Machine learning security

Tom Ristenpart
CS 6431
Detecting attacks

• **Anomaly detection** (Denning 1987)
  – Attacks defined as outlier behavior
  – E.g.: how far away from mean is some monitored behavior?

• **Supervised machine learning**
  – Lots of malicious and benign examples
  – Train an ML model to classify behavior as malicious or not
Spam volume

Billions of messages

http://www.senderbase.org/static/spam/#tab=1
This is an automatic notification of your current disk space usage on the CSE mail server:

csemailbox.ucsd.edu

Your account status:

Current utilization: 95.33%
Space used: 976 MB
Available space: 47 MB
Account limit: 1024 MB

Once your quota has been reached, mail will no longer be delivered to your account, and will be returned to the sender as undeliverable.

If you are not sure where to look for mail that can likely be deleted to clear space in your account, you may likely have large amounts of mail in your Trash and/or Junk folders. Also, you may have a large amount of mail accumulating in your Sent folder over time, if you have configured your mail client to automatically save sent messages.

Your account limit may be increased for an additional charge, as per the CSE Recharge Policy. Please contact CSEHelp regarding quota increases.

Please reply to this message or contact CSEHelp <csehelp@cs.ucsd.edu> if you have any questions or require assistance.

Thank you,
Spam Classifiers

This classifier will be trained from a large corpus of labeled data.

Ham

Spam
Supervised machine learning (ML)

(1) Gather some labeled data
(email<sub>1</sub>, spam), (email<sub>2</sub>, spam)
(email<sub>3</sub>, ham), (email<sub>4</sub>, ham)

(2) Choose features
\[ \text{ext(email)} = x_1, \ldots, x_n \]

(2) Train ML model \( f \) from data
\[ f (x_1, \ldots, x_n) = \text{score} \]
\[ y = \text{threshold(score)} \]

Models typically output score – real-valued measure of “spamminess”
Set a threshold to label binary class
Naïve Bayes Classifier

Represent email as “bag of words”

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Count</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>quota</td>
<td>1</td>
<td>$x_1$</td>
</tr>
<tr>
<td>webmail</td>
<td>4</td>
<td>$x_2$</td>
</tr>
<tr>
<td>wisconsin</td>
<td>0</td>
<td>$x_3$</td>
</tr>
<tr>
<td>viagra</td>
<td>0</td>
<td>$x_4$</td>
</tr>
</tbody>
</table>

Intuition: spam and ham have different distribution of keywords

$$\Pr[ \text{spam} \mid x_1, x_2, \ldots, x_n ] = \frac{\Pr[x_1, x_2, \ldots, x_n \mid \text{spam}] \Pr[\text{spam}]}{\Pr[x_1, x_2, \ldots, x_n]}$$

Bayes’ theorem

$$= \frac{\Pr[\text{spam}] \prod \Pr[x_i \mid \text{spam}]}{\Pr[x_1, x_2, \ldots, x_n]}$$

“Naïve”: assume words independent

$$\Pr[ \text{ham} \mid x_1, x_2, \ldots, x_n ] = \frac{\Pr[\text{ham}] \prod \Pr[x_i \mid \text{ham}]}{\Pr[x_1, x_2, \ldots, x_n]}$$
Naïve Bayes Classifier

Represent email as “bag of words”

\[
\begin{align*}
\text{quota} & : 1 & x_1 \\
\text{webmail} & : 4 & x_2 \\
\text{wiscosin} & : 0 & x_3 \\
\text{viagra} & : 0 & x_4 \\
\vdots & & \vdots
\end{align*}
\]

Intuition: spam and ham have different distribution of keywords

\[
\begin{align*}
\Pr[\text{spam} | x_1, x_2, \ldots, x_n] &= \frac{\Pr[\text{spam}] \prod \Pr[x_i | \text{spam}]}{\Pr[x_1, x_2, \ldots, x_n]} \\
\Pr[\text{ham} | x_1, x_2, \ldots, x_n] &= \frac{\Pr[\text{ham}] \prod \Pr[x_i | \text{ham}]}{\Pr[x_1, x_2, \ldots, x_n]}
\end{align*}
\]

