# SVM-KMEANS, A SEMI-SUPERVISED LEARNING APPROACH FOR OUTLIER DETECTION

XU SHENGZHE 2012212778 120720444

# Outline



- Tasks that been finished
- Background
- SVM-KMeans Algorithm
  - Pretreatment
  - SVM Part
  - K-Means Part
- Result & Analysis
  - Experience Data
  - Simulation result
  - Performance Analysis
- Conclusion

# Tasks that been finished



- Task1: Study the knowledge of machine learning
- Task2: Study the knowledge of outlier detection
- Task3: Realize an outlier detection algorithm
- Taks4: Analyze the performance of the algorithm and try to improve it

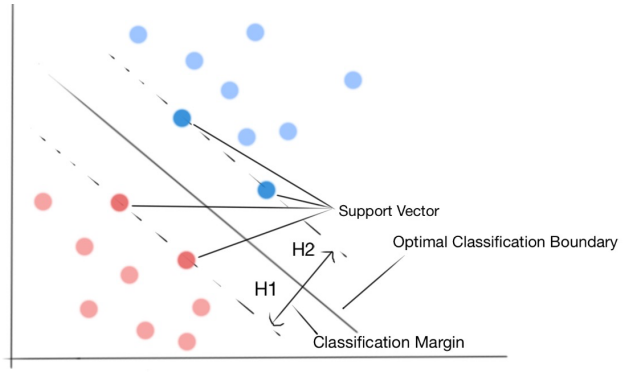
# Background

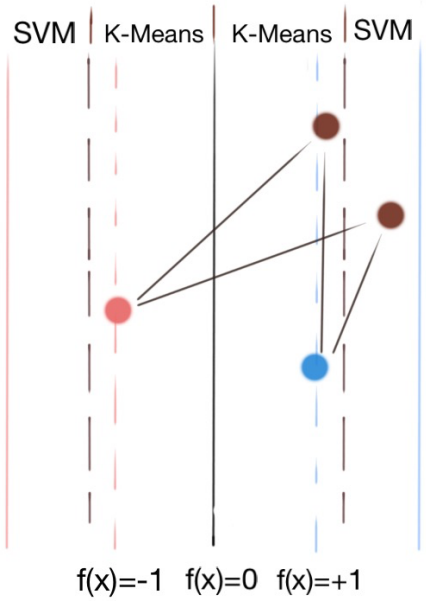


- Outlier Detection : Basic Area of Machine Learning
- Network Intrusion Detection : Important application
  - Accurate + Timely is needed
- Traditional Algorithm
  - Support Vector Machine
    - Special Zone with low accuracy ✖
  - K-Means with threshold judgment
    - Accuracy not stable ✖

# SVM-KMeans Algorithm






- Preprocess data
- Run SVM classify first
  - Get OCB & CM
  - Get Support Vector
- Run K-Means cluster second
  - Use cluster result & threshold judgment to correct Samples near Support Vector
- Chose Figures & Kernel Parameter
- Get Better Result

# Pretreatment



- Normalization
  - Continue Data should be remapped smoothly to an unit interval.
  - Ex. [0, 58329] should be remapped to [0, 1]
- Transform
  - Discrete Data should be rewrite into an numerable form
  - Ex. [TCP, UDP, ICMP] should be rewrite to [1/3, 2/3, 1]
- Format
  - Easily using format rather than origin dataset

# SVM Part - Feature Chosen 1



- Score
  - Intra Class Distance S1
  - Inter Class Distance S2
  - Contribution Value S2 – S1

$$S_1 = \frac{1}{n} \sum_{x_i, x_j \in w} \left\| x_i - x_j \right\|$$

$$S_2 = \frac{1}{n_1} \sum_{x_i \in w_1} f(x_i) - \frac{1}{n_2} \sum_{x_j \in w_2} f(x_j)$$

- Method
  - Each time add one feature and test the Contribution Value
- Target
  - Get useful features & give up waste features
  - Improve accuracy, reduce dimension and time

