Vulnerability of machine learning models to adversarial examples

Petra Vidnerová Roman Neruda

Institute of Computer Science
Academy of Sciences of the Czech Republic

ITAT 2016

Introduction

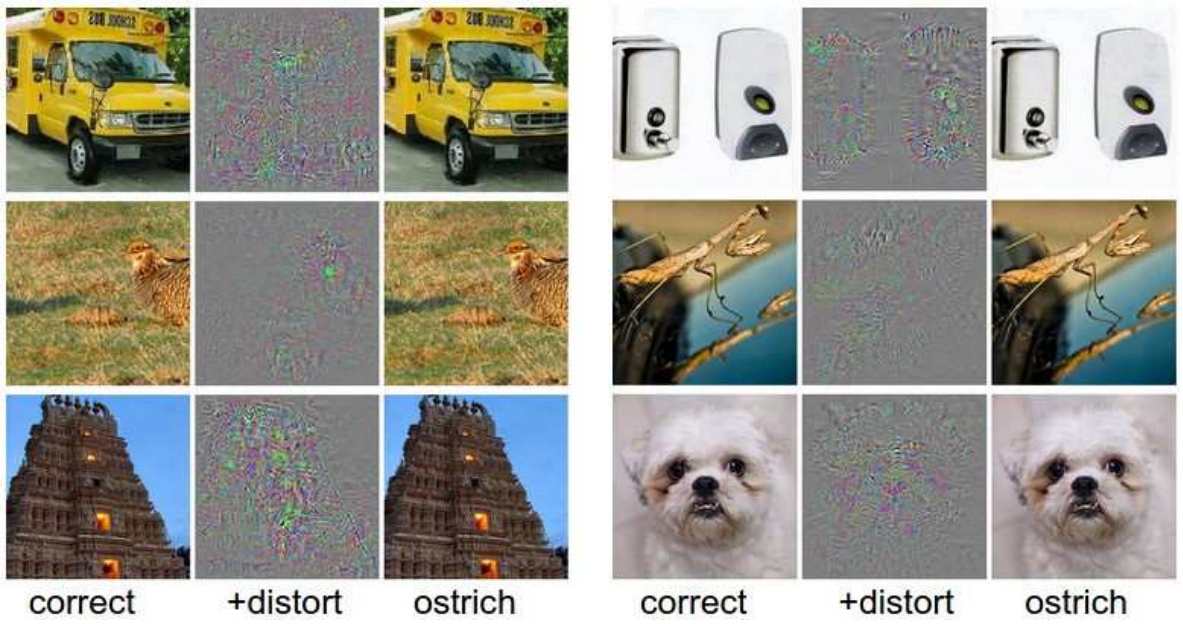
- Applying an imperceptible non-random perturbation to an input image, it is possible to arbitrarily change the machine learning model prediction.

Figure from *Explaining and Harnessing Adversarial Examples* by Goodfellow et al.

- Such perturbed examples are known as *adversarial examples*. For human eye, they seem close to the original examples.
- They represent a security flaw in classifier.

Works on adversarial examples I.