Classify as spam if:

\[
\Pr[\text{spam} | x_1, x_2, \ldots, x_n] > \Pr[\text{ham} | x_1, x_2, \ldots, x_n]
\]
Naïve Bayes Classifier

Represent email as “bag of words”

<table>
<thead>
<tr>
<th>Word</th>
<th>Count</th>
<th>x_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>quota</td>
<td>1</td>
<td>x_1</td>
</tr>
<tr>
<td>webmail</td>
<td>4</td>
<td>x_2</td>
</tr>
<tr>
<td>wisconsin</td>
<td>0</td>
<td>x_3</td>
</tr>
<tr>
<td>viagra</td>
<td>0</td>
<td>x_4</td>
</tr>
</tbody>
</table>

Intuition: spam and ham have different distribution of keywords

Classify as spam if:

\[
\frac{\Pr[\text{spam}] \prod \Pr[x_i | \text{spam}]}{\Pr[x_1, x_2, \ldots, x_n]} > \frac{\Pr[\text{ham}] \prod \Pr[x_i | \text{ham}]}{\Pr[x_1, x_2, \ldots, x_n]}
\]
Naïve Bayes Classifier

Represent email as “bag of words”

<table>
<thead>
<tr>
<th>Word</th>
<th>Value</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>quota</td>
<td>1</td>
<td>(x_1)</td>
</tr>
<tr>
<td>webmail</td>
<td>4</td>
<td>(x_2)</td>
</tr>
<tr>
<td>wisconsin</td>
<td>0</td>
<td>(x_3)</td>
</tr>
<tr>
<td>viagra</td>
<td>0</td>
<td>(x_4)</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
</tbody>
</table>

Intuition: spam and ham have different distribution of keywords

Classify as spam if:

\[
Pr[\text{spam}] \prod p_i^{x_i} > Pr[\text{ham}] \prod q_i^{x_i}
\]

Is multinomial Naïve Bayes a linear classifier?

\[
\log( Pr[\text{spam}] \prod p_i^{x_i} ) = \log( Pr[\text{spam}] ) + \sum x_i \log p_i = w^T x + b \quad \text{for } w_i = \log p_i
\]
Spam classifiers

MNB classifier

Spam / Ham

Extracts bag-of-words representation
Applies linear function f to it and returns if output positive

[Sahami et al. 1998]

<table>
<thead>
<tr>
<th>Feature Regime</th>
<th>Junk</th>
<th>Legitimate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>Words only</td>
<td>97.1%</td>
<td>94.3%</td>
</tr>
<tr>
<td>Words + Phrases</td>
<td>97.6%</td>
<td>94.3%</td>
</tr>
<tr>
<td>Words + Phrases + Domain-Specific</td>
<td>100.0%</td>
<td>98.3%</td>
</tr>
</tbody>
</table>

Table 1: Classification results using various feature sets.
Spam classifiers

• Real classifiers more complex than this
  – Other features: Who is sender? How many links embedded? Is it from an open mail relay?
  – Can update in real-time given labelings by user
  – For larger orgs, can leverage wide view across many email recipients

• Nowadays some companies do pretty good job of making sure spam doesn’t hit your inbox
  – 95% of email gets filtered as spam (2009, ENISA Spam Survey)
Spam classifiers

MNB classifier

Extracts bag-of-words representation
Applies linear function $f$ to it and returns if output positive

How would you build classifier-bypassing spam?
Threat models in ML use

- **Poisoning attacks:**
  - Insert bogus examples into training data

- **Evasion attacks:**
  - Find email that is spam but marked as ham

Key question (for both settings):
- What does adversary know?
- Membership queries are assumed possible

- **Learning the features:**
  - Use black-box access to classifier to determine what are features

- **Model extraction:**
  - Use black-box access to classifier to learn $f$
Lowd, Meek 2005

• Investigate evasion attacks in spam setting
  – ACRE = adversarial classifier reverse engineering
  – This refers to model evasion in membership query setting

