Evolutionary Algorithms (EA)

Omar Mohammed

Optimization

- Search for a solution to a defined problem
    - Unconstrained optimization
    - Constrained optimization
    - Satisfaction problem
    - Multiobjective problems
- Core of machine learning
    - Optimize the neural network weights (gradient descent)
    - Optimize the neural network hyperparameters (random / greedy)

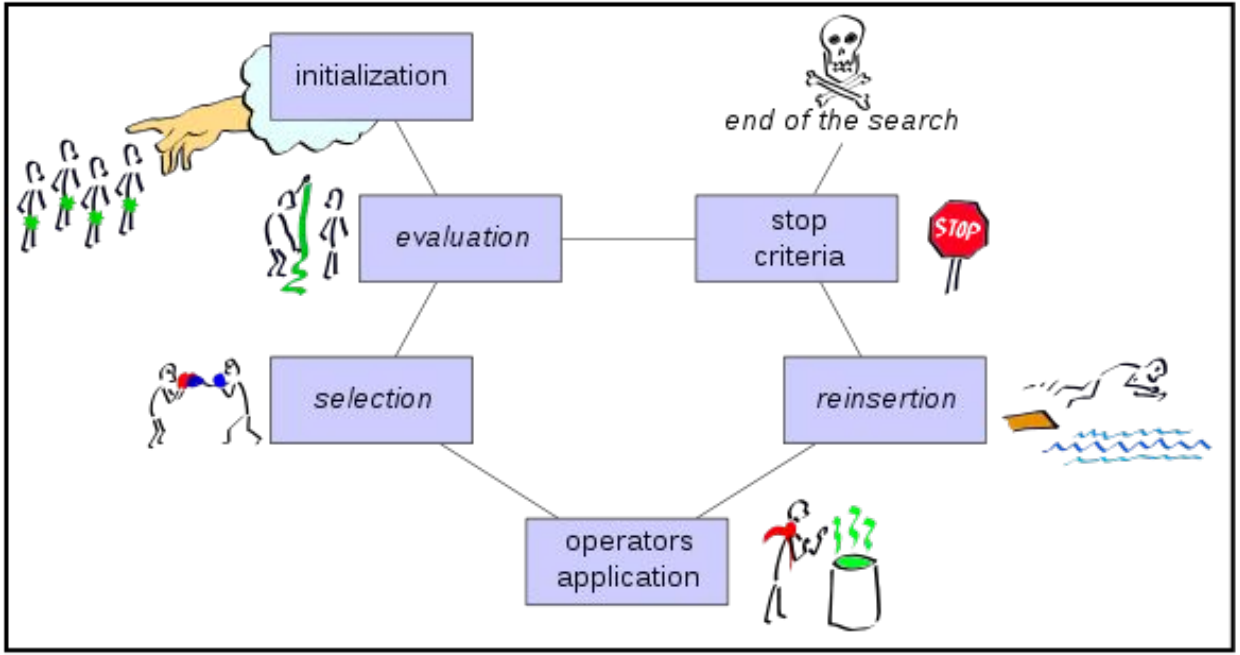
Types of optimization

- Assumption based (more efficient, less general)
  - Convex optimization
  - Gradient descent
    - Convex/non-convex, but differentiable
- Agnostic (more general, less efficient)
  - Random search
  - Meta-heuristics
    - Evolutionary algorithms
    - Genetic programming
    - Ant-colony
    - Bee optimization
    - ….etc

General Steps

1. (initialization) Start with N random solutions
    a. Generation 0
    b. Generation has N individuals
    c. The group of individuals is called a population
2. (evaluation) Evaluate the quality of each solution
3. Are we there yet?
    a. If not, continue
4. (secret sauce)
    a. Select the potential solution
    b. Little modification (mutation) and some sex (crossover)
    c. New children are born
    d. New generation is formed
5. Repeat

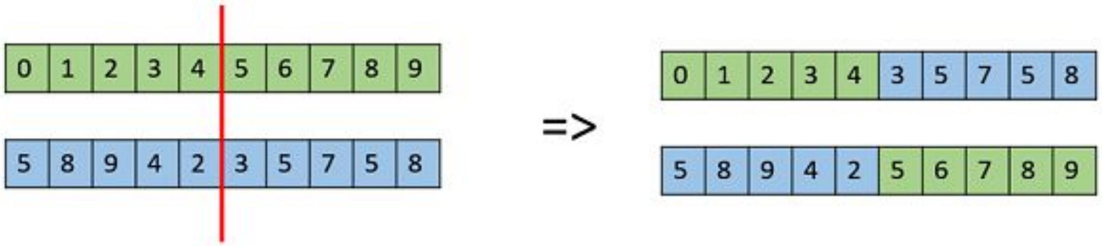
What is an evolutionary algorithm?



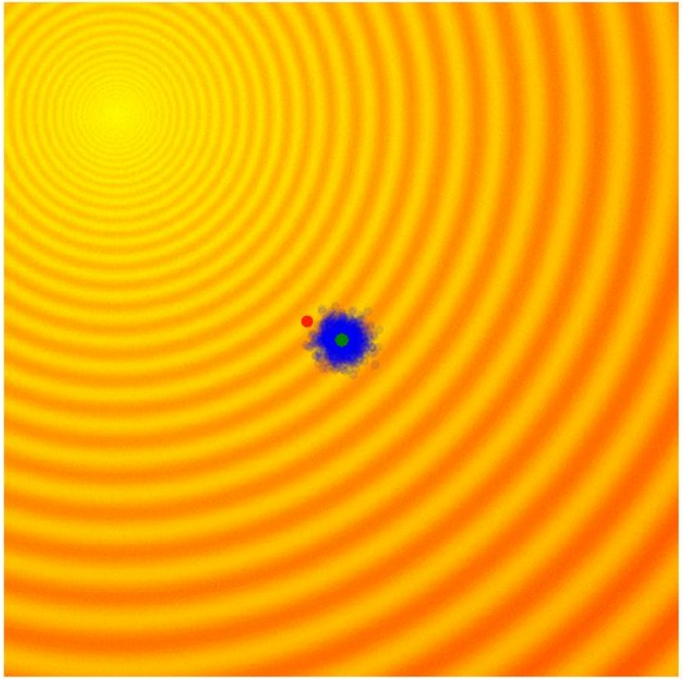
General schema of an Evolutionary Algorithm (EA)