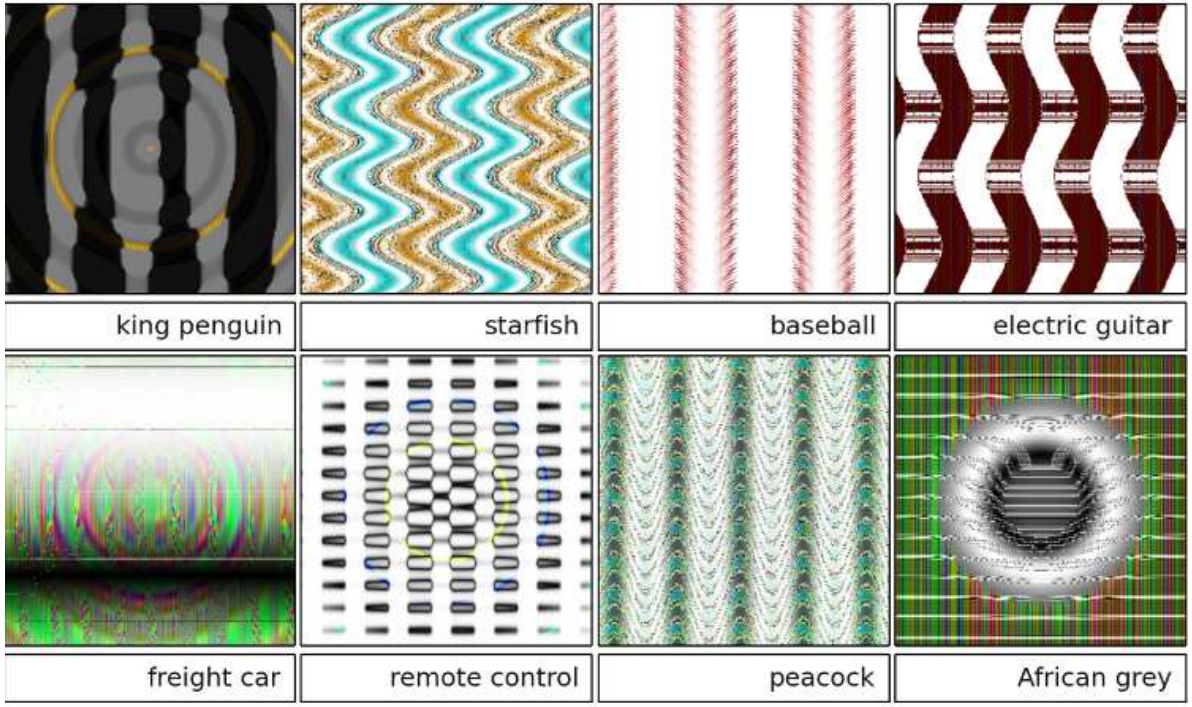
- *Intriguing properties of neural networks*. 2014,Christian Szegedy et al.



- Perturbations are found by optimising the input to maximize the prediction error (L-BFGS).

Works on adversarial examples II.

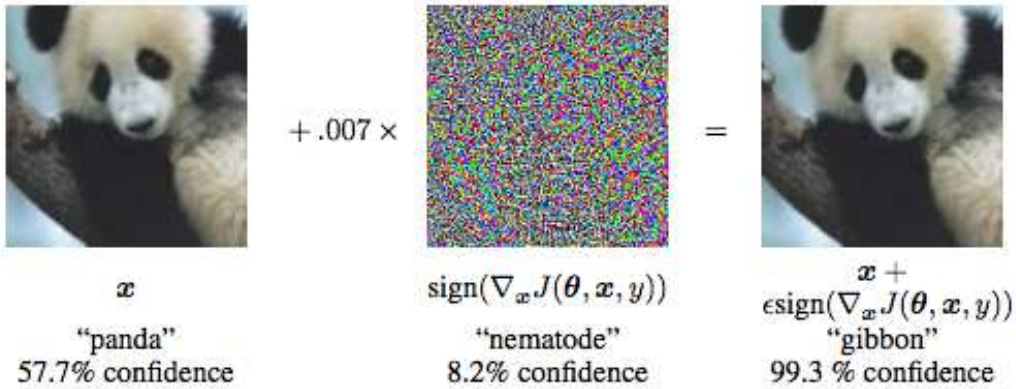
- *Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images*
  2015,Anh Nguyen, Jason Yosinski, Jeff Clune



- evolutionary generated images

Works on adversarial examples III.

- *Explaining and Harnessing Adversarial Examples* 2015,Goodfellow et al.



- adding small vector in the direction of the sign of the derivation
- linear behaviour in high dimensional spaces is sufficent to cause adversarial examples

Our work

- genetic algorihms used to search for adversarial examples
- tested various machine learning models including both deep and shallow architectures

Machine learnig models I.

