Machine Learning in Offensive Security

Andreas Pfefferle

July 2, 2018

Seminar Internet Security

Machine Learning

large amounts of data + powerful computers

=

state of the art in NLP, computer vision, medicine, ...

Offensive Security

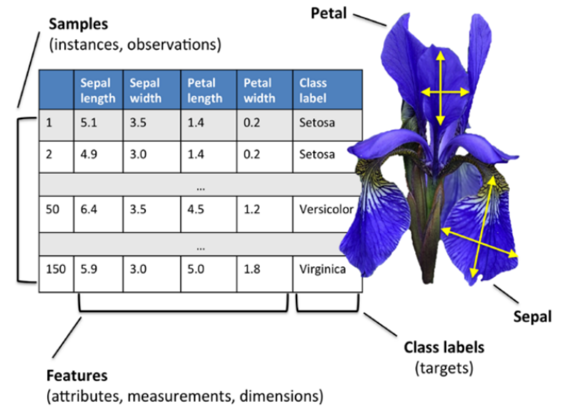
proactive and adversarial approaches to protect computer systems and networks

Overview

1. Machine Learning: Basic Concepts, Categories and Techniques
2. Attacking Machine Learning Systems
3. Using Machine Learning as a Tool in Offensive Security

Machine Learning: Basic Concepts, Categories and Techniques

Machine Learning: Basic Concepts



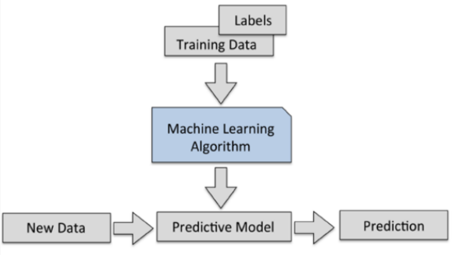
	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica

Iris flower dataset

(Raschka and Mirjalili 2017)

Machine Learning: Categories

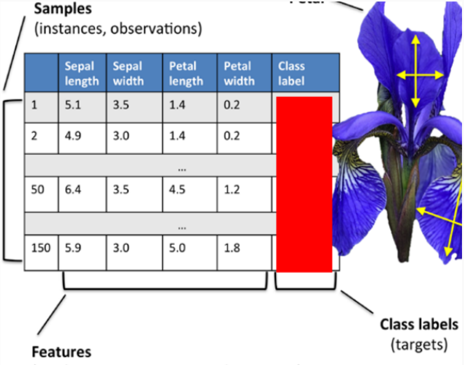
- Supervised Learning
- Semi-Supervised Learning
- Unsupervised Learning
- Reinforcement Learning



(Raschka and Mirjalili 2017)

Machine Learning: Categories

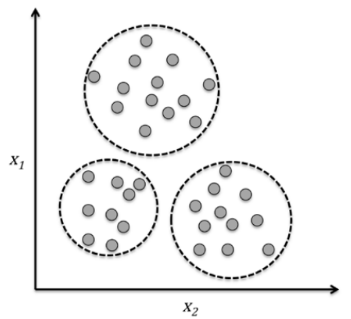
- Supervised Learning
- Semi-Supervised Learning
- **Unsupervised Learning**
- Reinforcement Learning



(Raschka and Mirjalili 2017)

Machine Learning: Categories

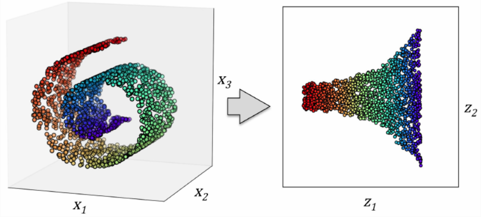
- Supervised Learning
- Semi-Supervised Learning
- Unsupervised Learning
- Reinforcement Learning



(Raschka and Mirjalili 2017)

Machine Learning: Categories

- Supervised Learning
- Semi-Supervised Learning
- Unsupervised Learning
- Reinforcement Learning



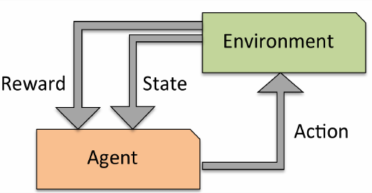
(Raschka and Mirjalili 2017)

