Para avaliar o desempenho de um método de classificação, é comum avaliar-se a matriz de confusão:

		Valor verdadeiro		
		Positivo	Negativo	Total
Valor predito	Positivo	VP	VN	$VP + VN$
	Negative	FP	FN	$FP + FN$
	Total	$VP + FP$	$VN + FN$	T

Tabela 2. Matriz de confusão

em que VP indica verdadeiros positivos, FP falsos positivos, VN verdadeiros negativos, FN falsos negativos, e T o total e exemplos.

Com a matriz de confusão, a avaliação do desempenho é feita com as seguintes métricas: acurácia, precisão, revocação, medida-F, sensibilidade, especificidade, coeficiente de Kappa e curva ROC. A curva ROC é um gráfico que exibe simultaneamente dois tipos de erros para todos os possíveis pontos de corte. O desempenho geral de um classificador, resumido em todos os possíveis pontos de corte, é dado pela área sob a curva ROC (AUC). Uma curva ROC ideal vai abraçar o canto superior esquerdo, então quanto maior a área sob a curva ROC (AUC), melhor o classificador [James et al. 2013]. É justamente este canto superior direito que vamos avaliar, no caso do SVM, para validar o modelo e obtermos indícios de um sobreajuste ou um subajuste.

2.6. Componentes principais

Como há 340 atributos no conjuntos de dados, é necessário utilizar um método que reduz a dimensionalidade do dados. O método de componentes principais realiza uma transformação dos atributos de entrada originais em um menor número de atributos não correlacionados e, portanto, mais significativos [Spinosa and de Carvalho 2005]. Desse modo, as componentes foram utilizadas para substituir os atributos na parte de classificação deste trabalho, com exceção ao método do Autoencoder, que já é uma redução de dimensão. Além disso, as componentes permitem a visualização gráfica do comportamento dos dados.

A porcentagem de variância explicada acumulada por todas as componentes é 100%. Assim, é possível determinar quantas componentes são necessárias para se ter uma porcentagem de variância explicada acumulada alta, em torno de 90% ou mais.

3. Resultados

Esta seção apresenta os resultados experimentais obtidos. Todos os algoritmos foram implementados utilizando a linguagem Python 3.6.6 tendo como apoio diversas bibliotecas, mas principalmente scikit-learn [Pedregosa et al. 2011] e keras [Chollet et al. 2015].

3.1. Análise descritiva

As primeiras 60 componentes principais explicam em torno de 95% da variabilidade dos 340 atributos. Assim, este trabalho foi feito utilizando as 60 primeiras componentes principais ao invés dos dados originais.

A Figura 2 mostra a disposição dos dados da classe positiva e negativa nas duas primeiras componentes principais (primeiro plano fatorial). É possível perceber não há grandes diferenças no comportamento dos dados de treino, validação e teste. Ou seja, nos casos casos o número de observações positivas e negativas é balanceado. As observações positivas possuem mais *outliers* e as negativas possuem menor variabilidade.

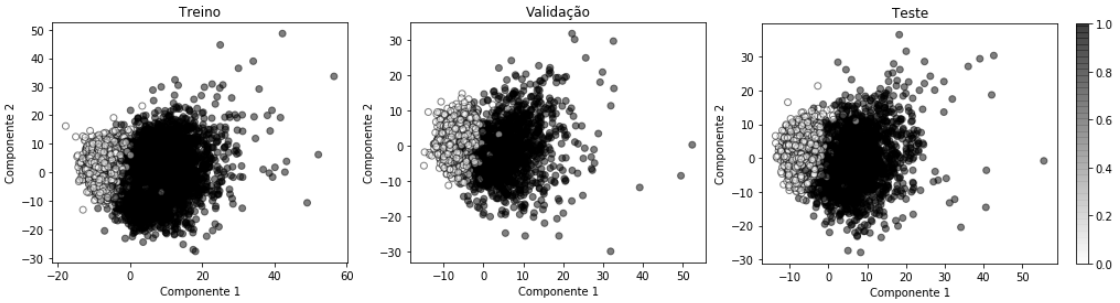

Figura 2. Gráficos da primeira componente pela segunda componente principal

3.2. Resultados com o Algoritmo dos K Vizinhos Mais Próximos

Os algoritmos KNN de duas classes e de uma classe foram executados variando o valor de K, com mínimo 1 e máximo 100. O KNN de uma classe foi primeiramente treinado só com a classe negativa (KNN de uma classe (0)) e depois só com a classe positiva (KNN de uma classe (1)). No caso do KNN de uma classe foram calculadas as acurácias para K=1 até K=100, para investigar melhor seu comportamento.