Shallow architectures

- *RBF network* — feedforward network with one hidden layer, linaer output layer. Local units realizing Gaussian function.
- *SVM* — one hidden layer of kernel units, linear output layer. Learning is based on searching for a separating hyperplane with the highest margin.
  Commonly used kernels:
  - linear $\langle x, x' \rangle$
  - polynomial $(\gamma \langle x, x' \rangle + r)^d$
  - Gaussian $\exp(-\gamma |x - x'|^2)$
  - sigmoid $\tanh(\gamma \langle x, x' \rangle + r)$.

Decision tree

Machine learning models II.

Deep architectures

Deep neural networks are feedforward neural networks with multiple hidden layers between the input and output layer. The layers typically have different units depending on the task.

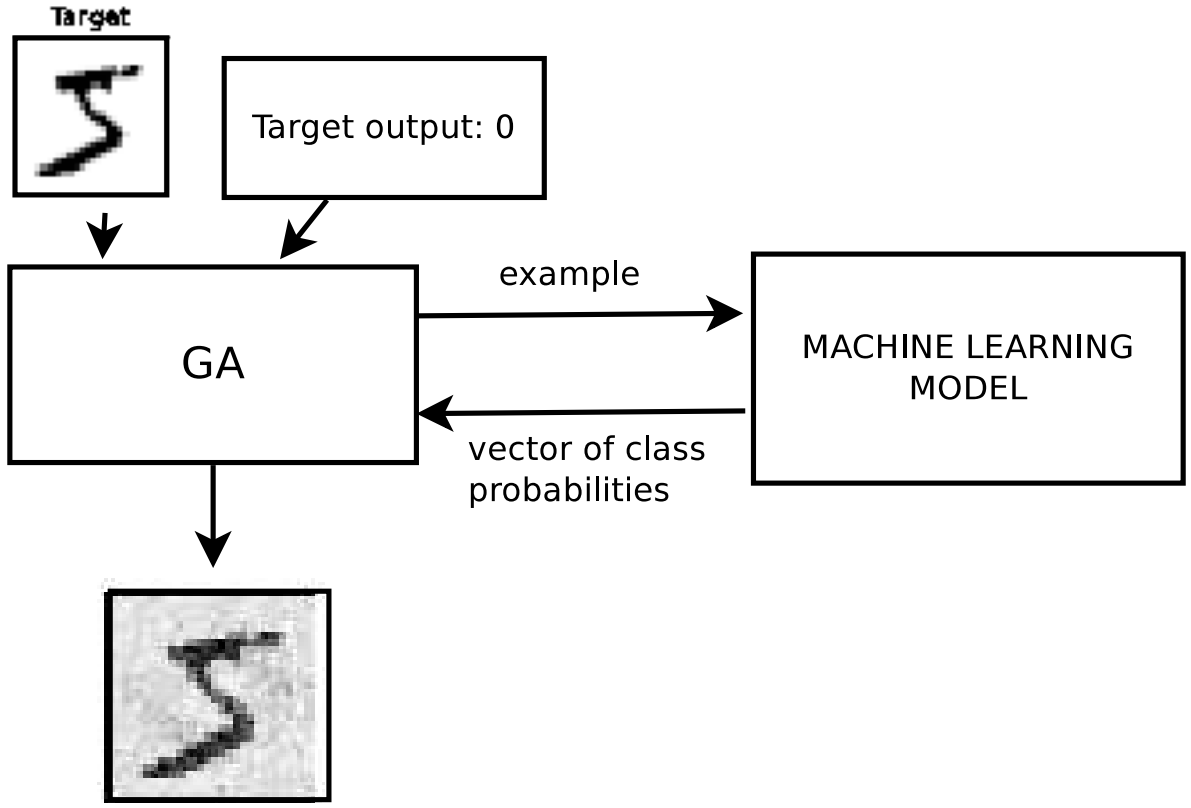
- MLP
  - Perceptron units with *sigmoid* function
  - Rectified linear unit (ReLU): $y(z) = max(0, z)$.
- CNN
  - *Convolutional units* perform a simple discrete convolution operation which for 2-D data can be represented by a matrix multiplication.
  - *max pooling layers* that perform an input reduction by selecting one of many inputs, typically the one with maximal value