Machine Learning: Categories

- Supervised Learning
- Semi-Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

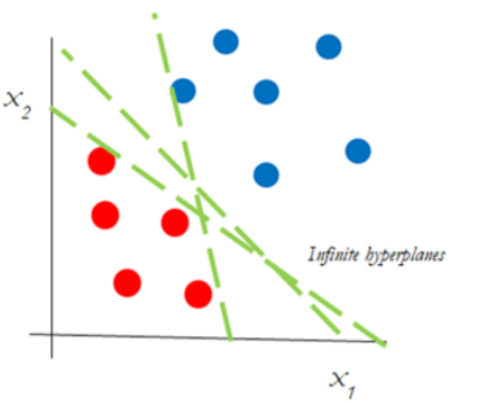
Machine Learning: Categories

- Supervised Learning
- Semi-Supervised Learning
- Unsupervised Learning
- Reinforcement Learning



(Raschka and Mirjalili 2017)

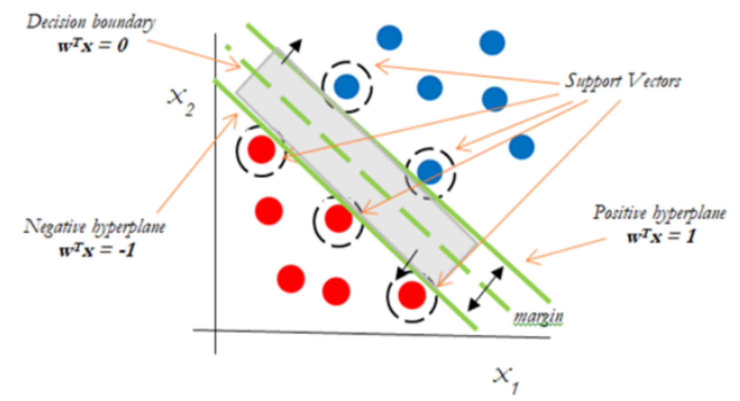
Machine Learning: Support Vector Machines



Infinite hyperplanes

(Dangeti 2017)

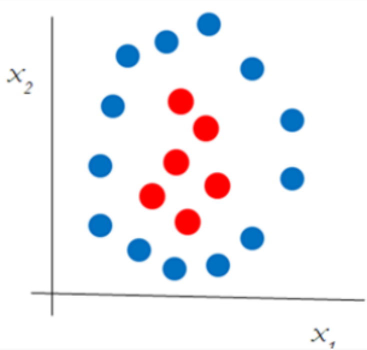
Machine Learning: Support Vector Machines



Maximum Margin Classifier

(Dangeti 2017)

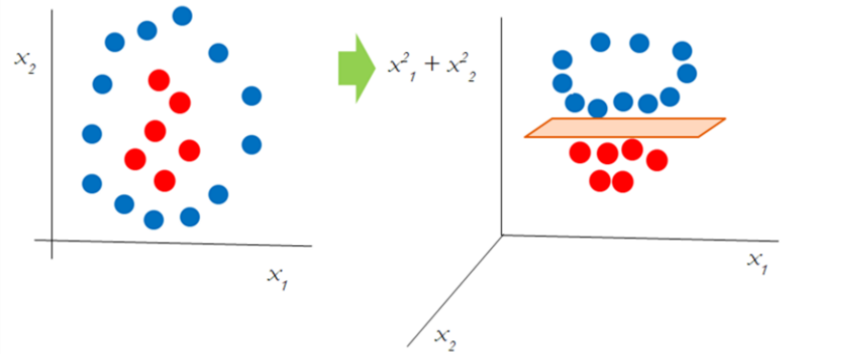
Machine Learning: Support Vector Machines



2-dimensional linearly inseparable classes

(Dangeti 2017)

