Exam project Report, Cyber Security

Riccardo Facciotti Cima, 722817

A.A. 2022/2023

Effectiveness of Adversarial Training in California Housing and Minimum Temperature in Melbourne datasets and benefit-cost analysis

Summary

Overview on evasion attack

- What is evasion attack.
  An evasion attack is an attack perpetrated against a machine learning model, which consists in using a specially modified input in order to induce the model to make a prediction error.
  The effectiveness of the attack depends on how vulnerable the targeted model is and how well the attack was crafted. This second variable depends in turn on the degree of knowledge that the attacker has towards the model and the data used to train him.
  Depending on this level of knowledge, evasion attack can be divided into:
  - white box (the opponent knows all the details of the model);
  - black box (the opponent has no access to the model);
  - gray box (all intermediate cases, for example one in which the opponent can insert any input into the network and observe the output).

- Practical example in image recognition.
  A very effective example to understand the causes and effects is that in the field of image recognition.
  Let's imagine we have a convolutional neural network trained to correctly classify dogs and cats. As long as I submit canonical images of dogs and cats to the neural network, it will make predictions consistent with its degree of accuracy. However, if I take the image of a dog and, pixel by pixel, I add or remove a small amount of color, in order to voluntarily obtain a decrease in the accuracy of the network, a parameter that I can monitor if I have a network similar to that attacked or exactly the attacked network, the resulting image will be imperceptibly different from the original, but the result of the network will be drastically worse.
  A concrete example is shown in [1] in which the FGSM attack is used to generate adversarial examples that mislead a neural network trained to do face recognition.

- Example in a different field from image recognition, malware recognition.
  Another case, less popular, but decidedly important, is that of the evasion attack on prediction models that use tabular data as input.
  An interesting example due to its potential consequences is that of malware classification. On the basis of a series of static characteristics of the software under examination and of dynamic characteristics obtained from its execution inside sandboxes, it is possible to create prediction models which say, with a certain probability, whether it is malicious software or not. If these models are vulnerable to evasion attacks, it is possible to slightly and automatically modify some features of the malware, making it pass as harmless and exposing computer systems to potential risks.
  An example is shown in [2] where adversarial examples are generated starting from the features API of the software from a dataset collected by McAfee by adding APIs that do not modify the effects of the code execution, but induce the model to think that such software is harmless.

How to avoid

- Why is there?
  The fact that some machine learning models are vulnerable to evasion attacks is given by the fact that in the development process of the machine learning model and during its operation, there is a lack of precautions, mechanisms or components that should prevent this type of attack. The Enisa Document for Securing Machine Learning Algorithms [3] contains a list of vulnerabilities that cause little robustness to the various types of attacks against machine learning algorithms including the evasion attack, and above all, for each vulnerability, a series of countermeasures is reported to mitigate them.

- Which vulnerabilities allow this attack and how to mitigate?
  - For example, a vulnerability that makes a model less robust to evasion attacks is the lack of tools capable of discriminating harmless images from images modified with malicious intentions. This can be done, for example, with a model upstream of the prediction pipeline, in which input is given and which tells whether the image can continue or must be discarded.
  - Another vulnerability is the not very resilient design of the model implied by the choice of the objective function, of the regularization techniques, of the hyperparameters.
  - Another vulnerability, which will also be the one I will deal within this project, is the absence of adversarial example in the training set. The operating principle is as follows: if I know that my model has difficulty processing a certain type of input, I myself generate the malicious inputs from which I want to defend myself and I teach my model to process them correctly. In fact, the solution to this vulnerability is to generate adversarial examples and use them in the model training phase.

Experiment

- Goal of the experiment.
  The goal of this project is to evaluate the possibility of integrating within the design and development process of ML models, in particular in the model training phase, an automatic dataset augmentation procedure by adding a certain percentage of adversarial examples to the training set, in order to make the model more robust to a given set of attacks.
  To evaluate the effectiveness of an intervention of this type, it is necessary to carry out experiments in which the training time of the base model and its relative robustness to an attacked testset are measured, and then compared with the training time of a model with augmented training set (obviously also taking into account the generation time of the adversarial examples) and relative robustness to the attacked testset itself.
  If the cost/benefit ratio is appreciable, it means that the idea is good.