Analisando a acurácia do KNN de uma classe (0) para cada valor de K, de 1 até 90, o maior valor de acurácia foi observado com K = 90. As medidas de desempenho foram calculadas na predição feita com K = 90. Para o KNN de uma classe (1) o máximo valor de acurácia foi observado quando K = 100, e as medidas de avaliação para a predição feita com K=100 estão reportadas na Tabela 3.

Método	Coef. de Kappa	Acurácia	Precisão	Revocação	Medida-F
KNN de duas classes	0,874	0,937	0,979	0,893	0,934
KNN de uma classe (0)	0,141	0,570	0,543	0,896	0,676
KNN de uma classe (1)	0,041	0,521	0,519	0,557	0,538

Tabela 3. Métricas calculadas com a predição para o teste

Para esses dados, o desempenho do KNN de uma classe é inferior ao desempenho do KNN de duas classes. Isso porque o KNN de uma classe tem a tendência de classificar mais os exemplos do teste na classe com que ele foi treinado. Então, por exemplo, no KNN de uma classe treinado apenas com a classe negativa, há uma tendência de classificar mais os exemplos de teste como exemplos da classe negativa.

3.3. Resultados com os Algoritmos de Máquinas de Vetores de Suporte

Primeiramente, testou-se o desempenho na classificação dos exemplos de teste do método de classificação de máquinas de vetores de suporte de uma classe usando a classe positiva na fase de treino, cujo resultados estão reportados na figura 3 e nas tabelas 4 e 5. Posteriormente, testou-se o desempenho usando a classe negativa na fase de treino, cujo resultados estão reportados na figura 4 e nas tabelas 6 e 7. Todos os hiperparâmetros envolvidos foram escolhidos por validação cruzada, por meio da estratégia *5-fold cross validation*.

É de interesse saber se nos pontos de corte ótimos, que correspondem ao máximo da diferença entre a senbilidade e especifidade, da curva ROC da validação também apresentam alta especificidade e sensibilidade no teste. Assim, avaliando nos pontos de cortes específicos, obteve-se os resultados apresentados nas tabelas 4 e 6.

Método	Ponto de corte	Sensibilidade	Especificidade
SVM de duas classes	0,4	0,938	0,962
SVM de uma classe	10778,9	0,93	0,96

Tabela 4. Desempenho do SVM e OC-SVM treinado com a classe negativa

Método	Coef. de Kappa	Acurácia	Precisão	Revocação	Medida F
SVM de duas classes	0,900	0,950	0,969	0,929	0,950
SVM de uma classe	0,854	0,927	0,894	0,968	0,930

Tabela 5. Medidas gerais de avaliação, OC-SVM treinado com a classe negativa

Para o SVM de uma classe, a acurácia relativamente alta juntamente com revocação baixa indicam que há um número considerável de observações sendo falso positivas, ou seja, sendo classificadas como positivas quando na verdade são negativas.

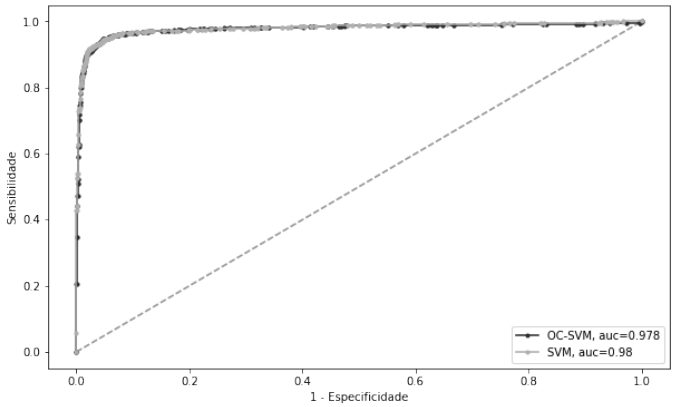

Figura 3. Curva ROC da predição do SVM e OC-SVM treinado com a classe negativa

Método	Ponto de corte	Sensibilidade	Especificidade
SVM de duas classes	0,44	0,934	0,964
SVM de uma classe	10955,3	0,589	0,875