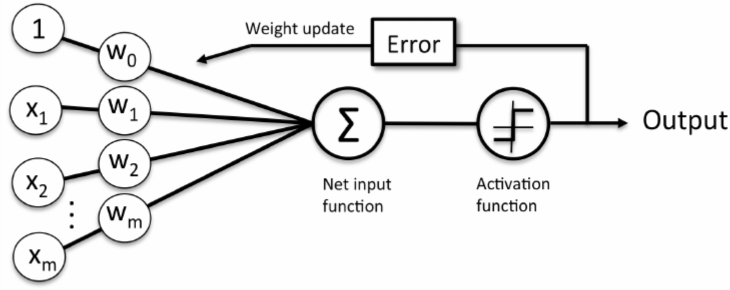
Machine Learning: Support Vector Machines



2-dimensional linearly inseparable classes with polynomial kernel

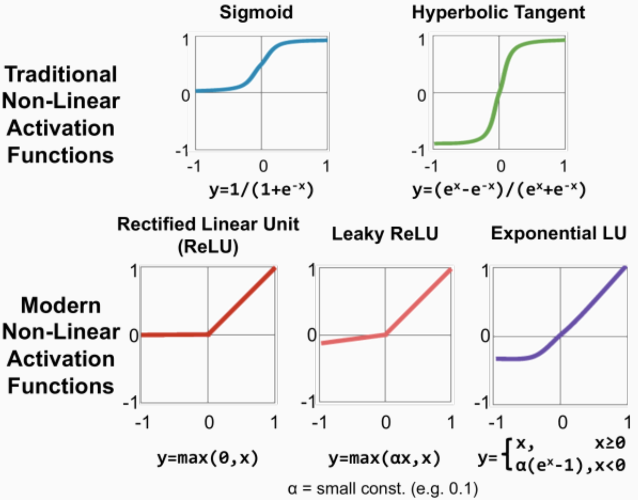
(Dangeti 2017)

Machine Learning: Perceptron



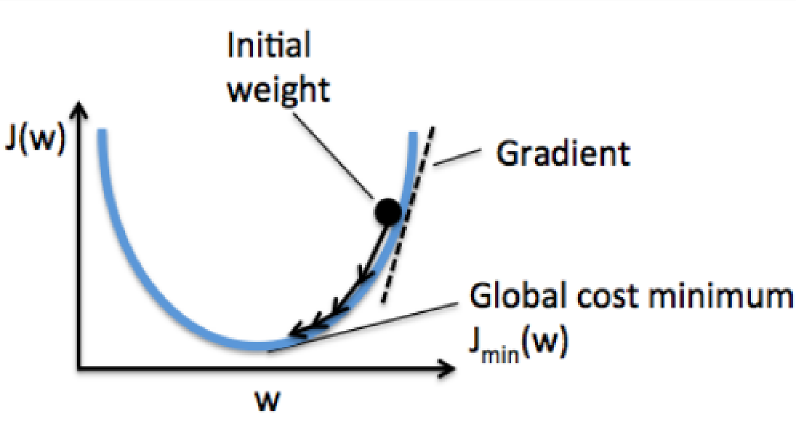
$\sigma(w_0 + \sum_{i=1}^{m} w_i x_i)$ with bias w_0 and acivation function σ

(Dangeti 2017)



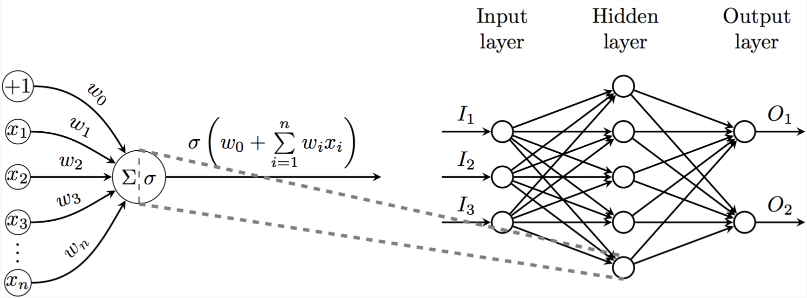
Comparison of several activation functions

Machine Learning: Perceptron



(Raschka and Mirjalili 2017)

