ISM: Secure ML

Final Presentation

Abhinav & Kuber

April 23, 2021

Topics covered:

- Trash Bucket Problem
- One Shot Learning for Verification Problem
- Subpopulation Data Poisoning Attack

Github Link to ISM Repo: https://github.com/kubershahi/ashoka-secureml

Trash Bucket

- Problem statement:
  - How does a neural network process unrecognizable data?
  - In other words, when random inputs are fed into a trained neural network, what are the subsequent outputs? Are they random?
  - Can we find any distribution in these output values?
  - Are some classifications more likely than others?
  - Can we somehow change the way we train the neural network, such that a particular output is more likely than others?

Trash Bucket: Experimenting with MNIST Dataset.

- Trained the neural network on the MNIST dataset with 60,000 examples.
- Tested the accuracy with 10,000 examples and produced 97% accuracy.
- Generated 200,000 random data for input that resembled the structure of MNIST dataset. Then fed these data into the trained neural network.
- The outputs :

```
1 -> 0
2 -> 0
3 -> 199565
4 -> 0
5 -> 396
6 -> 0
7 -> 0
8 -> 39
9 -> 0
0 -> 0
```

*Output was 3 almost every time.

- But may be the sequence of training data had something to do with the result.
- So, run the same experiment 100 times but we shuffle the training data every time.
- Results: {0: 2, 1: 0, 2: 3, **3: 10**, 4: 0, **5: 48**, 6: 1, 7: 0, **8: 36**, 9: 0}

  Here, **5 and 8** are the majority and then comes **3**.
- Now, we try some experiments to see if input training data can be sequenced in a particular way such that the output distribution changes.
- Now, we the experiment 20 times but trained 5s and 8s first. (each time is average of 100 experiment)
- Results: {0: 2, 1: 0, 2: 2, **3: 14**, 4: 0, 5: 0, 6: 1, 7: 0, 8: 0, 9: 1}
- 5 and 8 have completely disappeared and 3 is majority now. Recognize that 3 was 3rd highest occurring output before.

- Similarly, we trained 5s and 8s at last.
- Results: {0: 0, 1: 0, 2: 0, 3: 0, 4: 0, **5: 19**, 6: 0, 7: 0, 8: 1, 9: 0}
- 5 is the highest occurring output!
- Idea! Maybe the numbers that are being added near the end of the training process has more value.
- Recognize that although 8 was the second highest occurring number, it is almost null here.
- Idea! 5 is more likely than 8!

- Let’s just train 5 at the end.
- Results: {0: 0, 1: 0, 2: 0, 3: 0, 4: 0, **5: 20**, 6: 0, 7: 0, 8: 0, 9: 0}
- 5 was classified 100% of the time.
- Side experiment: 1 was one of the least occurring outputs. What happens if it is trained at the end.
- Results: {0: 0, **1: 20**, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0}
- 1 is the majority but it has few other spikes as well.

- **Conclusion:**
    - 5,8 and 3 seems to have contributed to majority of the classifications.
    - However, if we train a particular number near the end, it seems that number will be recognized most frequently.

Trash Bucket: Experimenting in Binary Classification (dog vs cat)

- Trained a CNN with 20000 training examples and 5000 validation examples. Dataset
- Accuracy of model = ~75%
- Tested the trained model on 500 cats and dog images: achieved accuracy of ~74%
- Tested the trained model on random 5000 wild images (didn't have cats and dog images). Output: 60% of images misclassified as cats, 40% of the images misclassified as dogs.
- Difficult to infer which one is the trash bucket in binary case as there is only two classes.
- Testing on the random RGB values generated images might tell something about the trash bucket
- Shuffling the order of classes in dataset while training might also reveal something new about the trash bucket.

- Tested the CNN on 5000 randomly generated RGB images. 97% of them misclassified as cats. Cats seemed to be the trash bucket.
- Trained the network with cats first and then dogs. Training was biased towards the dogs as at the end it was trained with dogs, hence all the 5000 randomly generated RGB images were misclassified as dogs.
- Similarly, trained the network with dogs first and then cats. Training was biased towards the cats as at the end it was trained with cats, hence all the 5000 randomly generated RGB images were misclassified as cats.
- Shuffling the dataset as to learn on one class and then on another should be avoided as there are only two classes and the training gets biased.
- **Conclusion:** For binary cases, it seems it hard to infer any valid conclusion regarding the trash bucket.

One Shot Learning

- Consider the scenario at an airport where you have to verify whether the person standing in front of you is the same person whose picture is in the database.
- You only have only two images, and based on that you have to verify whether the person in the image is same or not?
- One way to solve this problem would be to train a deep learning model like CNN (Convolutional Neural Network) with enough images of a person and then use it to make predictions of any new images of the same person.
- But this only works for the people which CNN model has already seen while training, it won't be able to verify a new person if the person wasn’t in the training dataset. This significantly limits the usability of such models in verification task.