• First paper that looks at model extraction as a step towards evasion
  – Give algorithm for continuous features and linear classifiers
  – Show ACRE
LM model extraction attack

Assume one knows features, or direction access to feature space membership queries

\[ f(\text{ext}(\text{email})) \]
LM model extraction attack

Assume one knows features, or direction access to feature space membership queries

Assume model $f$ is a linear function and it returns the sign of its output

Find points arbitrarily close to line, solve for line

They describe it differently, but same idea

Point out that on boolean inputs, extraction is NP-hard. Show ACRE algorithm that works anyway
Threat models in ML use

- **Poisoning attacks:**
  - Insert bogus examples into training data

- **Evasion attacks:**
  - Find email that is spam but marked as ham

Key question (for both settings):
- What does adversary know?
- Membership queries are assumed possible

- **Learning the features:**
  - Use black-box access to classifier to determine what are features

- **Model extraction:**
  - Use black-box access to classifier to learn $f$
srndic, laskov 2014

- investigate evasion attacks “in the wild”
- target: pdfrate by [smutz, stavrou 2012]
  - regexes to extract hundreds of features from pdfs
  - random forest (1000 decision trees)
PDFrate

A machine learning based classifier operating on document metadata and structure

Submit File

Upload File:  Choose File  No file chosen  Submit

Search File

File Hash:  Search

If you'd like to use SSL, use https://www.csmutz.com/pdfrate/
Srndic, Laskov 2014

- Investigate evasion attacks “in the wild”
- Target: PDFrate by [Smutz, Stavrou 2012]
  - Regexes to extract hundreds of features from PDFs
  - Random forest (1000 decision trees)

<table>
<thead>
<tr>
<th>Votes</th>
<th>FP Rate</th>
<th>TP Rate</th>
<th>FP Count</th>
<th>TP Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8</td>
<td>0.00021</td>
<td>0.9327</td>
<td>21</td>
<td>277</td>
</tr>
<tr>
<td>0.7</td>
<td>0.00057</td>
<td>0.9461</td>
<td>57</td>
<td>281</td>
</tr>
<tr>
<td>0.6</td>
<td>0.00132</td>
<td>0.9529</td>
<td>132</td>
<td>283</td>
</tr>
<tr>
<td>0.5</td>
<td>0.00244</td>
<td>1.0000</td>
<td>243</td>
<td>297</td>
</tr>
</tbody>
</table>
Srndic, Laskov 2014

• Investigate evasion attacks “in the wild”
• Target: PDFrate by [Smutz, Stavrou 2012]
  – Regexes to extract hundreds of features from PDFs
  – Random forest (1000 decision trees)
• The question: How to do evasion?
  – Modify a malicious PDF to be labeled as benign
Adversarial knowledge

Always assume know subset of features: Glean from description in papers, simple testing
The idea of surrogate models

- Train locally a *surrogate model* using either:
  - Training DB used by target
  - Separately collected training data
    - Must be “close” to training DB

- Use surrogate to mount evasion attack:
  - Mimicry attack
    - Change features to look like benign sample.
    - See if surrogate model misclassifies as benign
  - Gradient-descent based attack
    - Optimization problem: find smallest change needed to force surrogate to misclassify
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Classifier</th>
<th>Dataset</th>
<th>Attack(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>SVM</td>
<td>Surrogate</td>
<td>Mimicry, GD-KDE</td>
</tr>
<tr>
<td>FC</td>
<td>Random Forest</td>
<td>Surrogate</td>
<td>Mimicry</td>
</tr>
<tr>
<td>FT</td>
<td>SVM</td>
<td>Contagio</td>
<td>Mimicry, GD-KDE</td>
</tr>
<tr>
<td>FTC</td>
<td>Random Forest</td>
<td>Contagio</td>
<td>Mimicry</td>
</tr>
</tbody>
</table>
Discussion

• ML classifiers often an instance of security through obscurity
  – Features seem hard to hide
  – Models can sometimes be learned (Lowd/Meek) or approximated (surrogates)
  – Evasion ends up easy

• May still be useful to catch dumb attackers

• Can we have principled ability to, e.g., prevent learning features / extraction / evasion?