Relatório Trabalho 3

Computação Bioinspirada
Algoritmo Imunológico de Seleção Negativa
Luíza W. Diapp GRR20221252
Nádia Lobkov GRR———

Introdução

Não feito ainda

Algoritmo Imunológico de Seleção Negativa (NSA)

O Algoritmo Imunológico de Seleção Negativa (NSA) é inspirado nos princípios do sistema imunológico adaptativo humano, especialmente nos mecanismos de distinção entre *self* e *non-self*, conforme originalmente proposto por Forrest et al. [1]. Desde então, o NSA tem sido amplamente estudado e aplicado em tarefas de detecção de anomalias e sistemas de detecção de intrusão (IDS), destacando-se por sua capacidade de identificar padrões não observados previamente [2].

1. Definição do Conjunto *Self*

O algoritmo parte do pressuposto de que os dados de treinamento representam exclusivamente o comportamento legítimo do sistema, denominado *self*. A modelagem desse espaço é essencial, pois fundamenta a distinção entre padrões aceitáveis e potenciais anomalias. Estudos iniciais mostraram a importância crítica dessa etapa para a eficácia de qualquer sistema imunológico artificial [3].

2. Geração dos Detectores

A etapa de geração de detectores constitui o núcleo do NSA. Detectores sintéticos são criados e comparados com todos os padrões do espaço *self*. Se apresentarem similaridade superior a um limiar de afinidade τ, são descartados, seguindo o princípio da seleção negativa observado biologicamente [1]. Métodos de otimização e abordagens estocásticas para geração de detectores foram posteriormente explorados para melhorar eficiência e cobertura [4].

Os detectores que não apresentam correspondência com o *self* formam o conjunto *non-self*, que será utilizado na fase de detecção. A literatura demonstra que o tamanho e a diversidade desse conjunto influenciam diretamente a taxa de detecção e o número de falsos positivos [5].

3. Processo de Detecção

Durante a detecção, novas instâncias são comparadas com o conjunto de detectores maduros. Caso uma instância apresente correspondência com um detector *non-self*, ela é classificada como anômala. Essa abordagem possui a vantagem de não exigir exemplos de ataques durante o

treinamento, característica que torna o NSA especialmente eficaz em detectar intrusões inéditas, incluindo ataques do tipo *zero-day* [6].

Em síntese, o NSA oferece um mecanismo robusto e biologicamente inspirado para detecção de anomalias, apresentando vantagem significativa em cenários dinâmicos e de alta incerteza. Sua flexibilidade e capacidade de adaptação tornaram-no uma das técnicas imunológicas mais estudadas na segurança computacional.

Dataset NSL-KDD

A detecção de intrusões é um elemento fundamental da segurança de redes, e a avaliação de modelos de *Machine Learning* nessa área requer datasets representativos e bem estruturados. Nesse contexto, o conjunto de dados *NSL-KDD* se destaca como um dos benchmarks mais utilizados.

O NSL-KDD foi desenvolvido como uma versão aprimorada do tradicional *KDD Cup 99*, que, apesar de amplamente empregado, apresenta limitações conhecidas, como redundância excessiva, forte desbalanceamento entre classes e distribuição pouco realista das instâncias de ataque [7]. Tais problemas comprometiam a validade das avaliações, levando a métricas superestimadas e modelos com baixa capacidade de generalização.

Com o objetivo de mitigar essas deficiências, Tavallaee et al. [8] propuseram o NSL-KDD, composto por registros de conexões de rede filtrados, balanceados e rotulados como *normal* ou como diferentes tipos de ataque. Ao remover duplicações e melhorar a representatividade das classes, o dataset oferece um cenário de teste mais realista e desafiador.

Em síntese, o NSL-KDD constitui um avanço significativo em relação ao KDD Cup 99 e permanece uma referência central em estudos de detecção de intrusões e segurança de redes.

Estrutura dos Atributos

Cada registro do NSL-KDD possui 41 atributos descritivos e uma coluna de rótulo, totalizando 42 colunas. Esses atributos são tradicionalmente agrupados em três categorias:

1. Atributos Intrínsecos da Conexão (Basic Features)

Descrevem características essenciais da conexão TCP/IP, tais como: duração da conexão, protocolo utilizado, serviço de destino, estado da conexão, quantidade de bytes transferidos, entre outros indicadores de anomalias básicas.