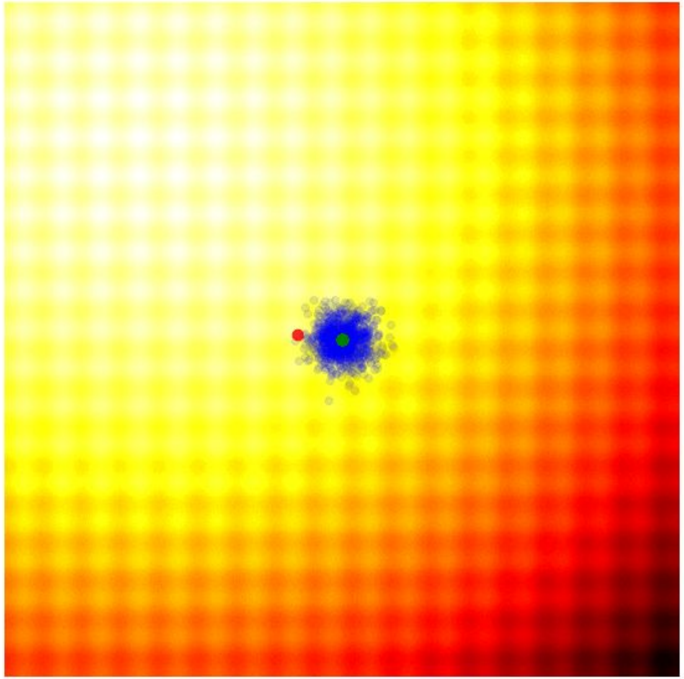
Secret sauce

- Mutation
  - Randomly change part of the individual
    - Ex: Add little random noise to the weights of a neural network
  - Encourage exploitation
- Cross-over
  - Exchange of genes
    Between two individuals
  - Encourages exploration
  - Harder to design
- Most researchers stick to Mutation only
  - Lots of work support this idea



Simple genetic algorithm performance



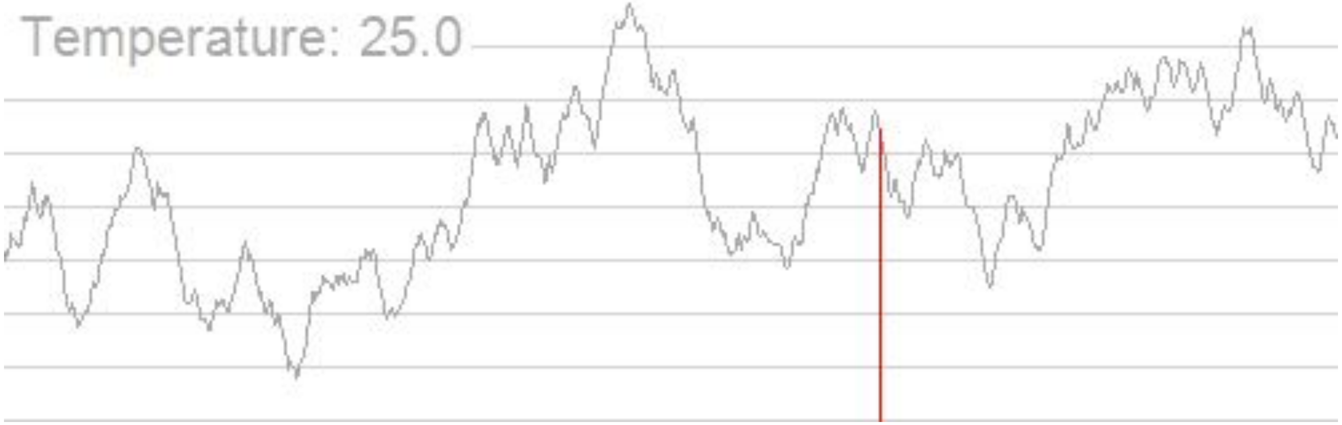


Example: simulated annealing

1. Initialize the system configuration.
   Randomize $\boldsymbol{x}(0)$.
2. Initialize T with a large value.
3. **Repeat**:
   a. **Repeat**:
      i. Apply random perturbations to the state $\boldsymbol{x} = \boldsymbol{x} + \Delta\boldsymbol{x}$.
      ii. Evaluate $\Delta E(\boldsymbol{x}) = E(\boldsymbol{x} + \Delta\boldsymbol{x}) - E(\boldsymbol{x})$:
      if $\Delta E(\boldsymbol{x}) < 0$, keep the new state;
      otherwise, accept the new state with probability $P = e^{-\frac{\Delta E}{T}}$.

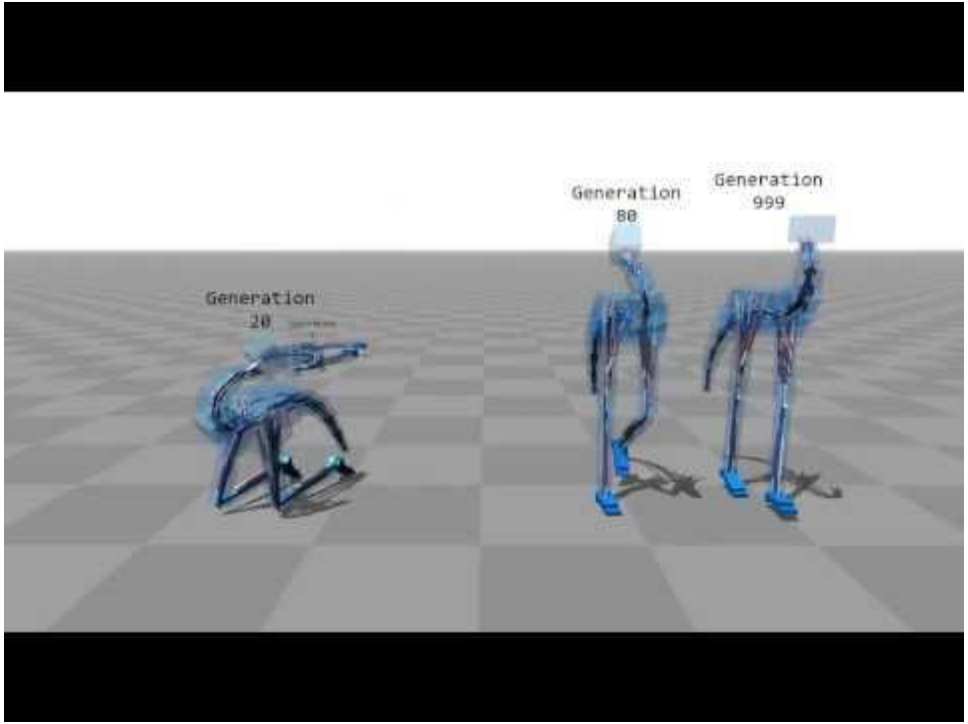
   until the number of accepted transitions is below a threshold level.
   b. Set $T = T - \Delta T$.
   until T is small enough.

