Deep Genetic Training

The Duality

Artificial Intelligence & Genetic Algorithms
Neural Network
Cost Function

- \( C(W, B, I, E) \) – Function of Weights, Biases, Input data and Expected output
- Different cost metrics, e.g. how far away from expected output a the output \( y(x) \) is using e.g. MSE ...

\[
C_{MSE}(w, b) = \frac{1}{2n} \sum_{n} |y(x) - a|^2
\]

- Or Cross Entropy \( C_{CE} = ... \)
- Or exponential cost \( C_{EXP} = ... \)

Approximation of a gradient

\[
\Delta C \approx \Delta w \frac{\partial C}{\partial w} + \Delta b \frac{\partial C}{\partial b}
\]
The Cost Function Landscape

• My Intuitive Analogy
  – Non linear multi dimensional landscape
  – Rocky mountains and deep valleys or large flat deserts
  – Steeps we can not cross and that we get stuck in
  – Somewhere there is a lowest point that we search
  – We have no global map. Just local sight
Back propagation...

- Back propagation is the standard method in NN training
  - Minimize a cost function
  - Gradient descent is “walking” down the hill
  - Stochastic sampling in combination with average vector direction is our local direction
  - Momentum and other methods to preserve speed and direction in tricky situations

- Identified strengths
  - Linear complexity $O(N)$ compared to Newton-Rapson or General Descent $O(N^?)$ (fast)
  - Can use different Cost metrics like Cross Entropy (stable)
  - Regularization (stable)
  - Tensor math based for acceleration (superfast)

- Identified problems
  - Start values of parameters (bias and weights)
  - Backprop based on pushing error adjustments from end of network
  - Error updates work like an energy pulse that drops the further we get into the net backwards
  - Gradient descent has problems with “saddle points”, “flat” cost function or with large variance in input
  - Difficult for deep networks
  - Difficult for more complex networks like true recursive or “any direction” networks
  - Closed loop, Must wait for all calculated gradients before update
Genetic programming

• Programming a system inspired by biological evolution

  – *Biological evolution base principle*
    The strongest and fittest survives.
    Good genomes are inherited. Good genome combinations between properties and between individuals.
    Adaptation to surroundings and reality.
    Mutations and “bad” genomes can combine to better genomes.

  – *Individuals, Populations and Generations*
    Not just one individual but many. Tribes will survive. Unique fitness will be kept. Time and generations will evolve genomes. True evolution.

  – *DNA as code*
    Describe software execution as a DNA (weights, parameters and mnemonics)

  – *Neural network to execute DNA*
    Different topologies in network is also based on DNA.
    Recursive and highly nonlinear connections are possible.

  – *HW executing genomes*
    Compiled network combined with DNA software codes.
    Executes packages of DNA
Deep Genetic Networks

- Based on NN principles
  - **Evaluation**
    Same functions to evaluate function as trad. NN. Same Cost function, same parameters. Equal to a NN
  - **Training using genetic principles**
    Crossover, Mutations and Breeding
    Population, Generations
  - **Train both topology and parameters**
    Recursive, Recurrent, Any direction
  - **HW accelerated**
    Code to accelerate execution.
  - **Non closed Training**
    Parallel on any number of computers
Genetic Principles

• What mathematical properties do we see
  – **Mutations**, A random stochastic normal distributed change in any part of the network.
    • The solution can jump any distance as the change is normal distributed but will have a mean and stddev behavior.
    • The change can occur anywhere in the network. The most inner layer can be affected in the first iteration.
  – **Crossover**, A combination between a mother solution and a father solution. In some cases you get good genes and in some cases you get bad. However you might get both good genes from both which makes you better
    • A new solution can be generated in a convex hull between two parent solutions
    • A new solution can jump a large step if parents are located far away and a very short step if they are located near each other.
  – **Breeding**, A combination I have found out. Its not either mother solution or father solution parameters but a linear continuous interpolation/extrapolation between two solutions along a multi dimensional vector that is normal distributed
    • A new solution can continue in a direction which might be same as the gradient descent.
    • A new solution can jump over saddle points
    • Not just perturbed random values but a higher likelihood that it continues in a direction
Genetic Training

• Population
  – A number of solutions
  – Survival of the fittest. Only The best survive to the next generation
  – Typically clusters of solutions around each best local minima
  – But also solutions further away

• Generations
  – Evolution in multiple generations
  – Population will evolve multiple “best local mimimas” like clusters that follows gradient descent
  – Mutations will find new minima eventually

• Distributed Societies
  – Evolution on multiple computers (simulations)
  – Exchange of the fittest DNA (best solution)
  – Linear progress scales to number of simulations
Improvements using Deep Genetic Learning

• Saddle points
  – The genetic solutions will be able to jump down into local saddle point minima where stochastic gradient decent will get stuck in gradient fluctuations

• Flat landscape
  – Genetic solutions can take any length in step. Steps are normal distributed but can have any length. Typically you see this when changes occur in the inner most layers.

• Multiple solutions
  – The genetic solver can handle multiple “good” solutions in parallel. Basically it can do multiple path gradient descents at the same time

• Scalability
  – The genetic solver can scale simulations on any number of computers which will allow a theoretical infinite speedup of training
Drawbacks using Deep Genetic Learning

• Stochastic random process
  – The genetic solver will be able to move in descent direction but not as efficient as back propagation so when the landscape is smooth downhill the genetic solver will be inefficient.

• Non predictive behavior
  – The genetic solver will sometimes be extremely fast and in some cases slower and this is bad for performance predictions
The Duality

• A combination of all good properties
  – By combining a genetic solver with a gradient descent you will be able to use both worlds of strengths
  – The genetic solver will find multiple solutions and jump to faster solutions
  – And when a new better solution is found the gradient descent can drive that solution down the hill to even better values.

• A state machine
  – By combining the Genetic Solver with the Stochastic Gradient Descent in a state machine, you can predict the best state as a function of your progress and switch state to the dual solver.
  – The flip flop mechanism will enable in total a fast gradient descent that can take large jumps and use multiple solutions in parallel scaled on multiple computers in general
The SDK
Realtime Analysis and Feedback

• Detect improvement
• Control execution
• Debug output from neurons and synapses
• Visualize progress and signals
Deep Genetic Training

The Duality

Anders Modén

anders@tooltech-software.com