Tabela 6. Desempenho do SVM e OC-SVM treinado com a classe positiva

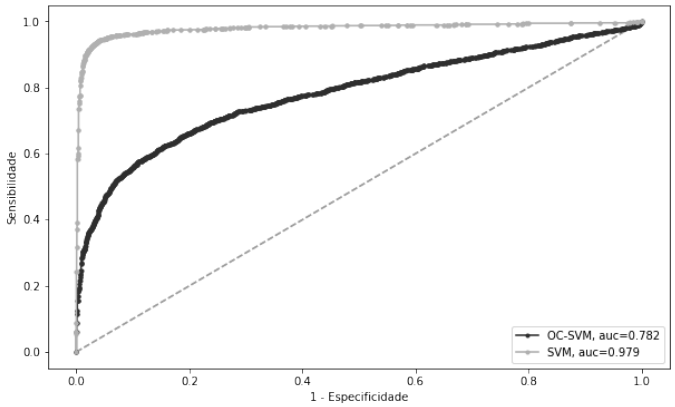

Figura 4. Curva ROC do SVM e OC-SVM treinado com a classe positiva

Método	Coef. de Kappa	Acurácia	Precisão	Revocação	Medida F
SVM de duas classes	0,898	0,949	0,967	0,929	0,948
SVM de uma classe	0,463	0,732	0,830	0,582	0,685

Tabela 7. Medidas gerais de avaliação, OC-SVM treinado com a classe positiva

Assim como no caso do KNN, o SVM de uma classe possui desempenho menor quando na fase de treino são utilizados apenas os dados da classe positiva. Nesse caso, o erro no teste é classificar a maior parte dos dados como positivos (há mais falso positivos).

3.4. Autoencoder

O Autoencoder foi implementado utilizando os atributos do treino e da validação, apenas da classe negativa. Ou seja, o algoritmo reconstruiu os exemplos da classe negativa utilizando os dados de treino e validação. Posteriormente, utilizou-se os dados de teste, com

a classe positiva e negativa, para fazer a classificação.

O modelo de Autoencoder utilizado possui 340 variáveis de entrada, 10 neurônios na primeira camada oculta, 128 neurônios na segunda camada oculta, e 340 saídas (associadas às variáveis de entrada). Essa foi a melhor configuração obtida após testes preliminares não exaustivos no conjunto de validação.

Os pesos entre a camada de entrada e a primeira camada oculta, juntamente com os pesos entre a primeira e a segunda camadas ocultas, caracterizam o encoder que resume informações. Os pesos entre a segunda camada oculta e a camada de saída caracterizam o decoder que decodifica (recupera) os valores apresentados na camada de entrada. Todas as camadas são densas, ou seja, todos os neurônios de uma camada são conectados com todos os neurônios da camada anterior. A camada de entrada e a primeira camada oculta possuem função de ativação *reLU (Rectified Linear Unit)*, dada por

$$R(z) = max(0, z)$$

em que z é o potencial de ativação do neurônio. A função de ativação utilizada na segunda camada oculta e na camada de saída foi a logística, dada por

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

A classificação foi feita comparando-se o erro quadrático médio do treino com o erro no teste. A Figura 5 mostra como se comporta o erro no treino e na validação conforme as épocas. O modelo do Autoencoder também mostrou um ótimo desempenho, prevendo bem as observações do teste.

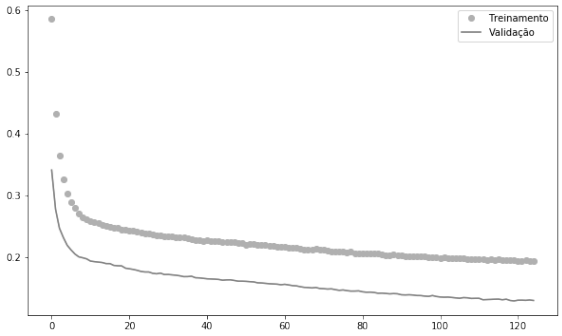

Figura 5. Valores do erro quadrático médio no treino e na validação, por época

A tabela 8 mostra as medidas de avaliação da classificação feita pelo Autoencoder nos exemplos de teste. É possível concluir seu desempenho é bom pois é tanto específico quanto sensível.

Método	Coef. de Kappa	Acurácia	Precisão	Revocação	Medida-F
Autoencoder	0,749	0,874	0,862	0,891	0,876

Tabela 8. Métricas calculadas com a predição para o teste

4. Considerações Finais

Este trabalho investigou o desempenho dos classificadores de uma classe para predição de piRNAs uma vez que há dificuldade para obtenção da classe negativa para predição de piRNAs. Além disso, este trabalho comparou o desempenho de classificadores binários com os classificadores de uma classe.