- Steps of the experiment.
  The steps of a typical experiment are the following.
  1. Dataset loading and related data preparation (reshape, normalization, subdivision)

2. Base model creation
3. Training 1 with original training set, and time measurement
4. Training 2 with augmented training set, and time measurement
5. Test 1 with model 1 and attacked testset, and performance measurement
6. Test 2 with model 2 and attacked testset, and performance measurement
7. Comparison of time1/performance1 and time2/performance2

Datasets description

- Why this two type of dataset?
  The two datasets that I have decided to use are the "California housing price dataset", of the tabular type, and the "Minimum temperature in Melbourne dataset", of the time series type. Among the various types of datasets we wanted to focus on two very different datasets to have a sort of generality in the results: obtaining similar results on two very different datasets is an indication of the fact that the tested hypothesis holds for a wide range of situations.

- Description of California housing dataset (tabular).
  The dataset was retrieved from sklearn.datasets and in this Kaggle page [4] there is this description: "The data pertains to the houses found in a given California district and some summary stats about them based on the 1990 census data." There are 20.600 rows and the columns are as follows:
  1. longitude;
  2. latitude;
  3. median_income;
  4. house_age;
  5. avg_rooms;
  6. avg_bedrooms;
  7. population;
  8. avg_occupation.

- Daily Minimum temperature in Melbourne dataset (time series).
  This dataset is taken from Kaggle [5], and is composed of 3650 rows which contain
  1. date (from 1/1/1981 to 31/12/1990);
  2. minimum daily temperature of the corresponding date.

Models' description

- NN with linear activation for tabular dataset

```
model_housing = Sequential()
model_housing.add(Dense(15, input_dim=8, activation='relu'))
model_housing.add(Dense(1, activation='linear'))
model_housing.compile(loss='mse', optimizer='adam')
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 15)                135

 dense_1 (Dense)             (None, 1)                 16

=================================================================
Total params: 151
Trainable params: 151
```

- RNN for time series dataset

```
model_temperature = Sequential()
model_temperature.add(LSTM(4, input_shape=(1,look_back)))
model_temperature.add(Dense(1))
model_temperature.compile(loss='mean_squared_error', optimizer='adam')
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 4)                 240

 dense_4 (Dense)             (None, 1)                 5

=================================================================
Total params: 245
Trainable params: 245
```

Attacks description

- FGSM

  Is a very simple and very fast adversarial example generation method that is generally used when dealing with image recognition. It consists in calculating the gradient of the loss function of the model at point x and using as modified sample the same x to which I add or subtract the quantity epsilon to each of its components, depending on the sign of the gradient components. Applying this method to a real value prediction task works in the same way.

  I thought about using this method because it is very simple and therefore the strengthening of the model in the face of this attack can be easily inserted in the design and construction process of a model, raising the safety standard at a probably low price.

- PGD

  Is a slightly more complex attack than FGSM, and consists of 4 steps:

  1. given x, start from a random perturbation in the sphere of radius r considered the distance L_p;
  2. take an epsilon-length step in the direction of maximum gradient rise;
  3. Project the new x back into the sphere of radius r;
  4. repeat 2 and 3 until convergence or reaching the maximum number of iterations;

  I chose this attack because it is still a perturbation of the maximum epsilon input, like FGSM, but more thoroughly researched, therefore probably more effective. I find it curious

to compare these two similar attacks but with different effectiveness to then see if the adversarial training done with one is also effective in countering the other.