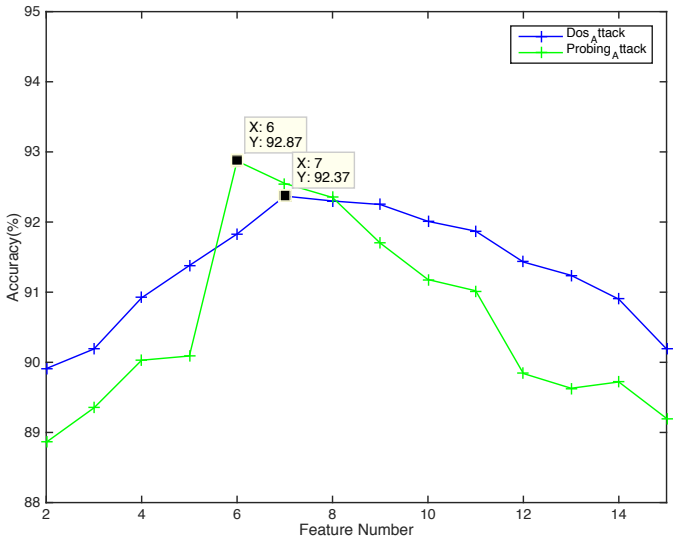
# SVM Part - Feature Chosen 2

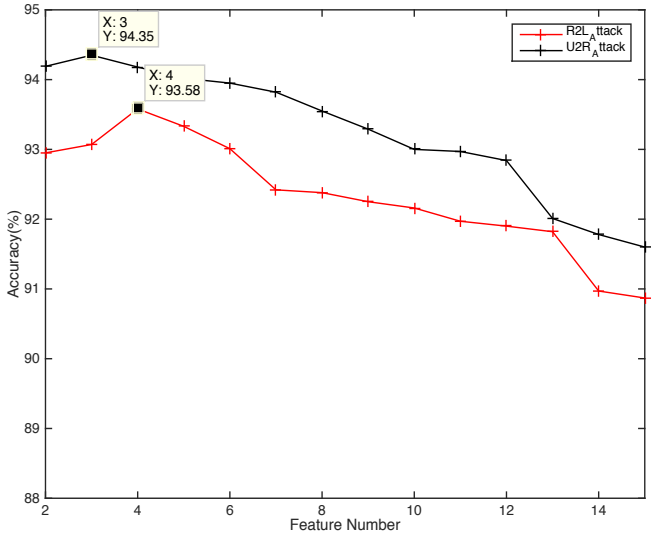


| Top 10 feature of Dos Attack (for example) | | | |
|---|---|---|---|
| Feature | Intra class distance S1 | Inter class distance S2 | Contribution value S2 - S1 |
| 2.protocol_type | 0.032 | 2.985 | 2.953 |
| 7.land | 0.041 | 2.943 | 2.902 |
| 9.urgent | 0.039 | 2.872 | 2.833 |
| 11.num_failed_logins | 0.051 | 2.802 | 2.751 |
| 16.num_root | 0.045 | 2.791 | 2.746 |
| 29.srv_serror_rate | 0.056 | 2.787 | 2.731 |
| 33.dst_host_srv_count | 0.047 | 2.773 | 2.726 |
| 25.serror_rate | 0.062 | 2.761 | 2.699 |
| 34.dst_host_same_srv_rate | 0.059 | 2.724 | 2.665 |
| 20.num_outbound_cmds | 0.068 | 2.730 | 2.662 |