2. Atributos Baseados no Conteúdo (Content Features)

Extraem informações a partir do conteúdo da conexão, sendo úteis para detecção de ataques das categorias R2L e U2R. Entre os principais atributos encontram-se: tentativas de login mal-sucedidas, indicadores de atividades suspeitas, número de comprometimentos detectados e

obtenção de privilégios elevados.

3. Atributos Estatísticos de Tráfego (Time-based e Host-based Traffic Features)

Avaliam padrões de tráfego em janelas temporais ou nas últimas conexões realizadas por um mesmo host. Exemplos incluem o número de conexões recentes para o mesmo serviço, taxas de erro, e contagem de conexões por endereço de destino.

4. Rótulos e Classes de Ataques

A quadragésima segunda coluna contém o rótulo da instância, indicando se a conexão é normal ou se corresponde a um tipo específico de ataque. Mas na análise feita nesse trabalho essa coluna foi tratada binária, ou seja, se era um ataque o valor da coluna era 1, caso contrário era 0. Assim, não houve distinção do tipo de ataque durante o treinando do dataset.

Implementação

A implementação do Algoritmo de Seleção Negativa (NSA) foi realizada em Python, utilizando o dataset NSL-KDD como base de experimentação. O fluxo é composto por etapas de pré-processamento dos dados, construção do espaço *self*, geração dos detectores (*non-self*) e, por fim, classificação e avaliação.

Pré-processamento e Seleção de Atributos

Inicialmente, os arquivos `KDDTrain+.txt` e `KDDTest+.txt` são carregados em estruturas *DataFrame* do *pandas*, utilizando o conjunto padrão de 42 colunas do NSL-KDD. Em seguida, a coluna `attack`, que contém o tipo específico de ataque (por exemplo, *neptune*, *smurf*, *guess_passwd*) ou o rótulo *normal*, é convertida em um rótulo binário `binary_label`: instâncias normais recebem valor 0 e qualquer tipo de ataque recebe valor 1. Dessa forma, o problema é tratado como detecção de anomalias binária (*normal* vs. *ataque*), sem distinção entre as diferentes categorias de ataque.

Após a criação do rótulo binário, as colunas `attack` e `level` são removidas do conjunto de atributos, restando apenas as características destinadas à construção do espaço de características utilizado pelo NSA.

Tratamento de Atributos Categóricos e Exclusão de Colunas Binárias

Entre as colunas originais do NSL-KDD, três são categóricas com alta cardinalidade ou semântica simbólica: `protocol_type`, `service` e `flag`. Em vez de aplicar codificação *one-hot* (via `get_dummies`), o que resultaria em um grande número de colunas binárias esparsas e aumentaria significativamente a dimensionalidade, optou-se pelo uso de uma estratégia simples de *embedding* baseada em frequência.

Para cada coluna categórica em `protocol_type`, `service` e `flag`, calcula-se a frequência relativa de cada categoria no conjunto de treino, normalizada para o intervalo $[0, 1]$. Cada valor categórico é então mapeado para esse escalar normalizado, produzindo novas colunas contínuas `protocol_type_emb`, `service_emb` e `flag_emb`. As colunas categóricas originais são então removidas, e apenas suas versões embarcadas são mantidas.

Adicionalmente, um subconjunto de colunas originalmente binárias foi excluído do modelo. A presença de muitas variáveis estritamente binárias, especialmente quando combinadas com codificações *one-hot*, tende a gerar um espaço de atributos altamente esparso, o que é pouco desejável para algoritmos baseados em distância como o NSA. Dessa forma, buscou-se manter um espaço de características predominantemente numérico contínuo (incluindo os *embeddings* escalares), mais adequado ao cálculo de distâncias L1 e à geração de detectores em um hiperespaço compacto.

Normalização dos Atributos Numéricos

Para as demais colunas numéricas, foi aplicada normalização Min–Max utilizando a classe `MinMaxScaler`, da biblioteca `scikit-learn`. O escalonamento é ajustado apenas sobre o conjunto de treino e posteriormente aplicado tanto ao treino quanto ao teste, garantindo que cada atributo numérico seja mapeado para o intervalo $[0, 1]$.