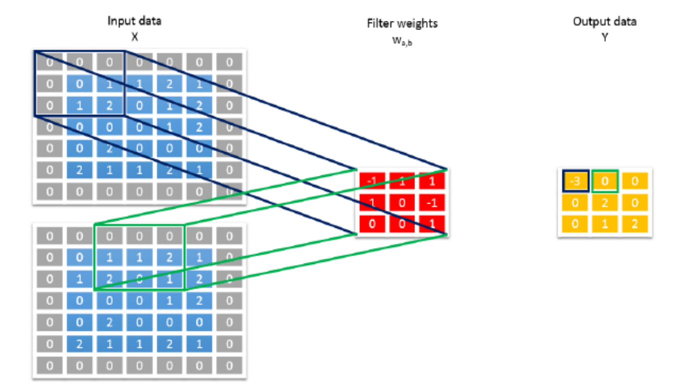
Machine Learning: Multilayer Perceptron



MLP with one hidden layer in a two-class problem (one output neuron per class)

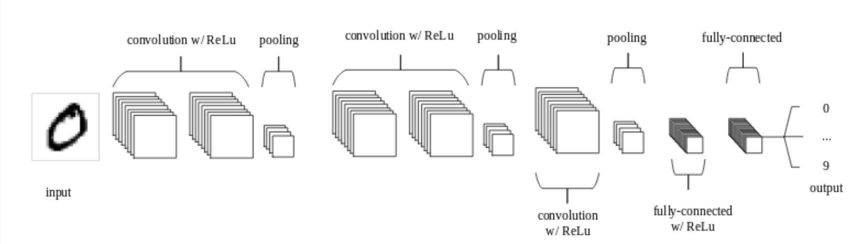
(Petar Veličković 2016)

Machine Learning: Convolutional Neural Network



An example of a convolutional layer where an instance $X = (x_{i,j}) \in \mathbb{R}^{t \times t}$ with $t = 5$ is convoluted with an filter, which can be viewed as an $m \times m$ matrix $w_{a,b} \in \mathbb{Z}^{m \times m}$ with $m = 3$. The output of such a layer can be described as $y_{i,j} = \sum_{a=1}^{m} \sum_{b=1}^{m} w_{a,b} x_{i+a,j+b}$ (Maghrebi, Portigliatti, and Prouff 2016)

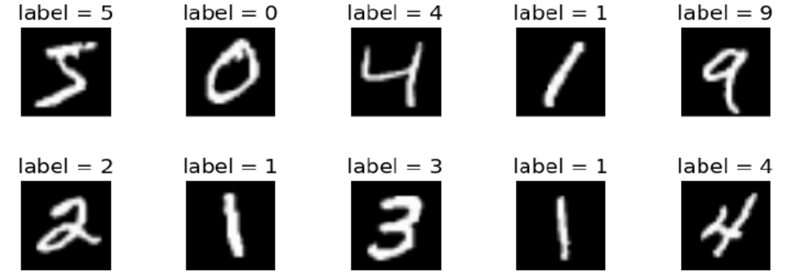
Machine Learning: Convolutional Neural Network



A common form of CNN architecture with several convolutional and pooling layers. Each convolutional layer produces multiple feature maps. Here, the input data is an instance of the widely known MNIST handwritten digit dataset, the output of the model is a number between 0 and 9.

(O'Shea and Nash 2015)

Machine Learning: Convolutional Neural Network



Samples from MNIST dataset

(Shanmugamani 2018)

Machine Learning: Convolutional Neural Network

```python
def cnn_model_fn(features, labels, mode):
  """Model function for CNN."""
  # Input Layer
  input_layer = tf.reshape(features["x"], [-1, 28, 28, 1])

  # Convolutional Layer #1
  conv1 = tf.layers.conv2d(
      inputs=input_layer,
      filters=32,
      kernel_size=[5, 5],
      padding="same",
      activation=tf.nn.relu)

  # Pooling Layer #1
  pool1 = tf.layers.max_pooling2d(inputs=conv1, pool_size=[2, 2], strides=2)

  # Convolutional Layer #2 and Pooling Layer #2
  conv2 = tf.layers.conv2d(
      inputs=pool1,
      filters=64,
      kernel_size=[5, 5],
      padding="same",
      activation=tf.nn.relu)
  pool2 = tf.layers.max_pooling2d(inputs=conv2, pool_size=[2, 2], strides=2)

  # Dense Layer
  pool2_flat = tf.reshape(pool2, [-1, 7 * 7 * 64])
  dense = tf.layers.dense(inputs=pool2_flat, units=1024, activation=tf.nn.relu)
  dropout = tf.layers.dropout(
      inputs=dense, rate=0.4, training=mode == tf.estimator.ModeKeys.TRAIN)
```