Example: simulated annealing



Example: shape/brain of the robot

Example: evolve controller for robots



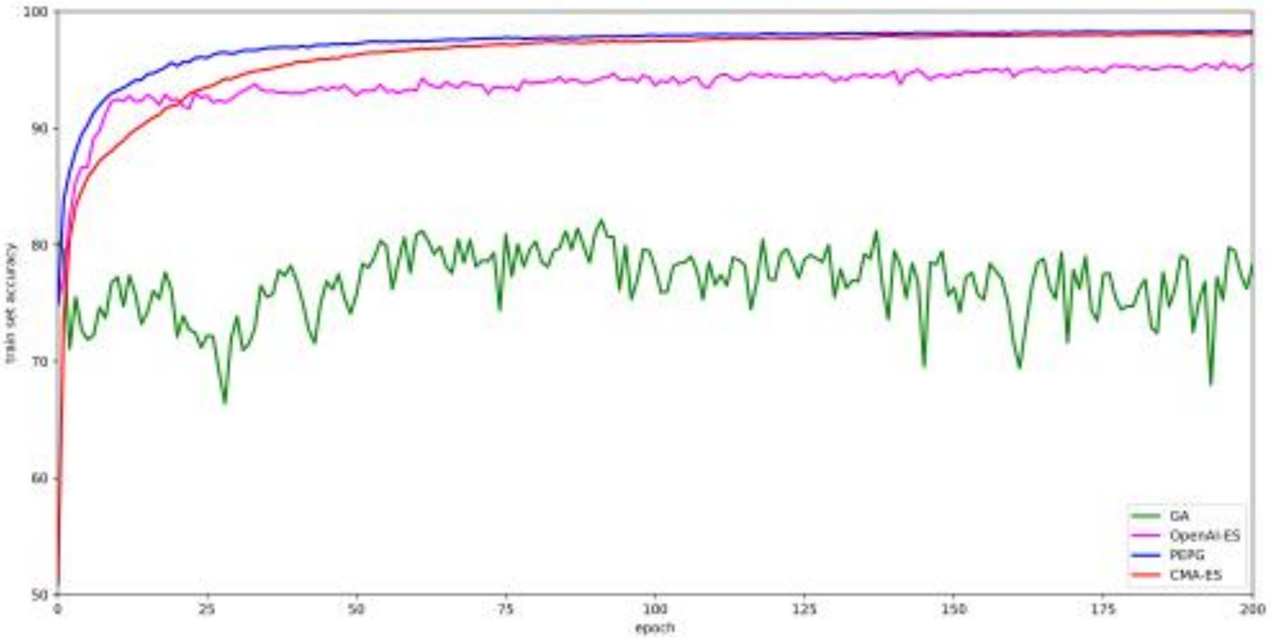
Example: soccer team

MNIST classifier

- 2 layer ConvNet
- ~11K parameters
- For EA algorithms
  - Population 101
  - Generations 300

Method	Train Set	Test Set
Adam (BackProp) Baseline	99.8	98.9
Simple GA	82.1	82.4
CMA-ES	98.4	98.1
OpenAI-ES	96.0	96.2
PEPG	98.5	98.0

MNIST classifier



Record in Atari games - simple genetic algorithm

Deep Neuroevolution: Genetic Algorithms are a Competitive Alternative for Training Deep Neural
Networks for Reinforcement Learning

- Super simple
- State of the art performance in Many Atari games

Algorithm 1 Simple Genetic Algorithm

Input: mutation power σ, population size N, number of selected individuals T, policy initialization routine ϕ.
for $g = 1, 2...G$ generations **do**
  for $i = 1, ..., N$ in next generation's population **do**
    if $g = 1$ **then**
      $\mathcal{P}_i^g = \phi(\mathcal{N}(0, I))$ {initialize random DNN}
      $F_i^g = F(\mathcal{P}_i^g)$ {assess its fitness}
    else
      if $i = 1$ **then**
        $\mathcal{P}_i^g = \mathcal{P}_i^{g-1}; F_i^g = F_i^{g-1}$ {copy the elite}
      else
        $k = \text{uniformRandom}(1, T)$ {select parent}
        Sample $\epsilon \sim \mathcal{N}(0, I)$
        $\mathcal{P}_i^g = \mathcal{P}_k^{g-1} + \sigma\epsilon$ {mutate parent}
        $F_i^g = F(\mathcal{P}_i^g)$ {assess its fitness}
  Sort $\mathcal{P}^g$ and F^g with descending order by F^g
Return: highest performing policy, $\mathcal{P}_1^g$

Why to use EA?

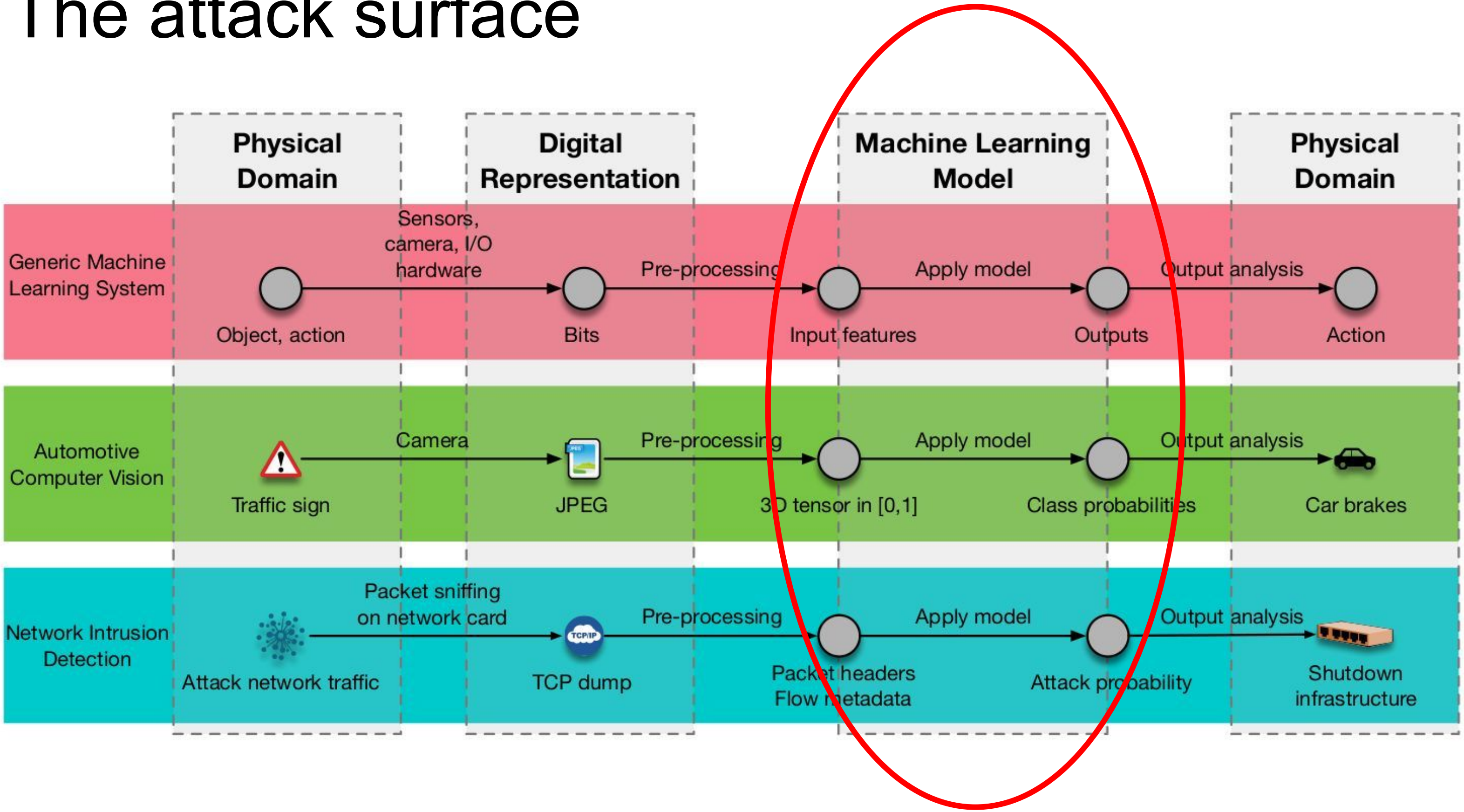
- Fast prototyping
  - In NN for example: No need to think about differentiability
- Hard to describe the evaluation function
  - Relationship between speed and shape of robot

