Monthly Meeting - February

ECE Capstone [Group 14]
February 22, 2019

History of Back propagation

- 1960s
  - Engineers began a discussion of backpropagation.

- In 1970
  - Linnainmaaz first mentioned backpropagation in his master's thesis, and was soon explicitly used to minimize the cost function by adjusting control parameters (weights).

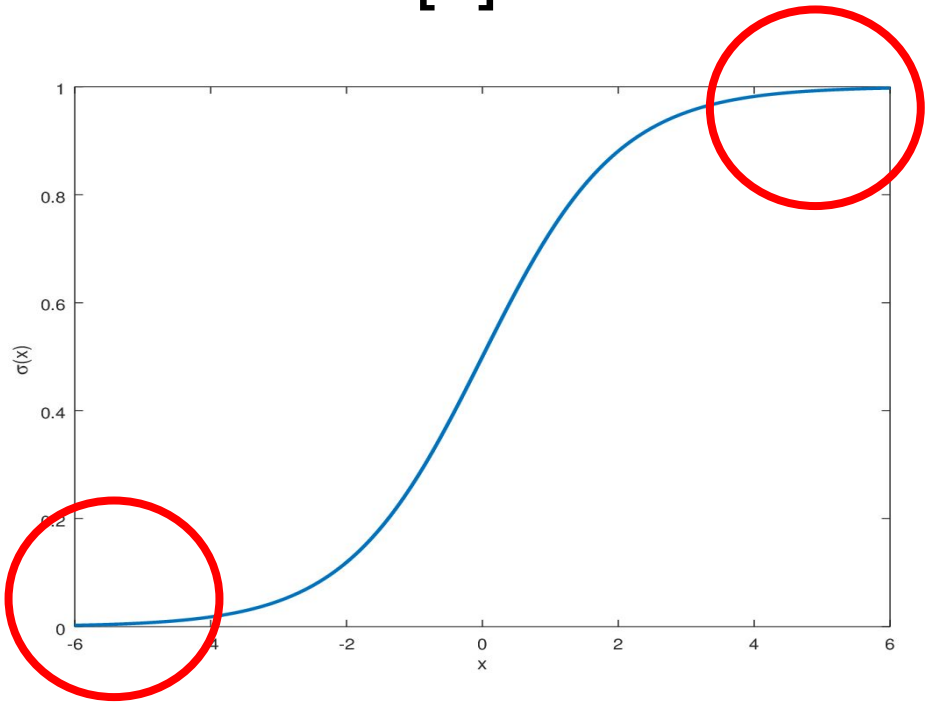
- In 1982
  - Werbos first described the above-mentioned neural network specific applications for efficient BP

- In 1986
  - The paper by Rumelhart showed the emergence of useful internal representations in hidden layers, significantly contributing to the popularity of BP in neural networks.

- So far
  - Backpropagation has been widely used in deep learning, making deep learning develop rapidly.

Trial 1 - Error Saturation Prevention[1]

- **Premature Saturation**: When an output node is pushed to an extreme value at the edges of the sigmoidal activation function.
- Small derivative means hardly any learning (weight updating) occurs



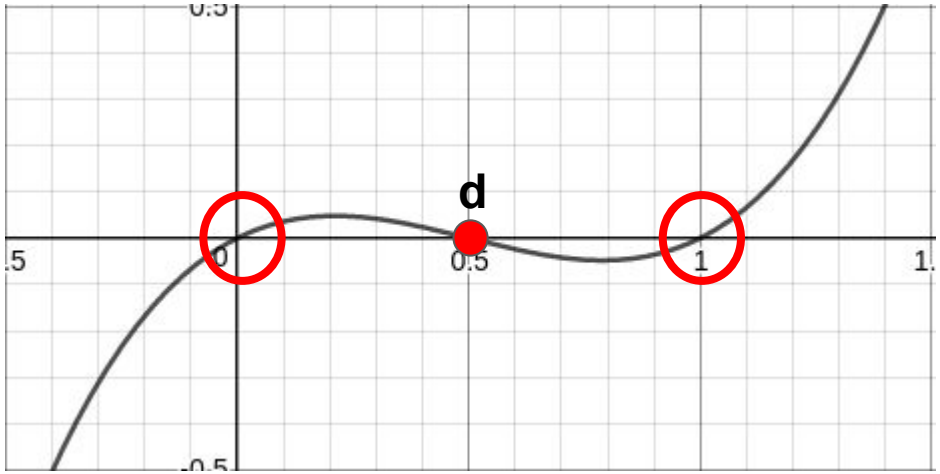
Hahn-Ming Lee, Chih-Ming Chen, Tzong-Ching Huang,
Learning efficiency improvement of back-propagation algorithm by error saturation prevention method,
Neurocomputing, Volume 41, Issues 1–4, 2001, Pages 125-143, ISSN 0925-2312,
https://doi.org/10.1016/S0925-2312(00)00352-0.

