UNIVERSITY of HOUSTON
DEPARTMENT OF COMPUTER SCIENCE

# Ensemble and deep learning

Rakesh Verma

# Ensemble learning

Key Idea: Instead of using one learner, use multiple learners and combine their decisions, e.g., majority vote

Many algorithms for ensemble learning:

- Boosting
- Bagging
- Stacking
- Available in Weka and other machine learning/statistical software

# Boosting

Key Idea: Take an ensemble (group) of weak learners and combine them into a single strong learner

Weak learner – accuracy is 0.5 + epsilon, where epsilon > 0 and small

For example: A short decision tree

Strong learner – Much higher accuracy

For example: Typically SVM

# AdaBoost

Example: 3 weak learners on a dataset of n-grams from malware/hamware

Purpose: Predict the class of given software, given some training data

Method: Several rounds of training, say 100

Round 1:

- Take a random sample of the dataset and test accuracy of each learner. Output best learner.
- Increase weight of the misclassified samples, so they have higher chance of getting picked in next round.
- Assign weight to best learner, weight is based on accuracy and add to group

# AdaBoost (contd)

Round 2 through 100:

- Repeat the same procedure, i.e., find the best learner on a new sample (higher chance of getting misclassified samples from previous round), adjust weights, combine the learners …
- AdaBoost is in Weka, scikit-learn, etc.

**Input:** Data set $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$;

Base learning algorithm $\mathcal{L}$;

Number of learning rounds $T$.

**Process:**

$D_1(i) = 1/m.$ % Initialize the weight distribution

for $t = 1, \cdots, T$:

$h_t = \mathcal{L}(\mathcal{D}, D_t)$; % Train a weak learner $h_t$ from $\mathcal{D}$ using distribution $D_t$

$\epsilon_t = \Pr_{i \sim D_i}[h_t(\boldsymbol{x}_i \neq y_i)]$; % Measure the error of $h_t$

$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$; % Determine the weight of $h_t$

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(\boldsymbol{x}_i) = y_i \\ \exp(\alpha_t) & \text{if } h_t(\boldsymbol{x}_i) \neq y_i \end{cases}$$

$$= \frac{D_t(i) \exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{Z_t}$$ % Update the distribution, where $Z_t$ is

% a normalization factor which enables $D_{t+1}$ be a distribution

end.

**Output:** $H(\boldsymbol{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$

# Deep Learning

- Neural networks with many hidden layers were languishing until
- With huge gains in computing power, G. Hinton worked on an image classification problem and got near human performance
- Many different configurations: CNN (process grid-like data), RNN (sequence data), LSTM, Autoencoders, etc., and exciting progress …

# However …

Adversarial machine learning shows that …

It is not clear what these networks are learning!

Early attacks were on SpamBayes (dictionary and targeted attacks)

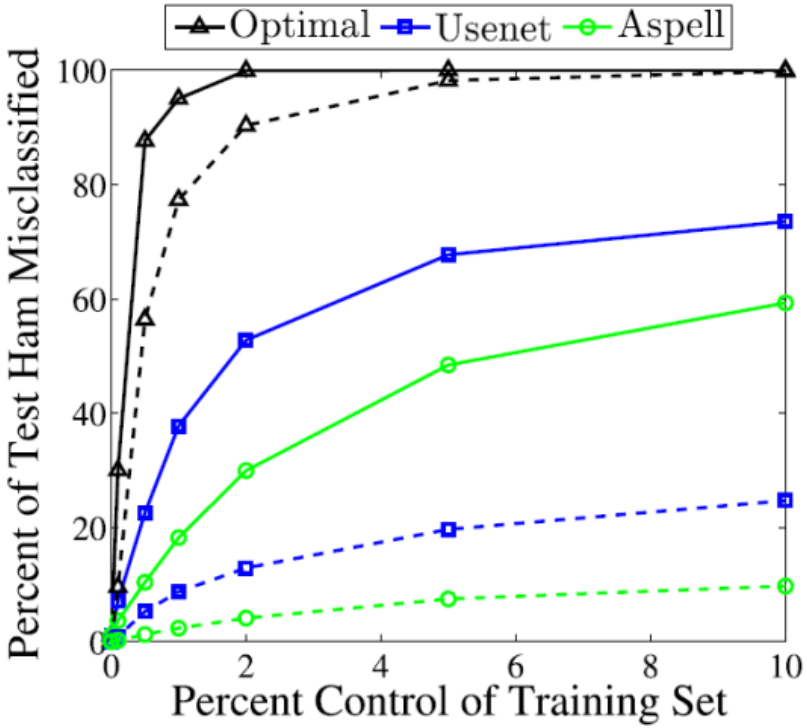
Recent attacks on Deep Learning models

Example: Take a picture and manipulate a few pixels to get misclassification

# Attacks on SpamBayes

- Dictionary attack
- Target attacks

[A taxonomy of attacks and more examples at:
The Security of Machine Learning by Barreno et al.]



# Transferability Attacks on deep learning models

Suppose target model has unknown:

- Weights
- machine learning algorithm
- training sets

Maybe even non-differentiable!

Substitute model “mimicking” target model with known and differentiable function

Generate attacks for this model and deploy against target – high probability of success

# Practical Attacks Against

Real classifiers trained by remotely hosted API (Google, Amazon, etc.)

Malware detectors and networks

Machine learning models that are back-ends of cameras in autonomous vehicles

# Failed Defenses

- Removing perturbation using autoencoders (a technique for reducing noise)
- Adding noise at training time (hoping it will generalize better)
- Dropout (a technique for generalization)
- Ensembles
- Double backpropagation
- Non-linearity

…

# Solution

Train with adversarial examples

Neural networks have an advantage in this space compared to:

SVMs, linear regression, k-NN