Metrics description

- A model is all the more robust to evasion attacks the more it is able to correctly classify the samples that are provided to it despite changes in the feature space. M1, adversarial trained is more robust then M2, not adversarial trained, if the error of M1, during the testing phase with an attack, is lesser then the M2 one's and the difference is statistically significant. The metrics, to measure the errors that I decided to use are:
    1. R2, span from negative infinity to 1.0, it gives a quick indication of how well the model is predicting;
    2. MAE, spans from 0 to positive infinity, it tells how far the predictions are from the true values, on average;
    3. MSE, spans from 0 to positive infinity, it tells how far the predictions are from the true values, on average, penalizing large errors more than small ones.

Experiments summary

- In order to have a global and summary view of the experiments conducted, I put below a list of the parameters that vary in the experiments, the single combinations of which correspond to a single experiment, for a total of 40, and the list of metrics to evaluate the robustness and efficiency of the generated models.
- Parameters:
    1. Datasets: [‘California Housing’, ’Daily minimum temperature in Melbourne’]
    2. Adversarial training percentage: [0.0, 0.25, 0.50, 0.75, 0.1]
    3. Train attack: [‘fgsm’, ‘pgd’]
    4. Test attack: [‘fgsm’, ‘pgd’]
- Metrics:
    1. T = Training time + adversarial generation time
    2. R2
    3. MAE
    4. MSE
    5. R2/T
    6. MAE/T
    7. MSE/T