Adversarial examples

Adversarial examples

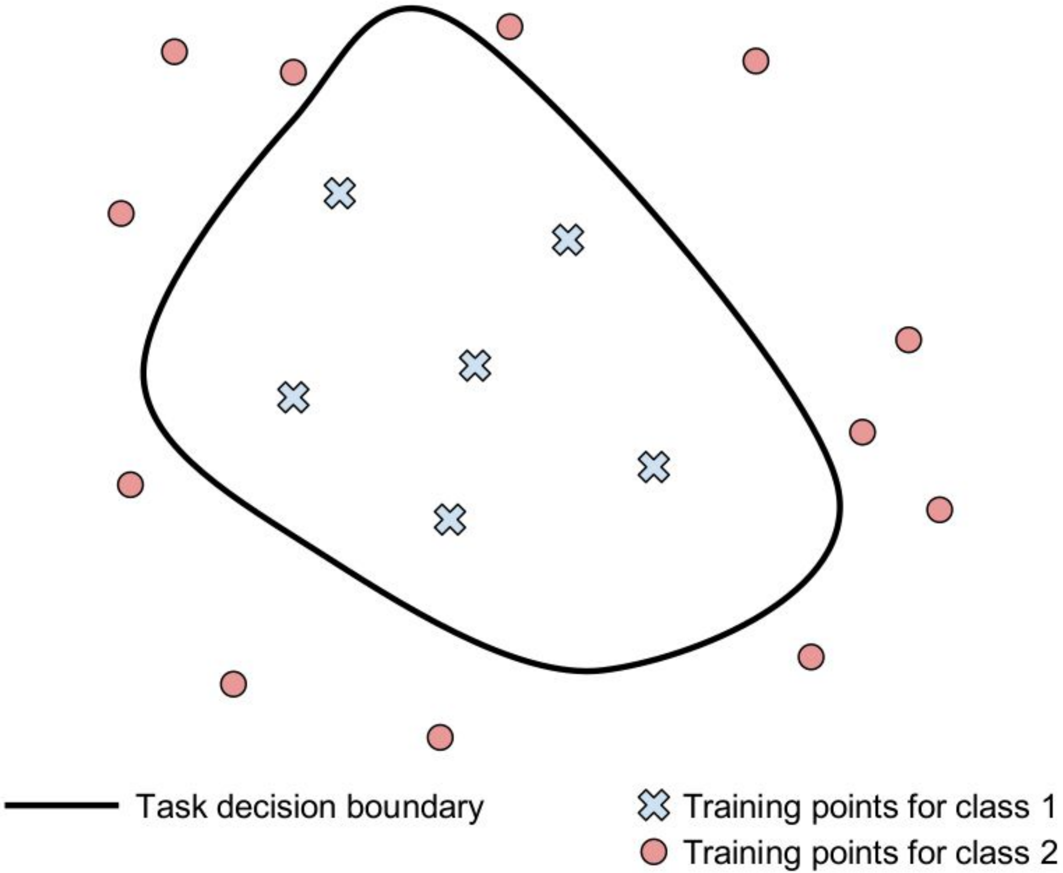
- Definition:
    - Data examples, that exploit the model limited knowledge about reality
- Important:
    - Compromise the integrity of the predictions (wrt expected outcome)
        - Can we trust the model results?
    - Compromise the availability of the system deploying machine learning
        - Effect of real-life data on the model

The attack surface



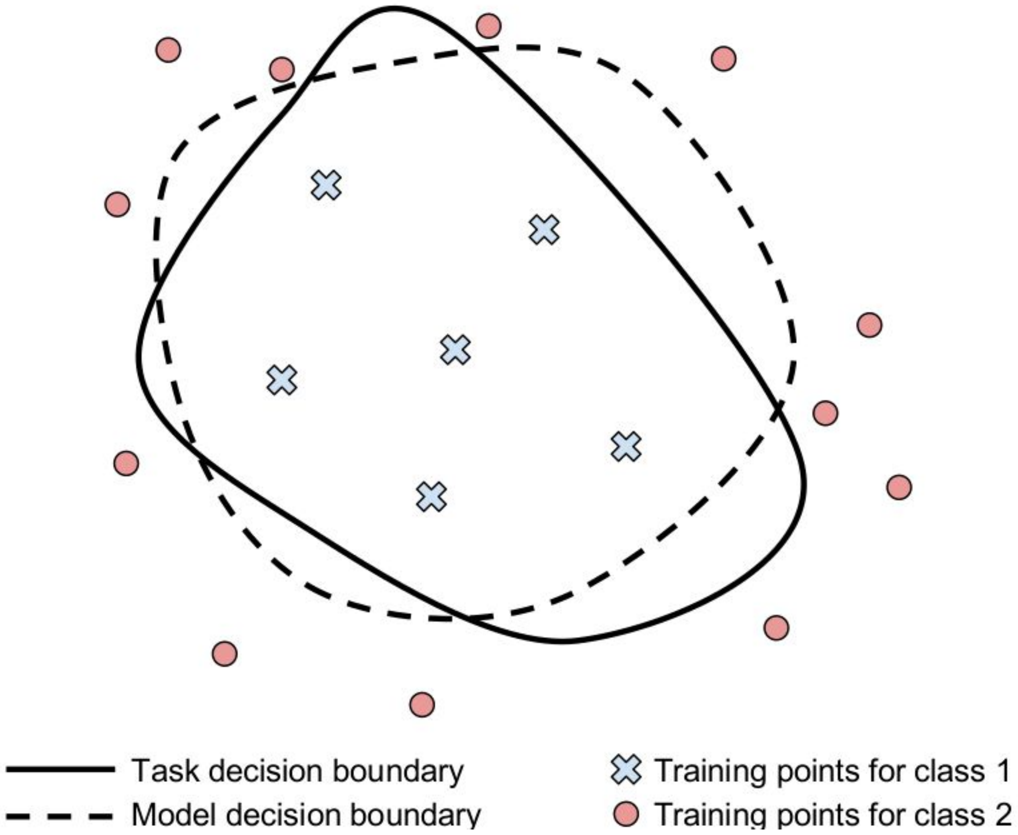
"Security and Privacy in Machine Learning", Nicolas Papernot, Google Brain