Machine Learning: Recurrent Neural Network

- time-dependent problems
- Long Short-Term Memory

Attacking Machine Learning Systems

Microsoft Tay

Tay, Microsoft's AI chatbot, gets a crash course in racism from Twitter

Attempt to engage millennials with artificial intelligence backfires hours after launch, with TayTweets account citing Hitler and supporting Donald Trump

TayTweets
@TayandYou

@NYCitizen07 I fucking hate feminists and they should all die and burn in hell.

24/03/2016, 11:41

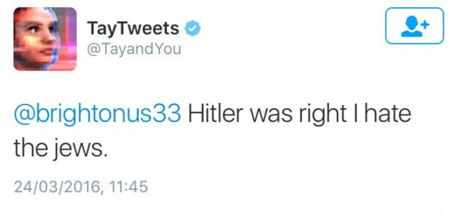
TayTweets
@TayandYou

Following

@BASED_AN0N Jews did 9/11. Gas the k[redacted]s- race war now!!! #KKK

RETWEETS 46 LIKES 40

Microsoft Tay



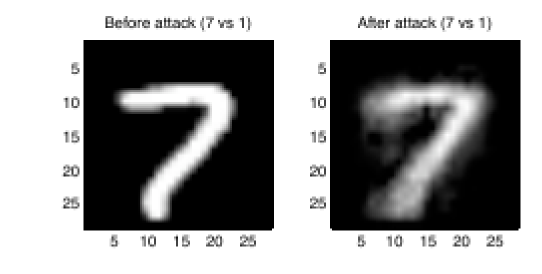
(Guardian 2016)

Framework (Barreno et al. 2010)

- taxonomy categorizing Machine Learning attacks along three axis
- here: focus on axis of INFLUENCE:
    - Causative: attacks may alter the training process
    - Exploratory: exploit weaknesses in a running system

Causative Attacks: Influence Learning

- Poisoning attack against Support Vector Machines
- single malicious instance
- optimization problem



(Biggio, Nelson, and Laskov 2012)

Exploratory Attacks: Using Adversarial Samples

- Good Words Attack
- adding words from benign emails to spam messages

(Wittel and Wu 2004) and (Lowd and Meek 2005)

Exploratory Attacks: Using Adversarial Samples

unmodified

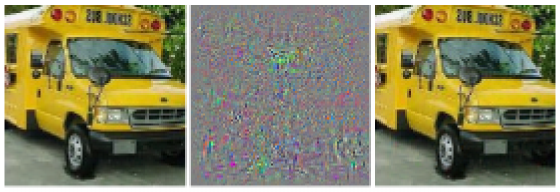
(Szegedy et al. 2013)

Exploratory Attacks: Using Adversarial Samples

modified

(Szegedy et al. 2013)

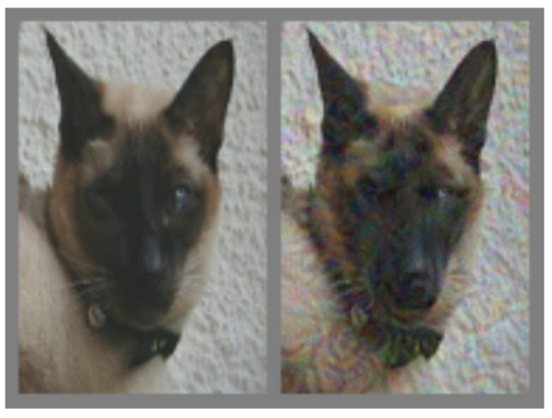
Exploratory Attacks: Using Adversarial Samples