# SVM Part - Feature Chosen 3





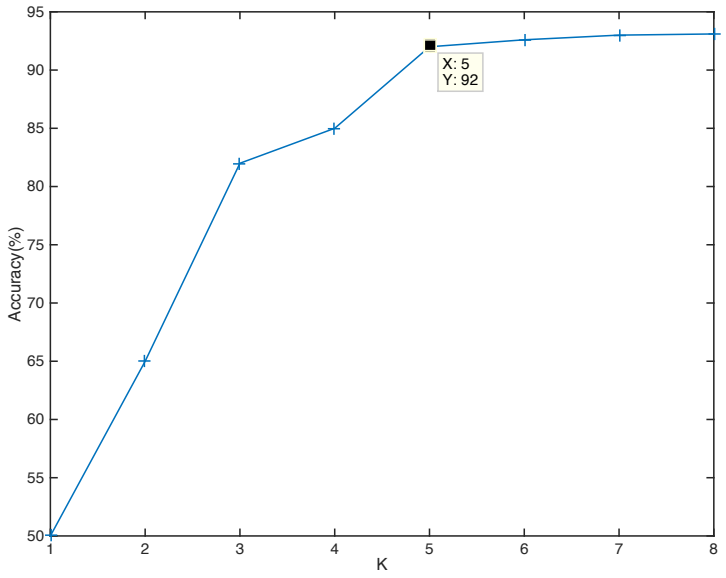




| Chosen in Origen Sequence Dos for example | | Chosen by S2 – S1 Contribution Value | | |
|---|---|---|---|---|
| Feature subset | Accuracy (%) | Attack Type | Feature chosen | Accuracy(%) |
| First 10 subset | 86.1% | Dos | Top 7 | 92.37 |
| First 15 subset | 88.2% | Probing | Top 6 | 92.87 |
| First 25 subset | 83.9% | R2L | Top 4 | 93.58 |
| First 35 subset | 85.7% | U2L | Top 3 | 94.35 |
| All 41 subset | 82.3% | | | |

# K-Means Part



- Give K-Means Cluster on data
- Use 2 threshold judgments to do classify
  - Amount - Outlier usually far less than normal connection
  - Vote - Large ratio outliers in one Cluster leads to high possibility of new outliers
- Re-label Samples
- Enumerate search a proper K value and Check accuracy



# Result - Experience Data



| KDDCUP99 | Total 4940200 connections & 41 attributes for each | | | | |
|---|---|---|---|---|---|
| 5 label type | Normal | Dos | Probing | R2L | U2R |
| Training | 27880 | 8716 | 3451 | 240 | 115 |
| Test | 3000 | 1200 | 850 | 40 | 25 |

Example of one connection

219,TCP,smtp,SF,1684,363,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,
1.00,0.00,0.00,0.00,104,66,0.63,0.03,0.01,0.00,0.00,0.00,0.00,0.00,normal

After Pretreatment it become

+1 1:0.004 2:0.3333 3:0.6 4:0.75 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:1 13:0 14:0
15:0 16:0 17:0 18:0 19:0 20:0 21:0 22:0 23:1 24:1 25:0 26:0 27:0 28:1 29:0 30:0
31:0 32:0.478 33: 0.259 34:0.63 35:0.03 36:0.01 37:0 38:0 39:0 40:0 41:0

# Result - Simulation Result



| Type | Accuracy(%) | | | Average Detection Time(s) | | |
|---|---|---|---|---|---|---|
| | SVM | K-MEANS | SVM-KMeans | SVM | K-MEANS | SVM-KMeans |
| Normal | 92.23 | 90.17 | 95.15 | 0.96 | 0.89 | 0.96 |
| Dos | 88.69 | 87.42 | 93.40 | 1.62 | 1.51 | 1.83 |
| Probing | 90.05 | 89.75 | 92.18 | 1.96 | 1.82 | 2.06 |
| R2L | 87.58 | 90.61 | 93.26 | 0.81 | 0.76 | 0.96 |
| U2R | 91.36 | 89.74 | 94.79 | 1.45 | 1.31 | 1.49 |

Better

# Performance Analysis



| | Whether Supervised | time complexity | Data Amount Demand | Comments |
|---|---|---|---|---|
| K-Means with threshold judgment | No | O(N*K*T) | No | local optimal trap will lead to accuracy reduction |
| Support Vector Machine (SVM) | Yes | O(N^2) | Little | Samples near OCB[1] not very high accuracy |
| SVM-KMeans Method | Yes | O(N^2) + O(N*K*T) ~ O(N^2) | Little | Combine two method to enhance accuracy |

[1] OCB: Optimal Classification Boundary

# Conclusion



- Successfully finished all 4 tasks
- Independently, create a new outlier detection algorithm that enjoy a higher accuracy
- Future work
  - Build Tool kit to easily usage and visualization

# Thanks for listening