Generalization error



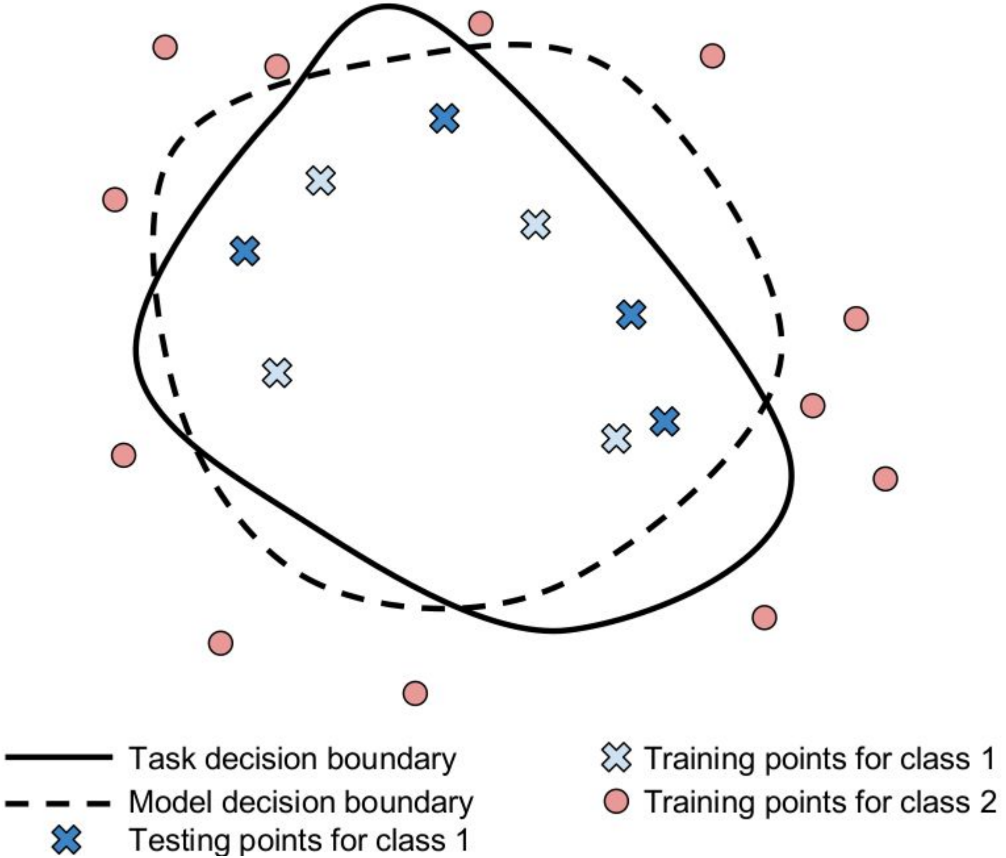
"Security and Privacy in Machine Learning", Nicolas Papernot, Google Brain

Generalization error



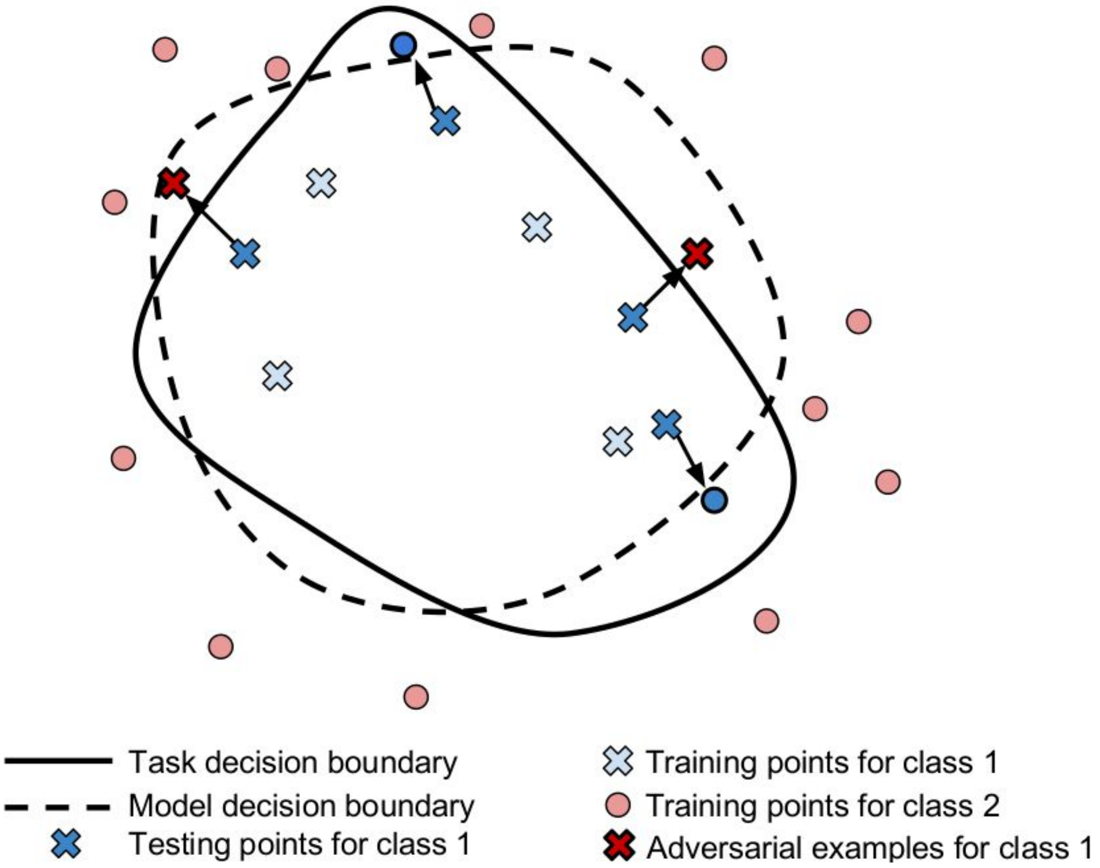
"Security and Privacy in Machine Learning", Nicolas Papernot, Google Brain

Generalization error



"Security and Privacy in Machine Learning", Nicolas Papernot, Google Brain

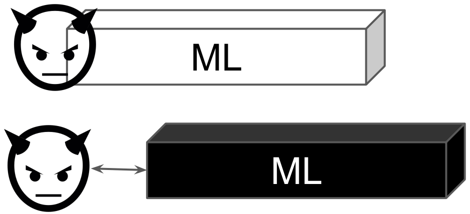
Generalization error



"Security and Privacy in Machine Learning", Nicolas Papernot, Google Brain

Types of adversarial examples

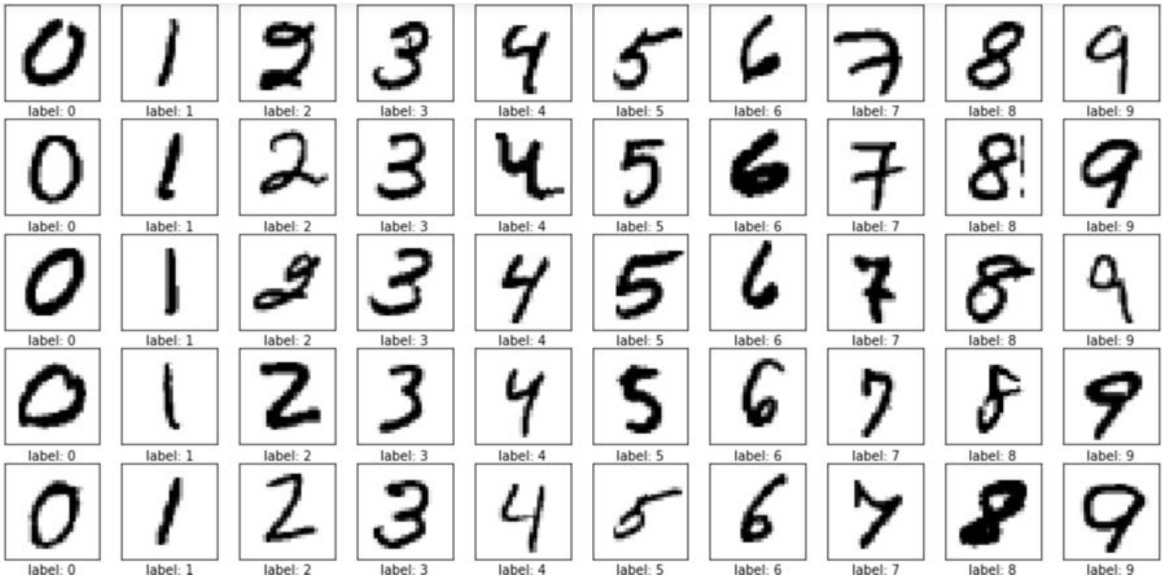
- Training time
    - Ex: Training in the presence of malicious errors
- Inference time
    - White-box attack (model inspection)
    - Black-box attack (model query)