Essa etapa é particularmente importante no contexto do NSA, pois o algoritmo utiliza distância de Manhattan (L1)* para medir similaridade. Sem normalização, atributos com maior amplitude numérica dominariam o cálculo de distância, distorcendo a noção geométrica de proximidade entre amostras. Já as colunas de *embedding* categórico, por definição, também estão em $[0, 1]$, de modo que todo o espaço de características final se encontra em uma escala comparável. O conjunto final de atributos (`X_train_final` e `X_test_final`) é construído concatenando-se as colunas numéricas normalizadas com as colunas categóricas que estão representadas por sua frequência, resultando em um vetor de características totalmente numérico e de dimensão fixa para cada instância.

*Escolha por Distância de Manhattan

Adotou-se a distância de Manhattan (L1) como métrica de similaridade em vez da distância Euclidiana (L2) ou da distância de cosseno. A L1 é definida como a soma das diferenças absolutas entre as coordenadas dos vetores e, em espaços de maior dimensão, tende a preservar melhor o contraste entre amostras do que a L2, que sofre mais com a concentração das distâncias. Além disso, a L1 é computacionalmente simples e menos sensível a valores extremos em um único atributo, pois não eleva as diferenças ao quadrado. Por outro lado, a distância de cosseno é mais adequada quando se deseja comparar apenas a direção dos vetores, desconsiderando sua magnitude, o que não é o caso deste trabalho, em que o desvio absoluto em cada atributo (após normalização para $[0, 1]$) é informativo para caracterizar o quanto um padrão se afasta do comportamento *self*.

Construção do Espaço *Self* e Geração dos Detectores

Para a etapa imunológica, apenas as instâncias rotuladas como *normais* (rótulo 0 em `binary_label`) são utilizadas para definir o espaço *self*. Essas amostras compõem a matriz `self_samples`, a partir da qual são calculados os valores mínimos e máximos de cada atributo (vetores `min_vals` e `max_vals`). Esses limites são usados para amostrar candidatos a detectores de forma uniforme em todo o hiperespaço de características normalizado.

A função `generate_vdetectors` implementa a geração de detectores no estilo V-detector, utilizando distância Manhattan. Em cada iteração, é amostrado um candidato aleatório dentro dos limites de cada atributo. Para esse candidato, calcula-se a distância L1 até todas as amostras *self*, e obtém-se a menor distância $d_{\min}$. O candidato só é aceito como detector se $d_{\min}$ for maior do que um limiar pré-definido, denominado *self radius* (`self_radius`). Quando aceito, o detector é armazenado com centro igual ao vetor candidato e raio igual a $d_{\min}$, caracterizando um V-detector cujo raio se estende até a instância *self* mais próxima.

No experimento apresentado, foram solicitados 7500 detectores (`num_detectors = 7500`) e adotado um raio mínimo `self_radius` = 1.4. Esses hiperparâmetros representam um compromisso entre cobertura do espaço *non-self*, taxa de falsos positivos e custo computacional. Um número muito pequeno de detectores tende a produzir baixa cobertura do espaço de anomalias, reduzindo a taxa de detecção. Por outro lado, um número excessivamente grande eleva o custo de classificação e pode levar a sobreposição redundante de regiões. De forma semelhante, um valor muito baixo de raio exigiria muitos detectores para cobrir regiões relevantes, enquanto um valor muito alto aumentaria o risco de invasão da região *self*, resultando em falsos positivos. Após experimentação empírica, o valor de 1.4 na escala L1 normalizada mostrou-se adequado para produzir detectores suficientemente afastados das amostras normais, mas ainda capazes de cobrir uma fração significativa do espaço de características associado a comportamentos anômalos.

Classificação e Avaliação

Na fase de detecção, cada amostra do conjunto de teste é representada por seu vetor em `X_test_final` e comparada ao conjunto de detectores gerados. A função `classify_with_detectors` implementa a seguinte lógica: para cada instância, calcula-se a distância L1 até o centro de cada detector; caso a distância até algum detector seja menor ou igual ao raio correspondente, a instância é classificada como *ataque* (rótulo 1); caso contrário, é classificada como *normal* (rótulo 0).

