BACKPROPAGATION

BACKWARD PROPAGATION OF ERRORS TO TRAIN ARTIFICIAL NEURAL NETWORKS

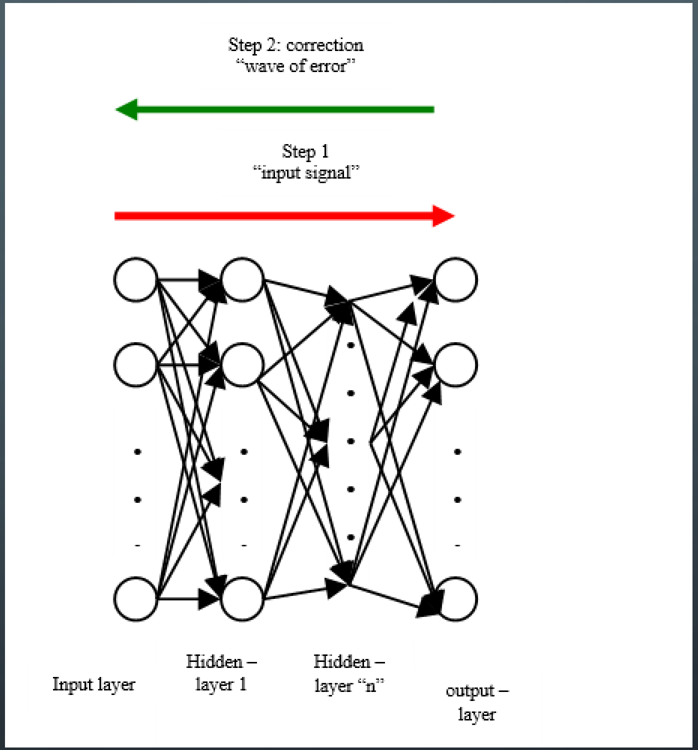
L665 APPLYING MACHINE LEARNING TECHNIQUES IN COMPUTATIONAL LINGUISTICS

SCOTT MCCAULAY FEBRUARY 27, 2018

WHAT IS BACKPROPAGATION?

- Most common method for training neural networks today
- Iterative process, each step starts with errors observed at the output layer, and adjusts the weights of each edge contributing to the errors, working backward through the network. Hence “backward propagation of errors”
- Tries to find an ideal combination of weights for test data, treats the weight calculation as an optimization problem using gradient descent
- Concepts of backpropagation date back to the 1960s in other fields, application to training Neural Networks was introduced by Rumelhart et al. in 1986

OVERVIEW OF THE PROCESS



STEP BY STEP

- Propagation
  - Plug in input values, calculate all hidden and output values by layer using weights, bias and activation functions
  - Calculate loss function based on errors at output layer (sum errors for all training data)
  - Calculate a delta value for each output and hidden node (larger error -> larger delta)
- Update Weights
  - Find a gradient value for each edge's weight based on the delta value of its output node and the activation value of its input node
  - Adjust gradient based on learning rate, adjusted value is applied to weight
- Repeat until convergence, or until assigned iteration count is reached

WHAT ABOUT BIAS?

- The bias value at each node is treated as another input, with a value always equal to 1
- This bias input to a node has a weight assigned to it. So to the backpropagation algorithm, bias optimization is simply part of weight optimization.

AVOIDING OVERFITTING

- Training can be too specific to the training data, and not generalize well when new data is introduced
- Overfitting has become more of a challenge as network size and capability have grown, since a complex network may represent test data very well
- Techniques are commonly used with backpropagation to minimize overfit
  - Regularization – add a complexity penalty to error calculation, based on large weights
  - Dropout – randomly set some nodes to zero, discourage dependence on specific paths

AN IMPLEMENTATION IN PYTHON

- scikit-learn is a Python library of machine learning and data analysis tools
- MLPClassifier (Multi-layer Perceptron classifier) is a neural network algorithm using backpropagation for training
- Some of the parameters passed to MLPClassifier are related to backprop
    - solver : {‘lbfgs’, ‘sgd’, ‘adam’} – choice of algorithms for weight optimization
    - learning_rate : {‘constant’, ‘invscaling’, ‘adaptive’} – weight adjustments over time

BACKPROPAGATION AND GPU COMPUTING

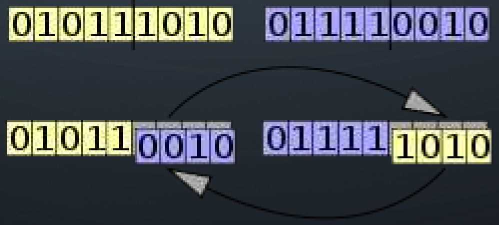
- Collaboration in the late 2000s between Google, Stanford and NVIDIA led to big improvements in Neural Network scaling and speed
  - Google Brain's neural network used 16,000 CPU cores. Comparable performance was achieved at Stanford using 12 GPUs
- GPUs are very well suited to the types of calculations used in backpropagation
- Fields that had dismissed Neural Networks years earlier began to reassess, as impressive and achievable results were demonstrated

MORE COMPLEX NETWORKS

- Backpropagation is still commonly used in neural networks with more complex structural or temporal organizations
- Recurrent neural networks
  - Backpropagation through time
  - Computationally fast, even more problems with local optima than feed forward networks
- Convolutional neural networks
  - Effective in visual recognition, but convolutional computation adds high computing requirements in both forward and backward propagation

GENETIC ALGORITHM AS A TRAINING ALTERNATIVE

- Once considered a viable or superior alternative to backpropagation
- Treat collection of weights as a genome. Evolve a population of networks, moving toward optimization through crossover and mutation among best-scoring networks
- May address problems inherent in gradient descent algorithms, but today much more computationally intensive and not in wide use


0101111010
0111110010
010110010
011111010