Results

Table of all results

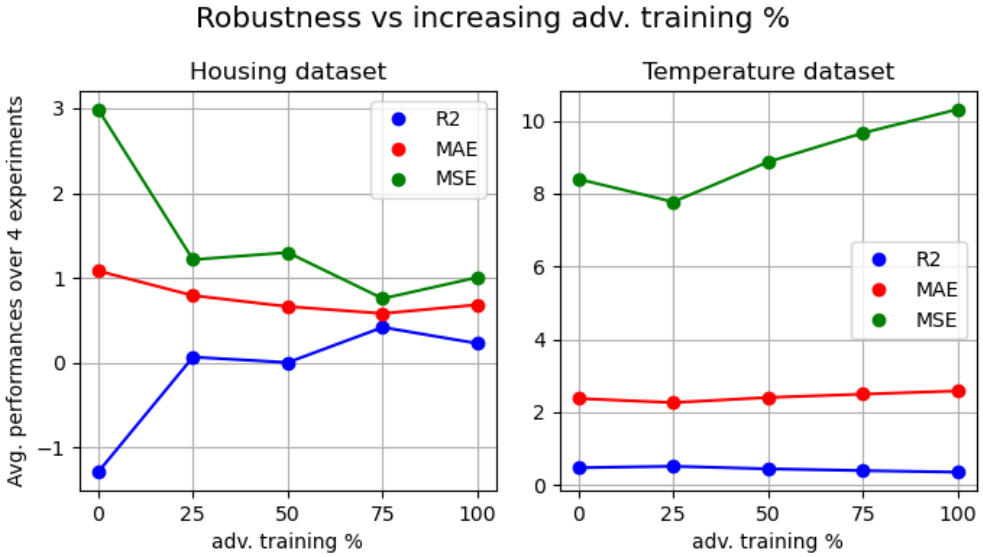
experiment	dataset	train	test	adversarial training	training time (s)	R2	MAE	MSE	R2/time	MAE/time	MSE/time
1	California Housing Dataset	FGSM	FGSM	0%	44,0309	0,0249	0,8722	1,2742	0,0006	0,0198	0,0289
2				25%	85,8504	0,0754	0,8852	1,2081	0,0009	0,0103	0,0141
3				50%	99,6314	-0,2144	0,6510	1,5869	-0,0022	0,0065	0,0159
4				75%	189,0869	0,5312	0,5486	0,6126	0,0028	0,0029	0,0032
5				100%	159,4417	0,0941	0,6891	1,1838	0,0006	0,0043	0,0074
6			PGD	0%	44,0309	0,1453	0,7965	1,1168	0,0033	0,0181	0,0254
7				25%	85,8504	0,3188	0,6866	0,8901	0,0037	0,0080	0,0104
8				50%	99,6314	-0,1581	0,6043	1,5133	-0,0016	0,0061	0,0152
9				75%	189,0869	0,5049	0,5719	0,6469	0,0027	0,0030	0,0034
10				100%	159,4417	0,0297	0,7343	1,2679	0,0002	0,0046	0,0080
11		PGD	FGSM	0%	43,9648	-2,9524	1,4375	5,1647	-0,0672	0,0327	0,1175
12				25%	1445,4351	-0,2122	0,9250	1,5840	-0,0001	0,0006	0,0011
13				50%	1830,7840	0,1269	0,7554	1,1409	0,0001	0,0004	0,0006
14				75%	4437,8610	0,3367	0,5902	0,8668	0,0001	0,0001	0,0002
15				100%	4360,3116	0,4006	0,6554	0,7832	0,0001	0,0002	0,0002
16			PGD	0%	43,9648	-2,3661	1,2296	4,3986	-0,0538	0,0280	0,1000
17				25%	1445,4351	0,0920	0,6784	1,1865	0,0001	0,0005	0,0008
18				50%	1830,7840	0,2646	0,6455	0,9610	0,0001	0,0004	0,0005
19				75%	4437,8610	0,3058	0,6182	0,9071	0,0001	0,0001	0,0002
20				100%	4360,3116	0,3915	0,6619	0,7951	0,0001	0,0002	0,0002
21	Minimum Daily Temperature in Melbourne Dataset	FGSM	FGSM	0%	75,8032	0,4576	2,4420	8,6458	0,0060	0,0322	0,1141
22				25%	106,6154	0,5048	2,3014	7,8936	0,0047	0,0216	0,0740
23				50%	136,5940	0,4379	2,4186	8,9604	0,0032	0,0177	0,0656
24				75%	162,8418	0,4002	2,4691	9,5612	0,0025	0,0152	0,0587
25				100%	184,7826	0,3838	2,5114	9,8234	0,0021	0,0136	0,0532
26			PGD	0%	75,8032	0,5115	2,2666	7,7876	0,0067	0,0299	0,1027
27				25%	106,6154	0,5595	2,1243	7,0218	0,0052	0,0199	0,0659
28				50%	136,5940	0,4547	2,3729	8,6926	0,0033	0,0174	0,0636
29				75%	162,8418	0,4136	2,4429	9,3477	0,0025	0,0150	0,0574
30				100%	184,7826	0,3799	2,5264	9,8851	0,0021	0,0137	0,0535
31		PGD	FGSM	0%	88,0081	0,4417	2,4551	8,9000	0,0050	0,0279	0,1011
32				25%	466,1594	0,4653	2,3975	8,5236	0,0010	0,0051	0,0183
33				50%	834,8211	0,4316	2,4303	9,0617	0,0005	0,0029	0,0109
34				75%	1177,5121	0,3620	2,5757	10,1698	0,0003	0,0022	0,0086
35				100%	1575,4919	0,3074	2,6834	11,0415	0,0002	0,0017	0,0070
36			PGD	0%	88,0081	0,4819	2,3257	8,2586	0,0055	0,0264	0,0938
37				25%	466,1594	0,5201	2,2232	7,6504	0,0011	0,0048	0,0164
38				50%	834,8211	0,4500	2,3842	8,7683	0,0005	0,0029	0,0105
39				75%	1177,5121	0,3990	2,4857	9,5805	0,0003	0,0021	0,0081
40				100%	1575,4919	0,3407	2,6063	10,5093	0,0002	0,0017	0,0067

Intuition from result:

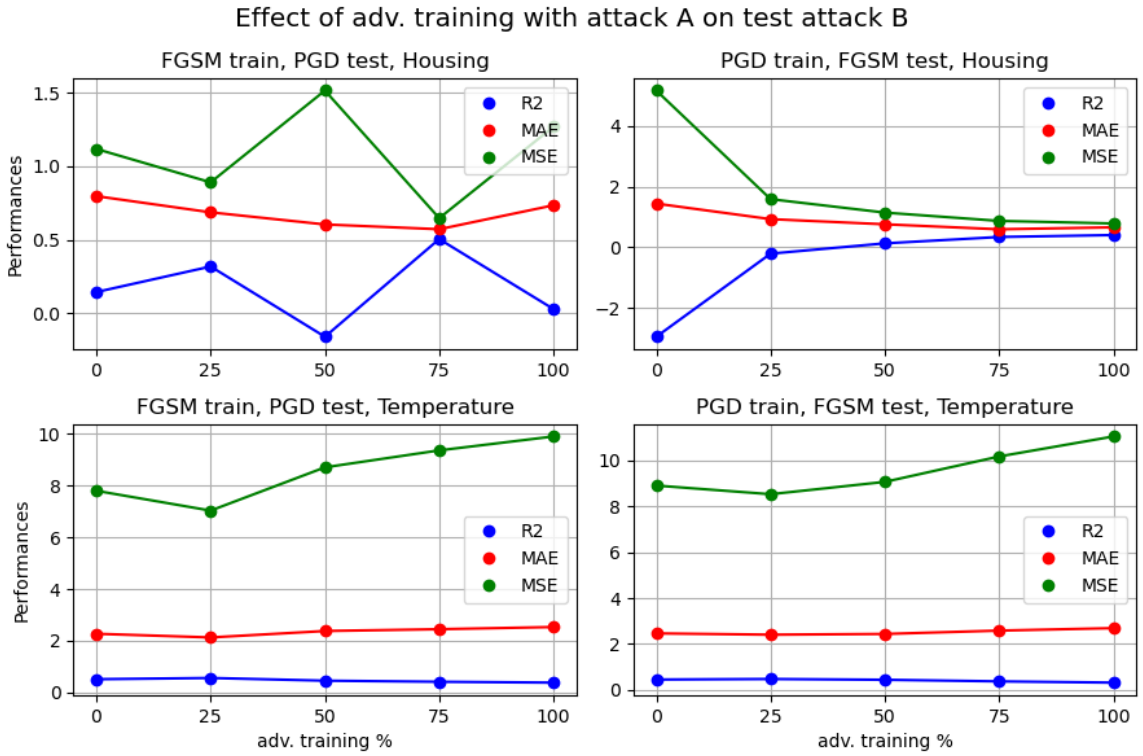
- Is it more harmful fgsm or pgd?
  Looking at the base model attacked with fgsm, we note that the associated metrics indicate lower robustness than those of the same model attacked with pgd. I explain this with 3 assumptions:
  1. the pgd function that I have implemented does not have random restarts, so it may be that the solution found is local minima of low quality;
  2. I didn't do parameter tuning on pgd method, so it may be that the number of iterations and the step size are not very effective;
  3. The norm I used in pgd method is the Euclidean one, so the adversarial example it produces is closer to the starting point than the one fgsm produces: the latter is a vertex of the hypercube of side r, the first is inside the hypersphere of radius r. So if the loss is not made in a strange way above the domain, it may be that the farther the example is from the starting point, the bigger the prediction error.

	Training time	R2	MAE	MSE
Housing	44.030864	0.024905	0.872241	1.274186
FGSM attack	43.964837	-2.952391	1.437504	5.164704
Temperature	75.803202	0.457641	2.442041	8.645846
FGSM attack	88.008097	0.441695	2.455148	8.900049
Housing	44.030864	0.145314	0.796478	1.116843
PGD attack	43.964837	-2.366096	1.229630	4.398576
Temperature	75.803202	0.511482	2.266558	7.787567
PGD attack	88.008097	0.481935	2.325668	8.258568