Por fim, os rótulos preditos são comparados aos rótulos binários verdadeiros (`y_test`), e são calculadas a matriz de confusão e métricas de desempenho padrão (como *precision*, *recall* e *F1-score*) por meio das funções `confusion_matrix` e `classification_report` da biblioteca `scikit-learn`. Como o modelo foi configurado para discriminar apenas entre *normal* e *ataque*, todas as variantes de ataque presentes no NSL-KDD são tratadas como uma única classe *non-self*, em conformidade com a proposta de detecção de anomalias do NSA.

Resultados

Após o pré-processamento descrito na Seção anterior, o conjunto de treino do NSL-KDD resultou em 125 973 instâncias com 35 atributos numéricos, enquanto o conjunto de teste permaneceu com 22 544 instâncias e a mesma dimensionalidade.

Na etapa de geração de detectores, foram solicitados 7 500 V-detectores, todos efetivamente obtidos. Os centros dos detectores foram amostrados uniformemente dentro dos limites mínimos e máximos de cada atributo, e os raios foram definidos pela distância L1 até a amostra *self* mais próxima. A estatística dos raios resultantes apresentou média de aproximadamente 13.12, com valores mínimos e máximos de 7.76 e 17.65, respectivamente, indicando que os detectores foram posicionados em regiões afastadas do conjunto *self*, cobrindo diferentes porções do espaço *non-self*.

Desempenho na Detecção de Intrusões

O desempenho do NSA foi avaliado no conjunto de teste do NSL-KDD, considerando um cenário binário em que a classe 0 representa tráfego *normal* e a classe 1 representa qualquer tipo de *ataque*. A Tabela 1 apresenta a matriz de confusão obtida.

Table 1: Matriz de confusão do NSA no conjunto de teste (rótulos verdadeiros nas linhas, predições nas colunas).

	Predito 0 (normal)	Predito 1 (ataque)
Real 0 (normal)	8 951	760
Real 1 (ataque)	4 498	8 335

A partir dessa matriz, foram calculadas as métricas de precisão, revocação e F1-score por classe, bem como a acurácia global, resumidas na Tabela 2.

Table 2: Métricas de classificação do NSA no conjunto de teste.

Classe	Precisão	Revocação	F1-score	Suporte
0 (normal)	0.6656	0.9217	0.7730	9 711
1 (ataque)	0.9164	0.6495	0.7602	12 833
Acurácia		0.7668		
Média macro	0.7910	0.7856	0.7666	22 544
Média ponderada	0.8084	0.7668	0.7657	22 544

Observa-se que o modelo atinge uma acurácia global de aproximadamente 76.7%. A classe *normal* apresenta alta revocação (92.17%), indicando que a maior parte do tráfego legítimo é corretamente identificado, embora com precisão moderada (66.56%) devido à presença de falsos positivos. Já a classe *ataque* apresenta alta precisão (91.64%), ou seja, quando o NSA sinaliza uma intrusão, esta tende a ser de fato um ataque, ao custo de uma revocação inferior (64.95%), o que indica que uma fração dos ataques ainda não é detectada. Esses resultados refletem o compromisso intrínseco do NSA entre cobertura do espaço *non-self* e controle de falsos alarmes em um cenário de detecção de intrusões binária.

Conclusão

Não feito ainda

References

[1] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer," in *Proceedings of IEEE Symposium on Security and Privacy*, pp. 202–212, IEEE, 1994.

[2] S. A. Hofmeyr and S. Forrest, "The immune system as a model for network intrusion detection," *Proceedings of the 1999 IEEE Systems, Man, and Cybernetics Conference*, 1999.

[3] J. Kim and P. J. Bentley, "Negative selection and classification techniques," *Artificial Immune Systems*, pp. 343–356, 2007.

[4] S. Jiang, C. Xu, and Z. Xu, "A deterministic detector generating algorithm in negative selection," *Pattern Recognition*, vol. 40, no. 4, pp. 1201–1232, 2007.

[5] J. Greensmith, U. Aickelin, and S. Cayzer, "Dendritic cells for anomaly detection," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 524–531, 2005.

[6] D. Dasgupta and F. Nino, "Advances in artificial immune systems," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 40–49, 2006.

[7] R. P. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation," *DARPA Information Survivability Conference and Exposition. DISCEX'00*, vol. 2, pp. 12–26, 2000.

[8] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, IEEE, 2009.