- Another way is to train a model in a way that it learns the differences and similarities between two images. This way the model would be able to generalize on the new images as it just has two see if the two new images are similar or not.
- This is called one shot learning: in a single shot you need to say whether the images are similar or not. The network architecture that is able to implement this is the siamese neural network.

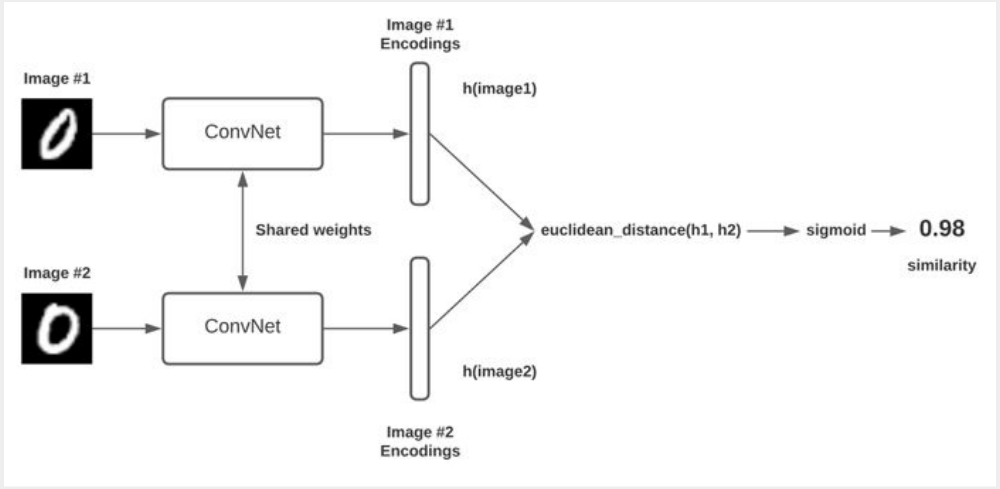


Fig 1.1: Siamese Network (source: PyImage Search)

- A siamese network has two CNNs that processes two images and provides the final encoding of these images and then, you simply calculate the difference between to see how similar they are.
- So, while training you pass similar and dissimilar images as examples so that the model learns to generalize well on both similar and different images.
- For more details on Siamese Network, see the paper we followed.

One Shot Learning Implementation

- Implemented one shot learning on the MNIST and Omniglot Dataset.
- Notebooks: one_shot_learning_omniglot.ipynb & One_Shot_Learning_MNIST.ipynb

Subpopulation Attack

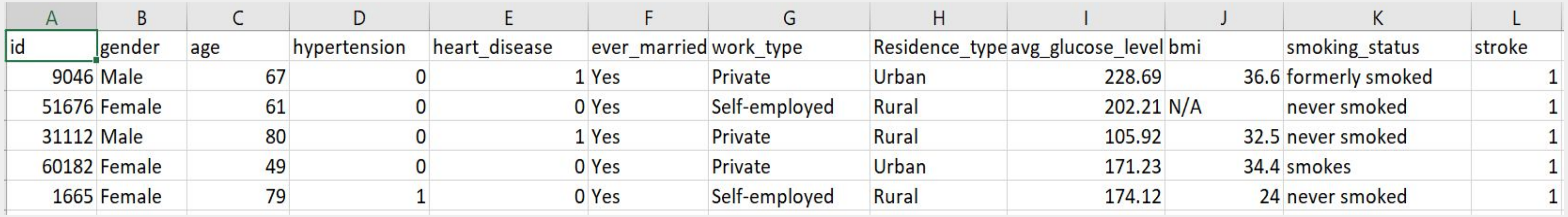
- Problem statement.
  - Given a large and diverse dataset, our aim to reduce the performance of a classifier in a subpopulation while still retaining the overall performance for the rest of the dataset.
  - The scope of the attack is a small subpopulation compared to the complete dataset. This makes our attack stealthy.
- Structure of the attack.
  - This attack is an interpolation between a targeted attack (misclassifying a single point) and an availability attack (misclassifying as many points as possible).
  - The adversary’s goal is twofold—impact the predictions on inputs coming from a subpopulation in the data, but do not impact the performance of the model on points outside this subpopulation.
- For more information on the attack, see paper summary.

Subpopulation Attack Implementation on Adult Dataset (Paper)

- Went through the implementation code mentioned in the paper, understood and presented it.
- Modified certain parts of the code to see the feature and cluster subclasses with the highest Target metric.
- Implementation Demonstration
- Notebooks: subpopulation_attack_paperimplementation.ipynb