Search for adversarial images

- To obtain an adversarial example for the trained machine learning model, we need to *optimize the input image with respect to model output*.
- For this task we employ a GA – robust optimisation method working with the whole population of feasible solutions.
- The population evolves using operators of selection, mutation, and crossover.
- The machine learning model and the target output are fixed.

Black box approach

- genetic algorithms to generate adversarial examples
- machine learning method is a blackbox
- applicable to all methods without the need to acess models parameters (weights)



Genetic algorithm

- *Individual:* image encoded as a vector of pixel values:

$$I = \{i_1, i_2, \ldots, i_N\},$$

  where $i_i \in < 0, 1 >$ are levels of grey and N is a size of flatten image.
- *Crossover:* operator performs a two-point crossover.
- *Mutation:* with the probability p_{mutate_pixel} each pixel is changed:

$$i_i = i_i + r,$$

  where r is drawn from Gaussian distribution.
- *Selection:* $3-$tournament

GA fitness

- The fitness function should reflect the following two criteria:
  - the individual should resemble the target image
  - if we evaluate the individual by our machine learning model, we would like to obtain a target output (i.e. misclassify it).

Thus, in our case, a fitness function is defined as:

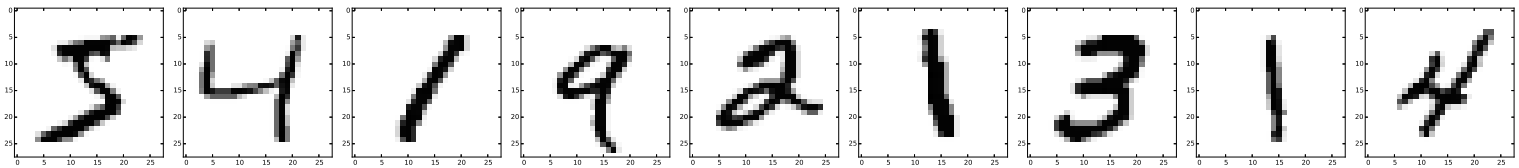
$$f(I) = -(\quad 0.5 * cdist(I, target\_image) \quad (1)$$
$$+ \quad 0.5 * cdist(model(I), target\_answer)), \quad (2)$$

where *cdist* is an Euclidean distance.

Experiments

Dataset

- MNIST dataset
- 70000 images of handwritten digits
- 28×28 pixels
- 60000 for training, 10000 for testing



Overview of models I.

- KERAS library, two models from examples
  - MLP — three fully connected layers, two hidden layers have 512 ReLUs each, using dropout; the output layer has 10 softmax units.
  - CNN — two convolutional layers with 32 filters and ReLUs, each, max pooling layer, fully connected layer of 128 ReLUs, and a fully connected output softmax layer.
  - Ensemble — 10 MLPs

Overview of models II.

- models from SCIKIT learn library
  - SVM — SVM with Gaussian kernel (SVM-rbf), polynomial kernel of grade 2 and 4 (SVM-poly2 and SVM-poly4), sigmoidal kernel (SVM-sigmoid), and linear kernel (SVM-linear).
  - DT — Decision Tree.
- our iplementation
  - RBF — RBF network with 1000 Gaussian units
- grid search and crossvalidation used to tune metaparameters

Classification Acurracy

model	trainset	testset
MLP	1.00	0.98
CNN	1.00	0.99
RBF	0.96	0.96
SVM-rbf	0.99	0.98
SVM-poly2	1.00	0.98
SVM-poly4	0.99	0.98
SVM-sigmoid	0.87	0.88
SVM-linear	0.95	0.94
DT	1.00	0.87