"Security and Privacy in Machine Learning", Nicolas Papernot, Google Brain

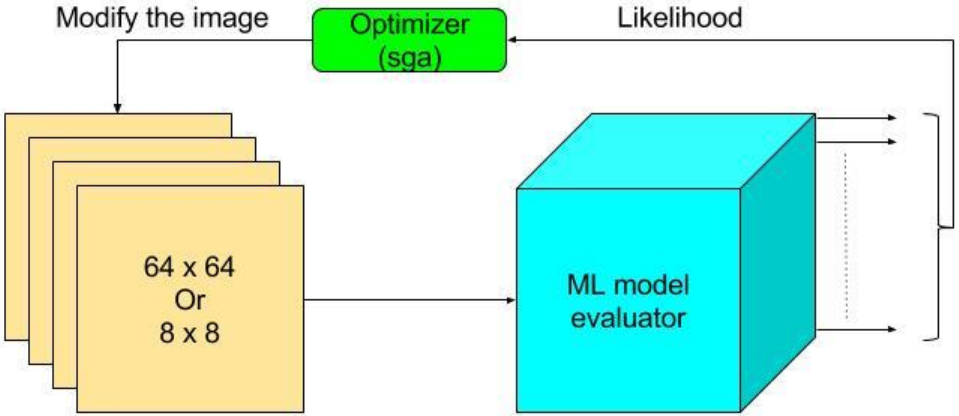
Experiment setup

1. Train a model on an MNIST data set (the evaluator)
   60K training data
   10K test data
   64 x 64 images
   a. For kNN, I use a subsample
      1617 training data
      180 test data
      8 x 8 images