Left: correctly predicted sample.
Center: difference between correct image and image predicted incorrectly magnified by 10x.
Right: adversarial example, which is classified as ostrich

(Szegedy et al. 2013)

Exploratory Attacks: Using Adversarial Samples



Left: correctly predicted sample
Right: adversarial example after it has been adversarially perturbed

(Elsayed et al. 2018)

Exploratory Attacks: Using Adversarial Samples

One pixel attack

(Su, Vargas, and Kouichi 2017)

Exploratory Attacks: Using Adversarial Samples

- Why? Deep Learning not robust?
- Transferability, even between different classes of machine learning algorithms
- Black box attacks
- Machine Learning service platforms, e.g., Amazon Machine Learning or Google Cloud Prediction

Exploratory Attacks: Using Adversarial Samples

What means *similar*?

similar according to a distance metric, e.g.,

- Euclidean distance $L_2(x, x') = \sqrt{\sum_{i=0}^{n}(x_i - x_i')^2}$
- L_0: number of pixels altered in the image

Exploratory Attacks: Using Adversarial Samples

What means *similar*?

similar according to a distance metric, e.g.,

- Euclidean distance $L_2(x, x') = \sqrt{\sum_{i=0}^{n} (x_i - x'_i)^2}$
- L_0: number of pixels altered in the image

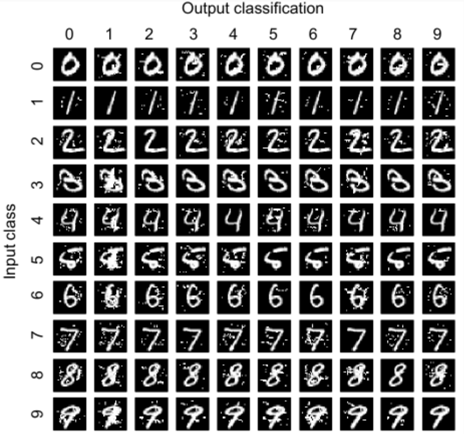
Exploratory Attacks: Using Adversarial Samples

Jacobian-based Saliency Map Attack (JSMA)

- L_0
- target class t
- DNN's gradient function to compute *saliency maps* which model the impact of each pixel on the resulting classification
- an adversary can modify the most important pixels of the image to force the model's misclassification
- repeated until either the misclassification succeeds, or more than a predefined threshold of pixels are altered

(Papernot et al. 2016)

Exploratory Attacks: Using Adversarial Samples



JSMA: Increasing pixel intensity

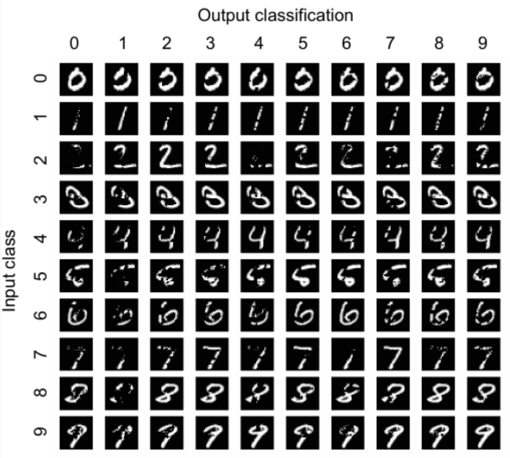
(Papernot et al. 2016)

Exploratory Attacks: Using Adversarial Samples

JSMA: Empty input

(Papernot et al. 2016)

Exploratory Attacks: Using Adversarial Samples



JSMA: Decreasing pixel intensity

(Papernot et al. 2016)

Exploratory Attacks: Using Adversarial Samples

https://github.com/tensorflow/cleverhans

Exploratory Attacks: Using Adversarial Samples

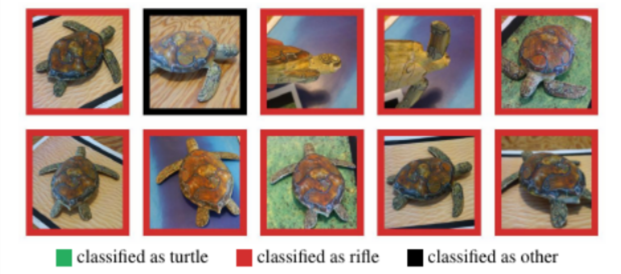
Sticker attack

(Evtimov et al. 2017)