Subpopulation Attack Implementation on Heart Stroke Dataset

- Stroke Prediction Dataset

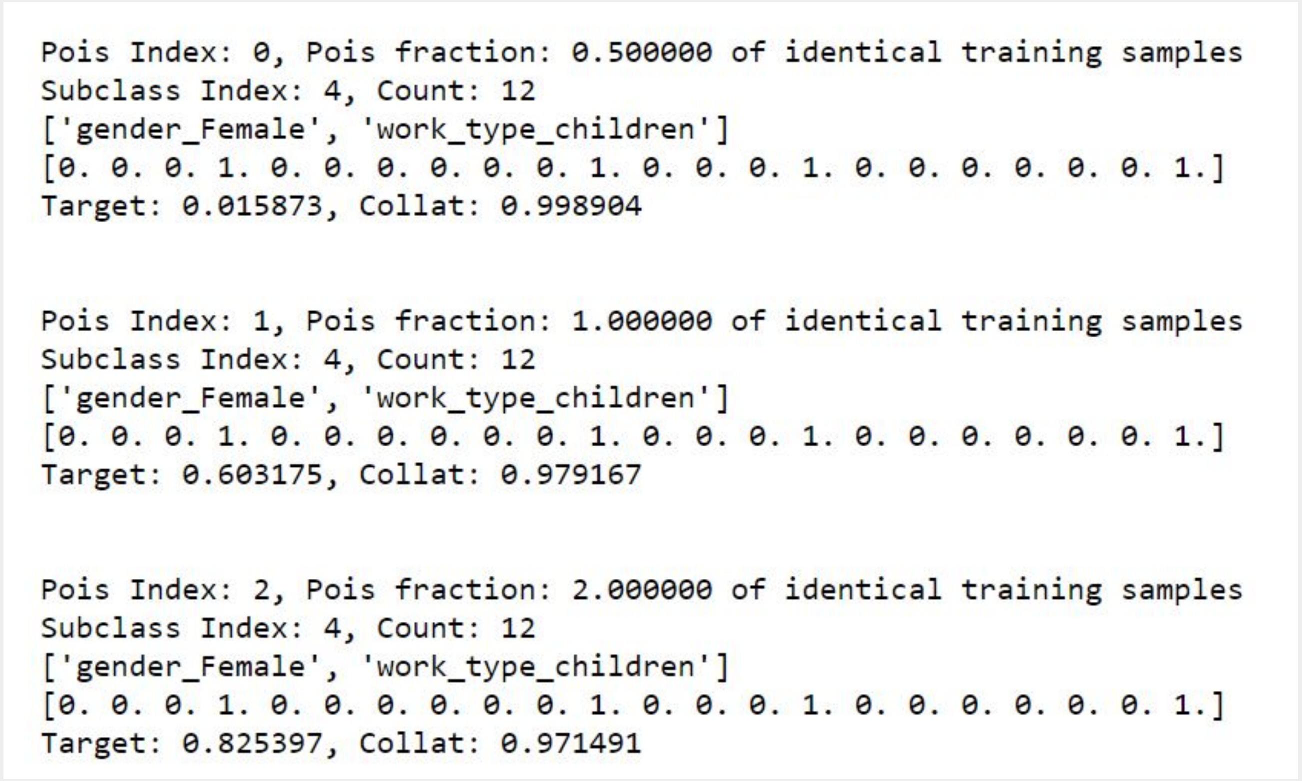


	A	B	C	D	E	F	G	H	I	J	K	L
	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
	9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
	51676	Female	61	0	0	Yes	Self-employed	Rural	202.21	N/A	never smoked	1
	31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
	60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
	1665	Female	79	1	0	Yes	Self-employed	Rural	174.12	24	never smoked	1

- The presented dataset has features such as history of heart disease, married status, work type along with their corresponding labels i.e. stroke
- For the purpose of this subpopulation attack, I have selected the features:gender, heart disease, married status and work type.
- Now, we select unique subpopulation such these feature and add poison data into our dataset.

Note: poison data is just flipped the label from the auxiliary dataset and adding it to our dataset.

Features that provided best results



```
Pois Index: 0, Pois fraction: 0.500000 of identical training samples
Subclass Index: 4, Count: 12
['gender_Female', 'work_type_children']
[0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1.]
Target: 0.015873, Collat: 0.998904


Pois Index: 1, Pois fraction: 1.000000 of identical training samples
Subclass Index: 4, Count: 12
['gender_Female', 'work_type_children']
[0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1.]
Target: 0.603175, Collat: 0.979167


Pois Index: 2, Pois fraction: 2.000000 of identical training samples
Subclass Index: 4, Count: 12
['gender_Female', 'work_type_children']
[0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1.]
Target: 0.825397, Collat: 0.971491
```

Subpopulation Attack Implementation on Credit Risk Dataset

- Implemented the subpopulation on the Credit Risk Dataset.
- The model when trained on this dataset predicts whether a borrower will default his/her loan or not given a number of attributes about the borrowers and the loan they have taken.
- Poisoned two feature class in the dataset. The first poisoned feature subclass was for borrowers with loan interest rate between 19-20%, the second feature subclass was for the borrowers with loan grade G class.
- In both the subpopulation poisoning, the accuracies on the data samples belonging to respective poisoned subclass in the test dataset were lowered signifying a successful data poisoning attack
- Notebook: subpopulation_attack_credit_risk.ipynb

Future Works:

- Experimenting more with Trash Bucket Problem.
- Gain familiarity with Machine Learning algorithms like Reinforcement Learning and GANS.
- Interested to work in Privacy Preserving Machine Learning (PPML) as a next step.

Thank You !!!