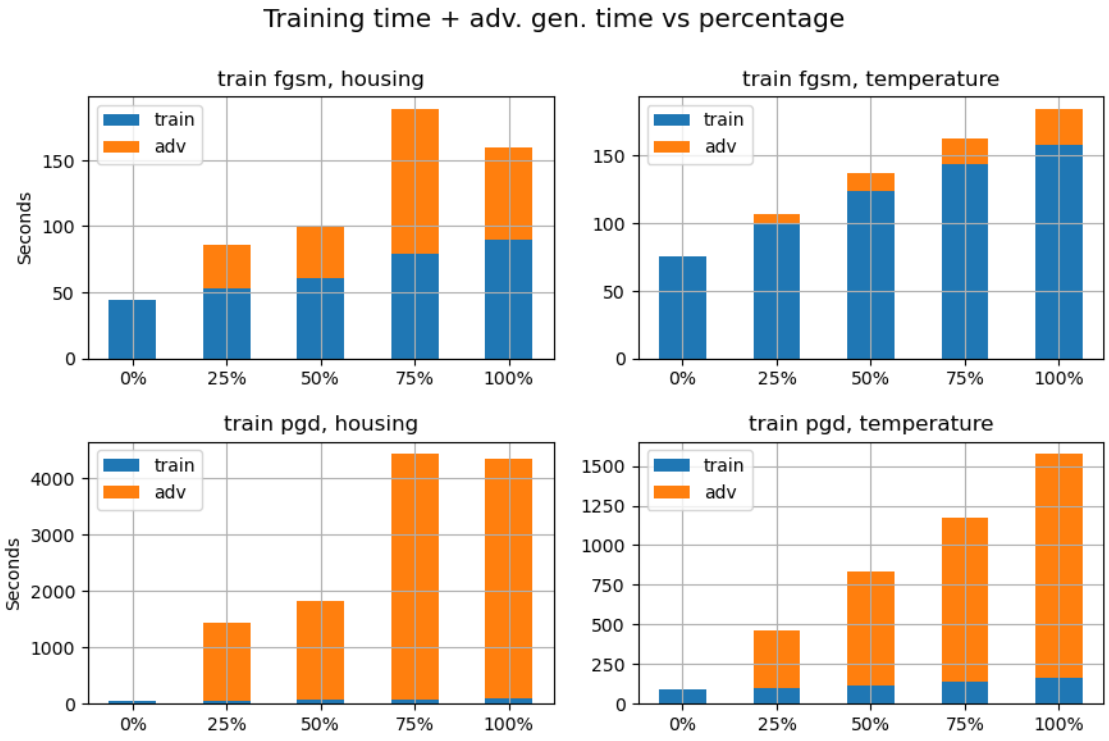
- Is increasing percentage increasing the robustness?
  One of the objectives of the experiment is to understand if by increasing the percentage of modified training set, i.e. by increasing it with its own samples but modified by an attack, the robustness to the same or similar attack would also increase. The graph shows that in the housing dataset the average of the MAE on 4 experiments has a decreasing trend up to 75%, as does the MSE and at 100% the error goes up, while the R2 has an increasing trend up to 75% and at 100% it goes down. This suggests that adversarial training has a positive effect on the robustness of the model, and as regards the deterioration in the percentage value of 100%, it is not possible to guess much, because it could be both a local and a global minimum in the parameter's optimality.
  Instead, as regards the temperature set, the trends are reversed, as if to indicate that adversarial training has no positive effects with this particular conditions.



- What is the impact of training with attack A on testing with attack B?
  Another thing we wanted to understand with the experiments was whether adversarial training performed with an attack A was effective in increasing the robustness of an attack B as well. In the housing dataset, in the experiments in which I train with FGSM and attack with PGD, the metrics fluctuate a lot depending on the percentage values, except for the MAE which seems to go down. When, on the other hand, I train with PGD and attack with FGSM, all the metrics indicate an improvement that increases as the percentage value increases. Also in this case in the temperature dataset, in all the cases considered, the post adversarial training metrics seem to indicate a deterioration.