Experimental Setup

GA setup

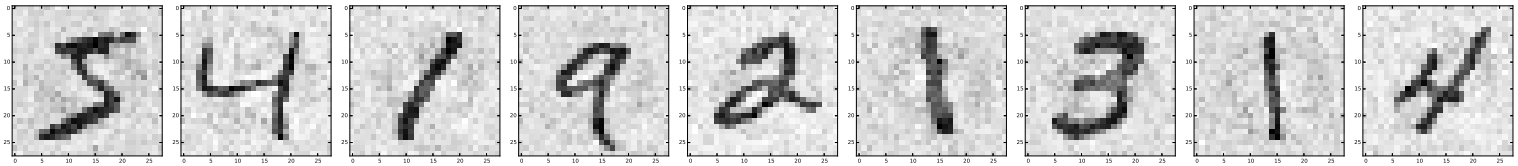
- population of 50 individuals
- 10 000 generations
- crossover probability 0.6
- mutation probability 0.1
- DEAP framework

Images

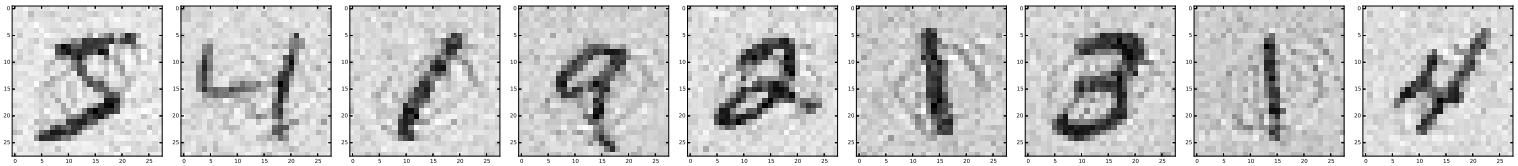
- for 10 first images from training set
- target: classify as zero

Evolved Adversarial Examples I.

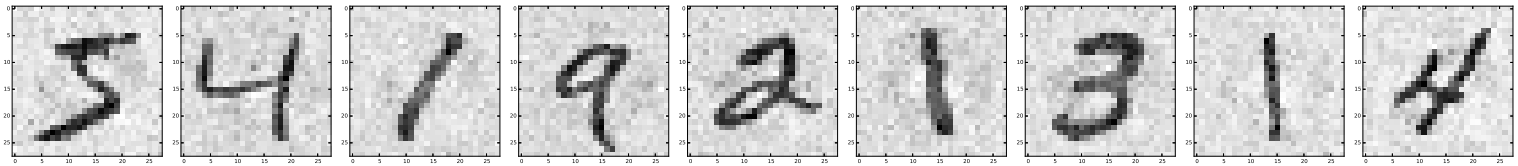
Adversarial examples evolved for MLP:



Adversarial examples evolved for CNN:

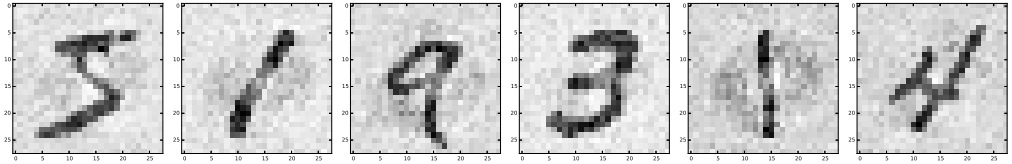


Adversarial examples evolved for ensamble of MLPs:

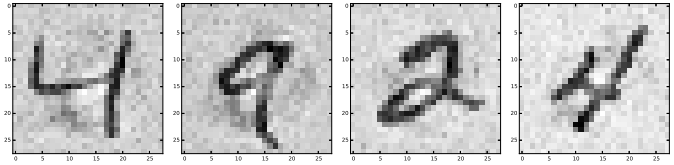


Evolved Adversarial Examples II.