- Learning Term (for output layer):

  ξ = (d - o)*o*(1-o)

  Where ‘d’ is desired output and ‘o’ is the actual output.

- If d = -0.5, and o = 0 or 1, almost no learning occurs



d = 0.5.
X-axis = ‘o’
y-axis = ‘ξ’
Problematic areas in red circles

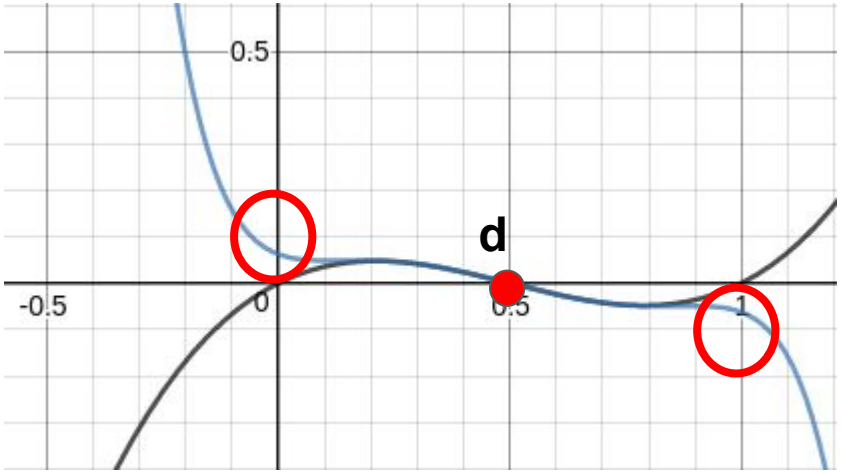
- Add a term that corrects for this issue:

  $y = \alpha(o-0.5)^n$

- Now the learning term is corrected for this situation:

  $\xi = (d - o)(o(1-o) + \alpha(o-0.5)^n)$

- Learning term is small when d ~ o which is desired



Old learning term in black, corrected in blue.
d = 0.5
X-axis = ‘o’
y-axis = ‘ξ’
Problematic areas in red circles

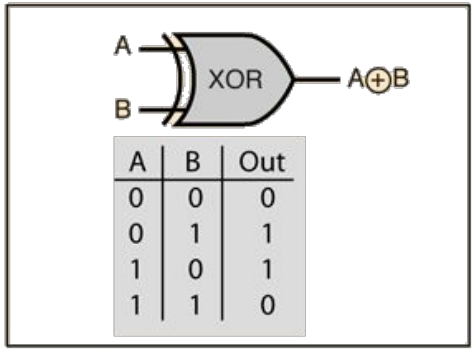
Moving Forward

- Goal is to implement this solution by myself (Mark) for a few weeks and see if it has promise.
- Will use Jian’s XOR network.
- Results will be reported on at the next sponsor meeting.

XOR Classification

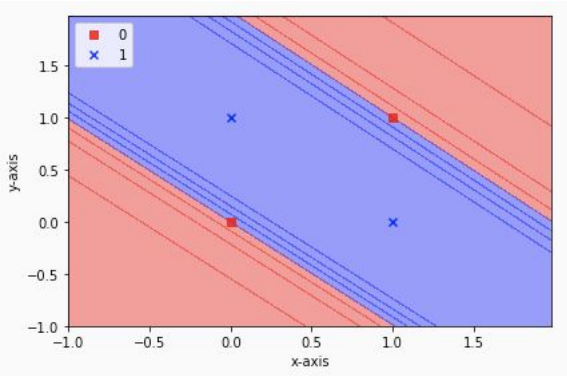
The input set of XOR gate can be considered as the simplest nonlinear classification problem. It can be considered as a good example for ANN classification:

1. Table 1 shows the basic I/O relationship of a XOR gate.
2. By adding gaussian noise, the scatter points can be used to represented the decision region



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

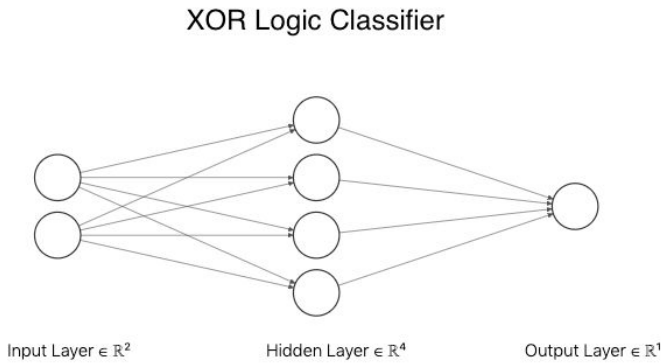
Table 1: XOR Logic



Python Implementation: Current Progress

By using numpy library, a two layer neural was constructed.