Primeiramente, a Figura 2 da análise descritiva mostra que as duas classes, piRNA e pseudo-piRNA, são diferenciáveis. Porém existem pontos nos dados que existem pontos sobrepostos que tornam a classificação mais difícil. Em geral, quanto mais separadas são as duas classes melhores são os desempenhos dos classificadores. Assim, como esperado, nenhum método resultou em uma classificação perfeita.

É importante ressaltar que, como o método de classificação de uma classe usa apenas os dados da classe positiva na fase de treino, os classificadores de uma classe foram treinados com metade dos dados dos classificadores de duas classes, o que pode ser considerada uma desvantagem para aqueles classificadores. Todavia, observou-se que no caso dos classificadores de Máquinas de Vetores de Suporte, o método de uma classe (treinado com a classe negativa) é quase tão bom quanto o de duas classes.

No caso do SVM de uma classe, observou-se que, quando treinado com a classe negativa, seu desempenho é ótimo tanto no teste quanto na validação. Especificamente no ponto de corte de ótimo do teste, ele também tem uma boa predição na validação, mostrando que não estamos em um caso de super ajuste e dando mais credibilidade ao modelo.

O Autoencoder foi treinado apenas com a classe negativa, visando o maior desempenho. O algoritmo mostrou alta acurácia, sensibilidade e especificidade.

Como trabalhos futuros é interessante implementar métodos como Naive Bayes de uma classe e Árvore de Decisão de uma classe. Além disso, é interessante investigar como escolher hiperparâmetros em dados que de fato uma das classes é escassa ou não existe na fase de treino.

Referências

Alashwal, H., Deris, S., and Othman, R. M. (2006). One-class support vector machines for protein-protein interactions prediction. *International Journal of Biological and Medical Sciences*, 1(2).

Carmen, L., Michela, B., Rosaria, V., Gabriella, M., et al. (2009). Existence of snorna, microrna, pirna characteristics in a novel non-coding rna: x-ncrna and its biological implication in homo sapiens. *Journal of Bioinformatics and Sequence Analysis*, 1(2):031–040.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Claverie, J.-M. (2005). Fewer genes, more noncoding rna. *Science*, 309(5740):1529–1530.

Cox, D. N., Chao, A., Baker, J., Chang, L., Qiao, D., and Lin, H. (1998). A novel class of evolutionarily conserved genes defined by piwi are essential for stem cell self-renewal. *Genes & development*, 12(23):3715–3727.

Falbel, D. (2018). Tensorflow for r: Predicting fraud with autoencoders and keras.

Hirakata, S. and Siomi, M. C. (2016). pirna biogenesis in the germline: from transcription of pirna genomic sources to pirna maturation. *Biochimica et Biophysica Acta (BBA)-Gene Regulatory Mechanisms*, 1859(1):82–92.

Huang, Y., Liu, N., Wang, J. P., Wang, Y. Q., Yu, X. L., Wang, Z. B., Cheng, X. C., and Zou, Q. (2012). Regulatory long non-coding rna and its functions. *Journal of physiology and biochemistry*, 68(4):611–618.

Iwasaki, Y. W., Siomi, M. C., and Siomi, H. (2015). Piwi-interacting rna: its biogenesis and functions. *Annual review of biochemistry*, 84:405–433.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer.

Khan, S. S. and Madden, M. G. (2014). One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374.

Liu, B., Wu, H., and Chou, K.-C. (2017). Pse-in-one 2.0: An improved package of web servers for generating various modes of pseudo components of dna, rna, and protein sequences. 09:67–91.

Luo, L., Li, D., Zhang, W., Tu, S., Zhu, X., and Tian, G. (2016). Accurate prediction of transposon-derived pirnas by integrating various sequential and physicochemical features. *PloS one*, 11(4):e0153268.

Manevitz, L. M. and Yousef, M. (2001). One-class svms for document classification. *Journal of machine Learning research*, 2(Dec):139–154.

Mattick, J. S. (2005). The functional genomics of noncoding rna. *Science*, 309(5740):1527–1528.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471.

Spinosa, E. J. and de Carvalho, A. C. (2005). Combining one-class classifiers for robust novelty detection in gene expression data. In *Brazilian Symposium on Bioinformatics*, pages 54–64. Springer.

Xie, C., Yuan, J., Li, H., Li, M., Zhao, G., Bu, D., Zhu, W., Wu, W., Chen, R., and Zhao, Y. (2013). Noncodev4: exploring the world of long non-coding rna genes. *Nucleic acids research*, 42(D1):D98–D103.

Zhang, Y., Wang, X., and Kang, L. (2011). A k-mer scheme to predict pirnas and characterize locust pirnas. *Bioinformatics*, 27(6):771–776.