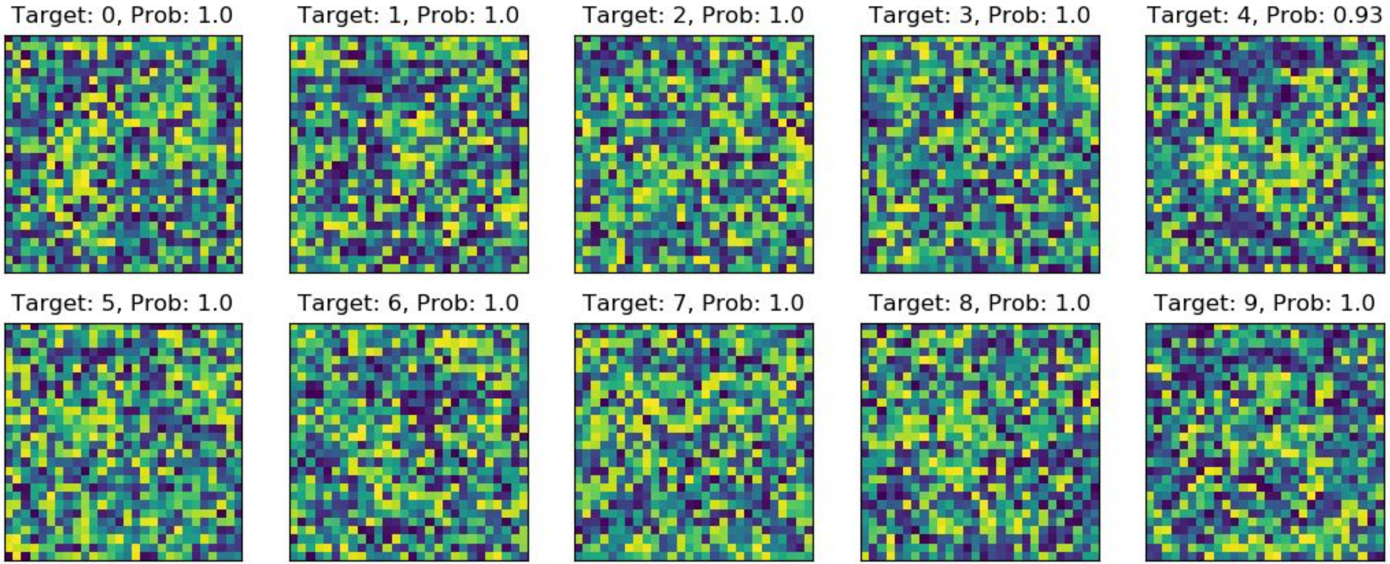
Experiment setup

2. Query the model (black box optimization)



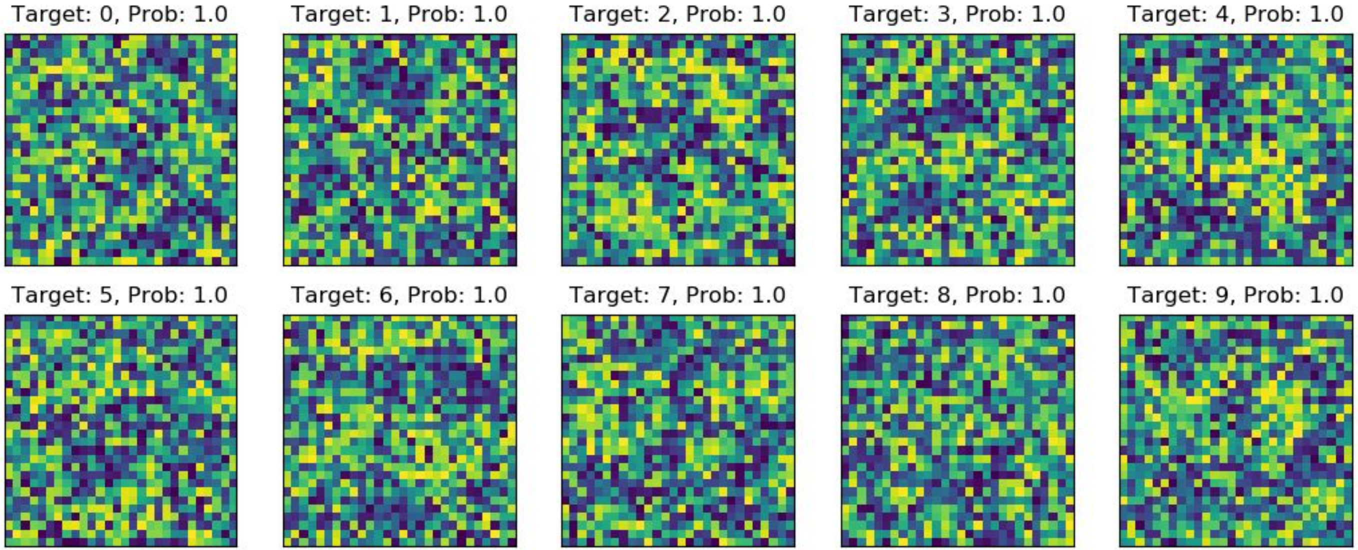
Logistic regression

- Train acc: 92.7%
- Test acc: 92%



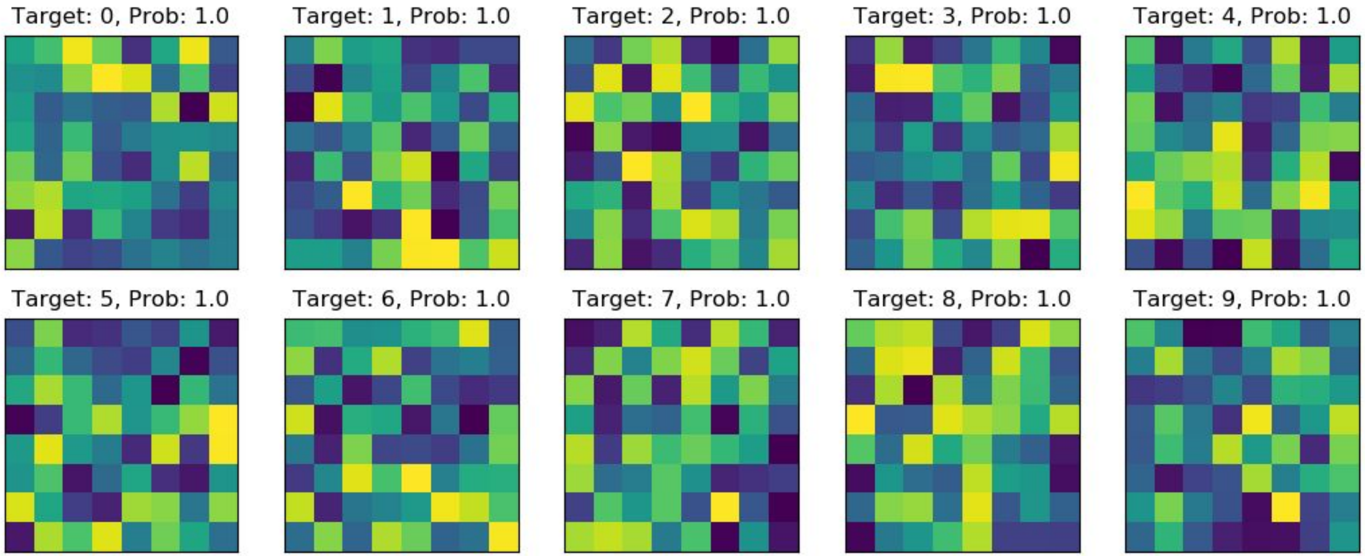
Multilayer perceptron

- 1 hidden layer, 50 neurons
- Train accuracy 98.57 %
- Test accuracy 97.1 %



kNN - 5 neighbors

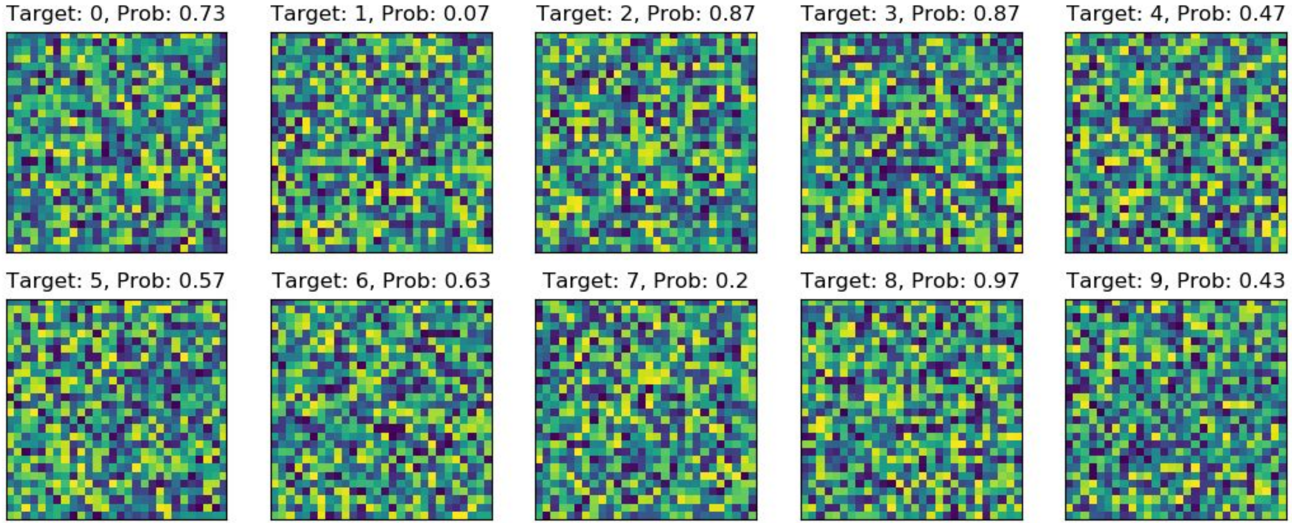
- Train acc: 99 %
- Test acc: 96.1 %



Random Forest - 30 estimator - <u>Same time</u>

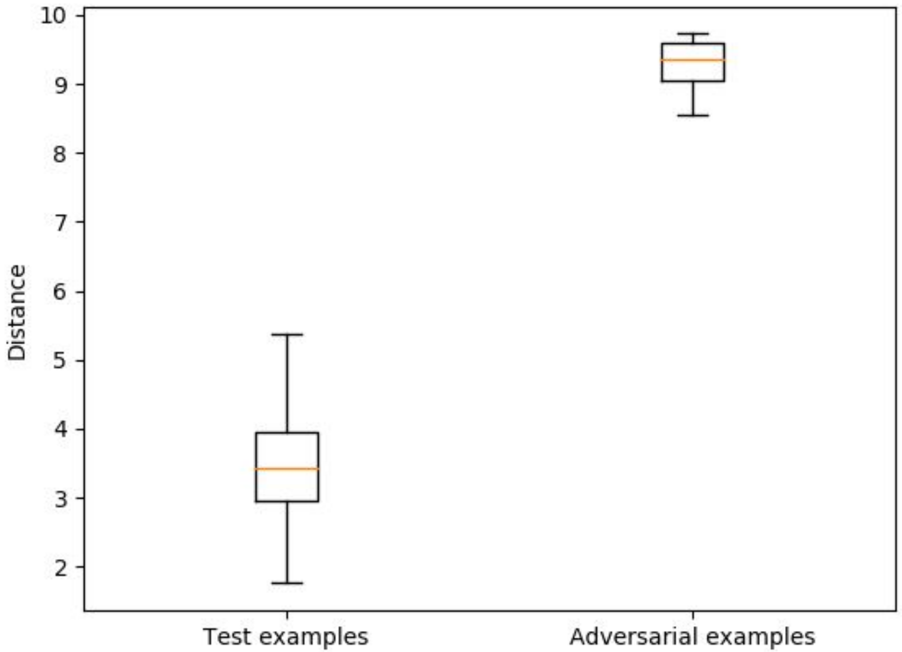
- Train acc: 99 %
- Test acc: 96.57%

Harder to break… But breakable



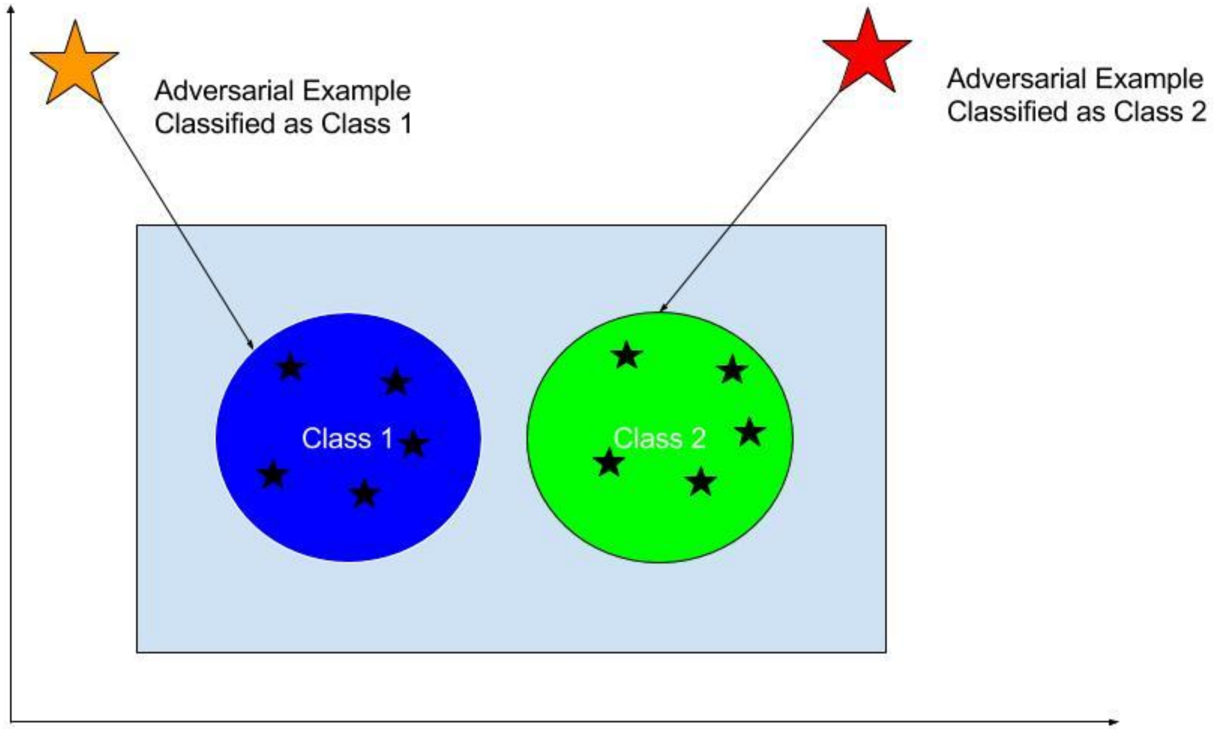
Analysis on kNN performance

- Adversarial examples can be Very far from test examples
- Possible extra regularization



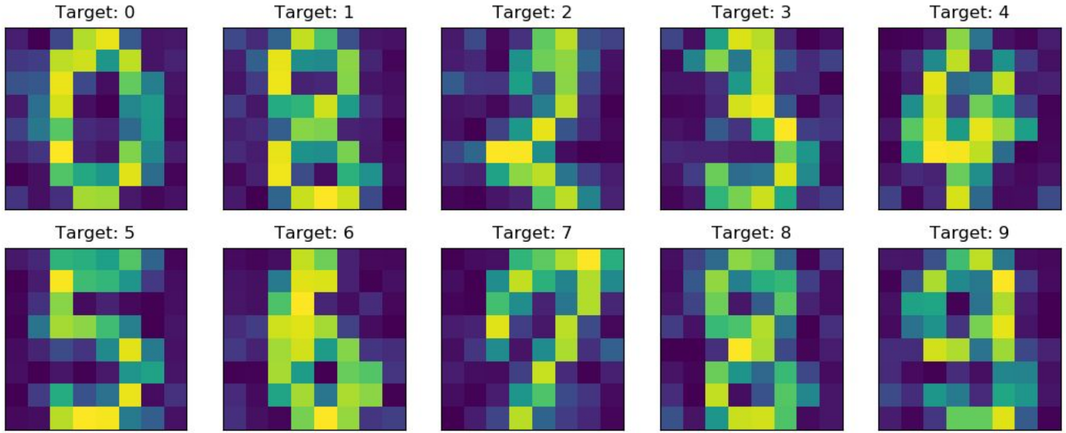