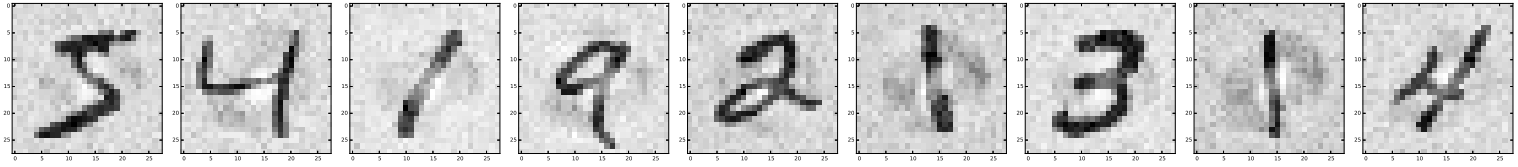
Adversarial examples evolved for SVM-poly2:



Adversarial examples evolved for SVM-poly4:

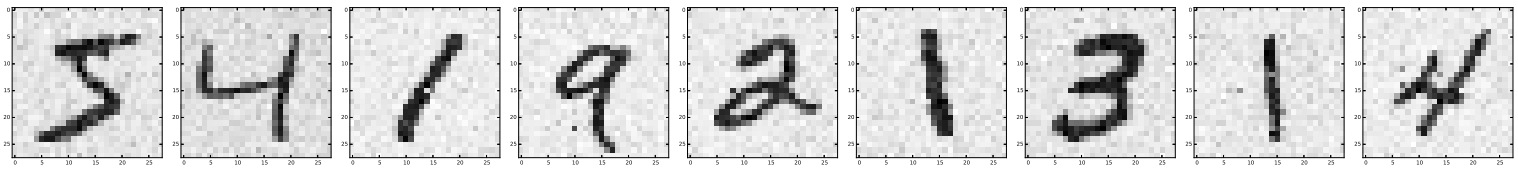


Adversarial examples evolved for SVM-sigmoid:



Evolved Adversarial Examples III.

Adversarial examples evolved for decision tree:



No adversarial examples found for:

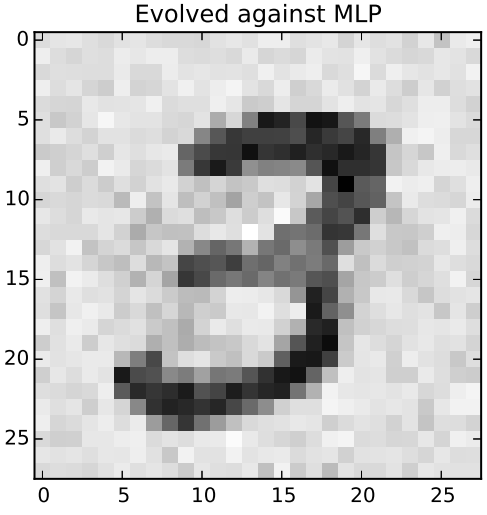
- RBF network
- SVM-rbf
- SVM-linear

Experimental Results

- MLP, CNN, ensemble of MLPs, SVM-sigmoid, and DT were always misclassifying the best individuals
- RBF network, SVM-rbf, and SVM-linear; never misclassified, i.e. the genetic algorithm was not able to find adversarial example for these models;
- SVM-poly2 and SVM-poly4 were resistant to finding adversarial examples in 2 and 5 cases, respectively.