1. The user can change the number of neurons for each layer.
2. Current structure:
   a. 2 input features
   b. 1 hidden layer with 4 neurons
   c. 1 output features
3. Data sets:
   a. Training sets: 80%
   b. Test sets: 20%
4. Accuracy: 100.0 %

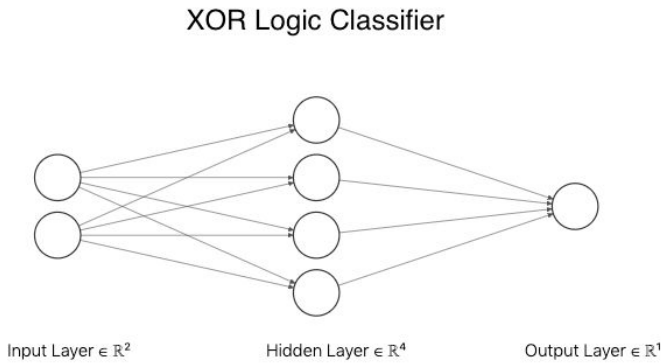


(GitHub code)

Python Implementation: Next Step

Since we are considering how the noise signal can affect the BP algorithm, and right now we've already have a successful XOR classifier, next step should consider the addition of the noise.

1. What type of the noise we are considering? (Gaussian?)
2. In the code, I print out the weights of the hidden layer step by step, to discover the effect of the noise, maybe we can observe the weights changes by the noise level.



Perspectives - Next part

- I (Yuhao) will discuss the current network training algorithms,
- Analyzing the problems in the algorithm,
- Raising some possible methods of improvement

General Training Algorithm

We can split the training algorithm of fully connected network into 4 steps:

- Forward calculation (for activation functions, and the forward network flow)
- Loss calculation (i.e. applying loss function)
- Backward calculation (for gradient of each layer)
- Weight Update (or, optimization, e.g. SGD)

No improvement on the **Forward Calculation** and **Loss Calculation** steps. So we can focus on the last two steps: the optimization problem.

As we know, the goal of machine learning algorithm is solving the optimization problems.

Optimization: Gradient Free

To avoid expensive auto gradient computation, Derivative-Free Optimization (DFO) is a good way.

When to use:

- Discontinuous Function
- Gradient is hard to compute

Example:

- Facebook's open source tool for DFO (GitHub)

Optimization: Gradient Free

Disadvantages:

- Performance is lower than Derivative based ones in high dimensional problems.

Therefore, DFO is only suitable in hyperparameters tuning. It might not the best choice of building neural network.

Reference

- Rios L M, Sahinidis N V. Derivative-free optimization: a review of algorithms and comparison of software implementations[J]. Journal of Global Optimization, 2013, 56(3): 1247-1293.

- https://stats.stackexchange.com/questions/207450/in-neural-nets-why-use-gradient-methods-rather-than-other-metaheuristics

Local Minima vs. Global Minima

In paper “Anna Chromanska et al., The Loss Surface of Multilayer Networks”, authors mentioned that:

- For large-size networks, most local minima are equivalent and yield similar performance on a test set.
-
- The probability of finding a "bad" (high value) local minimum is non-zero for small-size networks and decreases quickly with networks size.
-
- Struggling to find the global minimum on the training set (as opposed to one of the many good local ones) is not useful in practice and may lead to overfitting

Local Minima vs. Global Minima

- So, the global minima is not the preferable choice in practice. It is not necessary to use a global optimal solver in our project.

Some useful discussions about the DFO and local minimas:

- https://stats.stackexchange.com/questions/207450/in-neural-nets-why-use-gradient-methods-rather-than-other-metaheuristics
- https://stats.stackexchange.com/questions/235862/is-it-possible-to-train-a-neural-network-without-backpropagation

Optimization - Weight Update

- As we discussed, the non-gradient methods are not the best choice in neural network. Then we can focus on the fourth step - **Weight Update Algorithm**, in other words, some gradient based algorithm.

Optimization - Popular Ones

The optimizers in popular libraries (TensorFlow, PyTorch):

- Adadelta: Adaptive Learning Rate
- Adagrad: Adaptive Subgradient
- Adam, Sparse Adam, Adamax: Adam based
- SGD, ASGD
- LBFGS
- RMSprop
- Rprop
- Momentum
- Ftrl

Optimization - Popular Ones

- Live Visualizations: link
- More Visualizations: GitHub

From the Visualizations, we can find one orientation of the algorithm improvement, which is also one of the biggest problem in current optimization algorithms:

- Saddle point in high dimensional space.