Analysis on kNN performance

- What if we add an extra objective? (Regularization)
  - Max likelihood
  - Min the distance to nearest neighbors

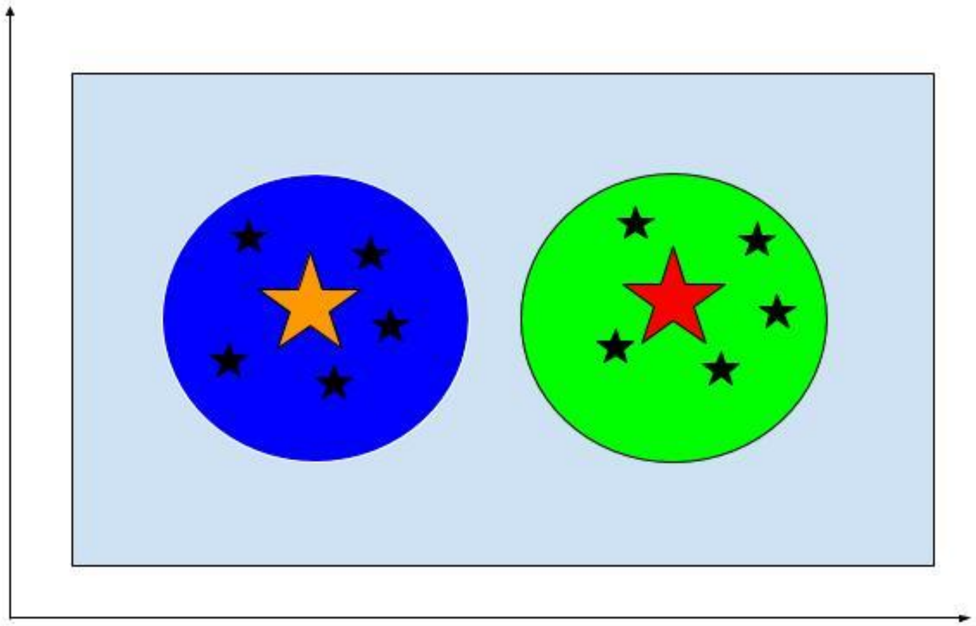


kNN: likelihood + distance to nearest neighbor

- Letters starting to appear
- Mount to an averaging Problem
- Target 1 is interesting!
    - Started from the position Closer to 8
    - A bad/unclear objective Can ruin the example



Visualization for the kNN with the new loss function



Resources

- http://blog.otoro.net/2017/10/29/visual-evolution-strategies/
- PyGMO optimization framework https://esa.github.io/pagmo2/index.html
    - Nice example https://esa.github.io/pagmo2/docs/python/tutorials/coding_udp_simple.html
- https://www.tutorialspoint.com/genetic_algorithms/index.htm