- What is the time overhead?
  As can be seen from the graph below, the increase in time required for generation of adversarial example + model training is not linear. From 0 to 0.25, in housing dataset and fgsm, for example the time doubles, while from 25 to 50, it increases by 16%, then from 0.5 to 0.75, the time doubles again and finally, strange case, from 0.75 to 1.0, time decreases. Same conditions but by changing the dataset, the time trend becomes more regular: from 0 to 0.25 it increases by 41%, from 0.25 to 0.5 by 28%, from 0.5 to 0.75 by 19% and from 0.75 to 1.0 by 13%.
  Instead, when we take into consideration pgd as generator of adversarial example the times are very long: with housing dataset from 0 to 0.25 the time is almost 34 times greater, then from 0.25 to 0.5 it increases by 27%, from 0.5 to 0.75 it increases by 4 times and from 0.75 to 1.0 the time decreases. With dataset temperatures and always pgd, from 0 to 0.25 the time is 5 times greater, from 0.25 to 0.5 it increases by 79%, from 0.5 to 0.75 it increases by 41%, from 0.75 to 1.0 it increases by 34%.



- Why is plain model robustness similar to that of adv trained models?
  In the papers that talk about evasion attacks applied to the field of image recognition or malware detection, we note a great effectiveness of this attack. However, from the results I obtained from the experiments conducted, the difference between the performance of the basic model and the various modified models are not striking. Between the average of the MAE of the base model on all experiments and the average of the MAE of the model at 0.25 on all experiments, the difference is only 0.2 points and on average it is also the percentage value that has the smallest error (experiments with the temperature dataset raise the average, because under the conditions of the experiments, adversarial training seems to be counterproductive). The same trend emerges with the other metric parameters. I think that the discrepancy between these results and those obtained in the literature with classification tasks are due to the fact that these two tasks require two different output functions and this difference accentuates the displacement of y in the face of small variations of x: if the output function is a sigmoid, I just need to move slightly x to notice a relevant difference in y, in case x, after the various transformations, is close to 0, while it will be more difficult to notice the difference and get the model wrong, between $f(x)$ and $f(\hat{x})$ if the output function is a hyperplane, because small variations of x produce small variations of y (depending on the gradient).

percentage	Mean R2	Mean MAE	Mean MSE	Std R2	Std MAE	Std MSE
0	-0.406939	1.728158	5.693292	1.409679	0.718994	3.215933
25	0.290480	1.527698	4.494778	0.277925	0.792781	3.532314
50	0.224139	1.532793	5.085647	0.278352	0.929829	4.052721
75	0.406683	1.537781	5.211586	0.077791	1.022439	4.767301
100	0.290968	1.633544	5.661161	0.145601	1.015424	4.991974

Conclusions

- Is it worth including adversarial training in the development process?
  The following conclusions obviously refer to the particular conditions of these experiments, with the particular parameters and related values used and the various implementation details of the case (model architecture, attacks implementation). This premise is simply to avoid generalizing results without logical foundations.
  Having said that, I can observe that the robustness of the models with adversarial training measured with MAE does not increase more than 0.2 points and with MSE no more than 1.2, therefore a modest improvement, while the time spent on generating adversarial examples and respective training, only passing from the basic model to the one with 0.25 more points in the training set, it doubles, and this in the case of FGSM, i.e. the least expensive attack. The most disturbing case is when I use PGD to generate adversarial example, the most onerous attack: the time from base model to 0.25 in the housing dataset increases by 34 times, and there are also disproportionate increases passing in the other percentage values and in the other dataset.

There are cases in which the safety of a model, on the scales, weighs more than performance, however, given the results obtained, I believe it is smarter to look for more efficient methods of strengthening.

Bibliography

[1] Arbena Musa, Kamer Vishi & Blerim Rexha, "Attack Analysis of Face Recognition Authentication Systems Using Fast Gradient Sign Method", 16 Sep 2021

[2] Yonghong Huang, Utkarsh Verma, Celeste Fralick, Gabriel Infante-Lopez, Brajesh Kumar, Carl Woodward, "Malware Evasion Attack and Defense", 2019

[3] ENISA, "Securing machine learning algorithms", dec 2021

[4] www.kaggle.com/datasets/camnugent/california-housing-prices, accessed date: 15 April 2023

[5] www.kaggle.com/datasets/paulbrabban/daily-minimum-temperatures-in-melbourne, accessed date: 15 April 2023