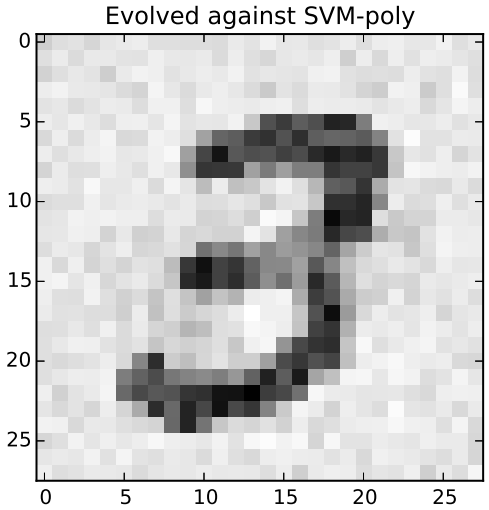
Generalization

- some adversarial examples generated for one model are also missclassified by other models



	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
RBF	0.30	0.04	0.17	**0.75**	0.02	-0.03	-0.04	-0.01	-0.07	-0.00
MLP	**0.96**	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.01
CNN	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
ENS	**0.86**	0.00	0.01	0.04	0.00	0.00	0.00	0.00	0.01	0.08
SVM-rbf	0.00	0.00	0.00	**0.99**	0.00	0.00	0.00	0.00	0.00	0.01
SVM-poly	0.04	0.00	0.01	**0.91**	0.00	0.00	0.00	0.00	0.02	0.02
SVM-poly4	0.03	0.00	0.01	**0.93**	0.00	0.00	0.00	0.00	0.01	0.01
SVM-sigmoid	**0.49**	0.00	0.03	0.30	0.00	0.04	0.00	0.01	0.10	0.02
SVM-linear	0.25	0.02	0.10	**0.30**	0.02	0.05	0.02	0.03	0.18	0.06
DT	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00

Generalization



	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
RBF	0.32	0.02	0.17	**0.86**	-0.01	-0.09	-0.09	-0.03	-0.12	0.01
MLP	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
CNN	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
ENS	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
SVM-rbf	0.00	0.00	0.00	**0.99**	0.00	0.00	0.00	0.00	0.00	0.00
SVM-poly	**0.87**	0.00	0.02	0.04	0.00	0.00	0.00	0.00	0.04	0.02
SVM-poly4	**0.38**	0.01	0.11	0.23	0.01	0.02	0.01	0.02	0.15	0.04
SVM-sigmoid	**0.55**	0.01	0.04	0.19	0.01	0.05	0.01	0.01	0.13	0.02
SVM-linear	**0.71**	0.01	0.02	0.06	0.01	0.02	0.01	0.01	0.15	0.01
DT	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00

Generalization — Summary

- adversarial example evolved for CNN was never misclassified by other models, and CNN never misclassified other adversarial examples than those evolved for the CNN;
- adversarial example evolved for DT was never misclassified by other models, and DT misclassified other adversarial examples only in 4 cases
- adversarial examples evolved for MLP are misclassified also by ensemble of MLPs (all cases except two) and adversarial examples evolved for ensemble of MLPs are misclassified by MLP (all cases);

Generalization — Summary

- adversarial examples evolved by the SVM-sigmoid model are misclassified by SVM-linear (all cases except two);
- adversarial examples for the SVM-poly2 model are often (6 cases) misclassified by other SVMs (SVM-poly4, SVM-sigmoid, SVM-linear), and in 4 cases also by the SVM-rbf. In three cases it was also missclasified by MLP and ensemble of MLPs, in one case, the adversarial example for SVM-poly2 is misclassified by all models but CNN (however, this example is quite noisy);

Generalization — Summary

- adversarial example for the SVM-poly4 model is in two cases misclassified by all models but CNN, and DT; in different case by all but the CNN and RBF models, and in one case by all but CNN, RBF, DT, and SVM-rbf models;

- RBF network, SVM-rbf, and SVM-linear were resistant to adversarial examples by genetic algorithm, however they sometimes misclassify adversarial examples of other models. These examples are already quite noisy, however by human they would still be classified correctly.

Summary

- We have proposed a GA for generating adversarial examples for machine learning models by applying minimal changes to the existing patterns.
- Our experiment showed that many machine models suffer from vulnerability to adversarial examples.
- Models with local units (RBF networks and SVMs with RBF kernels) are quite resistant to such behaviour.
- The adversarial examples evolved for one model are usually quite general – often misclassified also by other models.

Thank you! Questions?