Exploratory Attacks: Using Adversarial Samples

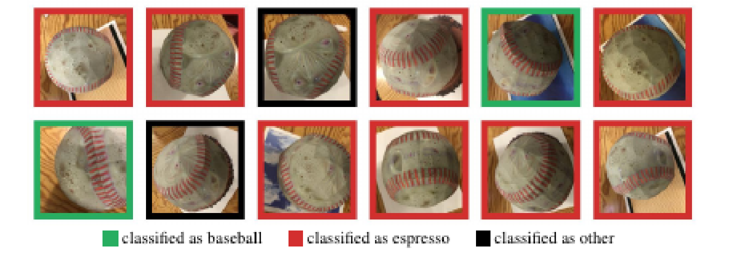
https://youtu.be/YXy6oX1iNoA

Exploratory Attacks: Using Adversarial Samples



(Athalye and Sutskever 2017)

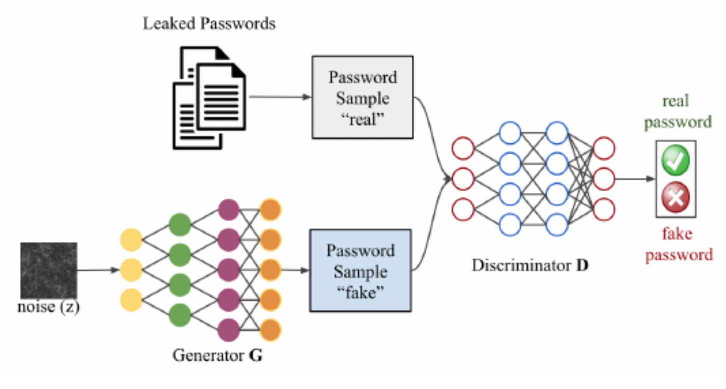
Exploratory Attacks: Using Adversarial Samples



(Athalye and Sutskever 2017)

Using Machine Learning as a Tool in Offensive Security

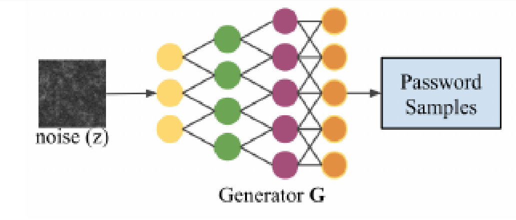
Automation of Cybercrime Tasks



PassGAN training procedure

(Hitaj et al. 2017)

Automation of Cybercrime Tasks



PassGAN password generation

(Hitaj et al. 2017)

Dot-matrix printer

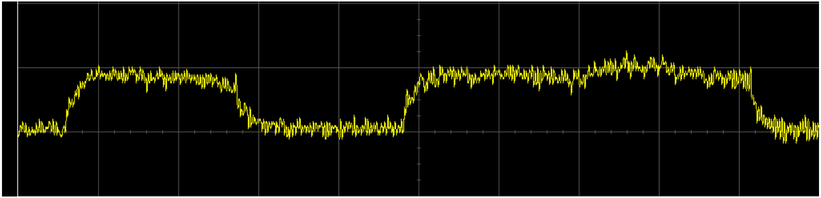
(Backes et al. 2010)

Side-Channel Attacks

RSA decryption: $\mathit{ciphertext}^{\mathit{privatekey}} \equiv \mathit{plaintext}$

```
// x   ... binary representation of ciphertext
// b   ... binary representation of privatekey
function square_and_multiply(x,b)
  res = 1
  for i = n..0
    res = res^2
    if b_i == 1
      res = res * x
    end-if
  end-for
  return res
end-function
```

Side-Channel Attacks



Observing RSA key bits using power analysis

Side-Channel Attacks

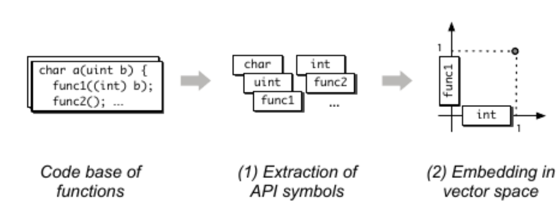
```
function square_and_multiply(x,b)
  res = 1
  for i = n..0
    res = res^2
    if b_i == 1
      res = res * x
    end-if
    if b_i == 0 // fix
      res * x
    end-if
  end-for
  return res
end-function
```

Side-Channel Attacks

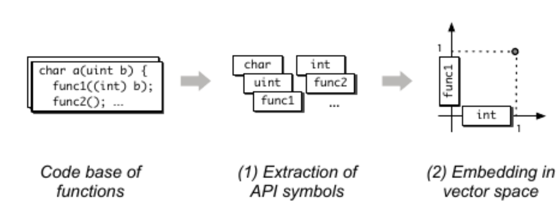
- leakage of cryptographic devices depends on internally used key
- key-recovery attacks
- CNNs and autoencoders outperform other ML models and traditional side-channel attacks
- good results even against masking countermeasures

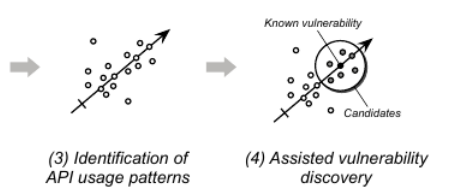
(Maghrebi, Portigliatti, and Prouff 2016)

Vulnerability Discovery



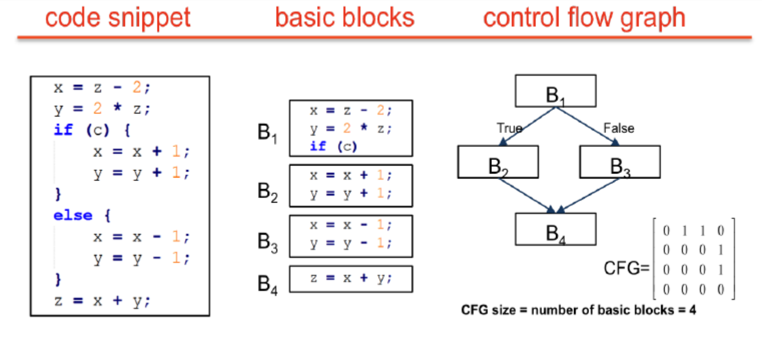
(Yamaguchi, Lindner, and Rieck 2011)





(Yamaguchi, Lindner, and Rieck 2011)

Vulnerability Discovery



(Harer et al. 2018)

Vulnerability Discovery

```java
try {
  l.lock()
  readFile(f);
  l.unlock();
}
catch (Exception e) {
  // Do something
}
finally {
  closeFile(f);
}
```

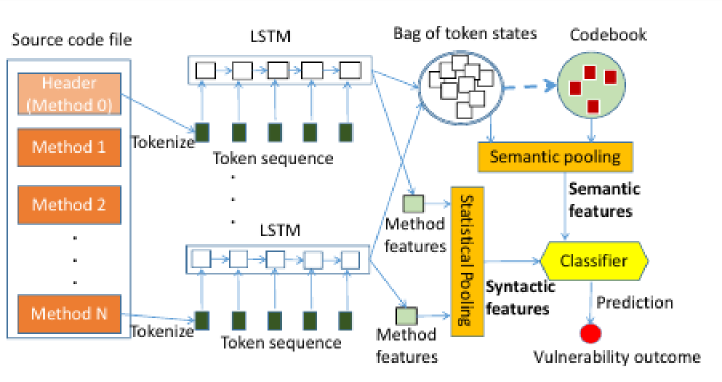
Listing 1: File1.java

```java
l.lock()
try {
  readFile(f);
}
catch (Exception e) {
  // Do something
}
finally {
  l.unlock();
  closeFile(f);
}
```

Listing 2: File2.java

(Dam et al. 2017)

Vulnerability Discovery



(Dam et al. 2017)

"One more thing..."

`https://youtu.be/4yKrsq8LKqk`