Distributed Training on HPC

Presented By: Aaron D. Saxton, PhD
Statistics Review

- Simple $y = m \cdot x + b$ regression
  - Least Squares to find $m,b$
  - With data set $\{(x_i, y_i)\}_{i=1}^{n}$
  - Very special, often hard to measure $y_i$
- Let the error be
  - $R = \sum_{i=1}^{n} [(y_i - (m \cdot x_i + b)]^2$
- Minimize $R$ with respect to $m$ and $b$.
  - Simultaneously Solve
    - $R_m(m, b) = 0$
    - $R_b(m, b) = 0$
  - Linear System
- We will consider more general $y = f(x)$
  - $R_m(m, b) = 0$ and $R_b(m, b) = 0$ may not be linear
Statistics Review

- Regressions with parameterized sets of functions. e.g.
  - \( y = ax^2 + bx + c \) (quadratic)
  - \( y = \sum a_i x^i \) (polynomial)
  - \( y = Ne^{rx} \) (exponential)
  - \( y = \frac{1}{1+e^{-(a+bx)}} \) (logistic)
Statistics Review

• Polynomial model of degree ‘n’
  • “degrees of freedom” - models capacity

Gradient Decent

- Searching for minimum
- \( \nabla R = \{R_{\theta_0}, R_{\theta_2}, ..., R_{\theta_n}\} \)
- \( R(\tilde{\theta}_{t+1}) = R(\tilde{\theta}_t + \gamma \nabla R) \)
- \( \gamma \): Learning Rate
- Recall, Loss depends on data
- Expand notation,
  - \( R(\tilde{\theta}_t; \{(x_i, y_i)\}_n) \)
  - Recall \( R \) and \( \nabla R \) is a sum over \( i \)
- Intuitively, want \( R \) with ALL DATA ….. ? \( (R = \Sigma_{i=1}^{n}[(y_i - f_{\theta_i}(x_i))^2]) \)
Gradient Decent

Fictitious Loss Surface With Gradient Field
Stochastic Gradient Decent

- Recall $R$ is a sum over $i$ ($R = \sum_{i=1}^{n}[(y_i - f_{\theta_t}(x_i))^2]$)
- Single training example, $(x_i, y_i)$, Sum over only one training example
  - $\nabla R_{(x_i,y_i)} = \langle R_{\theta_0}, R_{\theta_2}, ..., R_{\theta_n} \rangle_{(x_i,y_i)}$
  - $R_{(x_i,y_i)}(\hat{\theta}_{t+1}) = R_{(x_i,y_i)}(\hat{\theta}_t + \gamma \nabla R_{(x_i,y_i)})$
  - $\gamma$: Learning Rate
  - Choose next $(x_{i+1}, y_{i+1})$, (Shuffled training set)

- SGD with mini batches
- Many training example, $(x_i, y_i)$, Sum over many training example
  - Batch Size or Mini Batch Size (This gets ambiguous with distributed training)
- SGD often outperforms traditional GD, want small batches.
Neural Networks

- Activation functions
  - Logistic
  
  \[ \sigma(x) = \frac{1}{1 + e^{-x}} \]

  \[ \sigma(x) = \begin{array}{c}
  \begin{array}{c}
  \text{Logistic}
  \\
  \sigma(x)
  \\
  \end{array}
  \\
  \end{array} \]

- ReLU (Rectified Linear Unit)
  
  \[ \sigma(x) = \begin{cases} 
  x & \text{if } x > 0 \\
  0 & \text{otherwise}
  \end{cases} \]

- Arctan
  
  \[ \sigma(x) = \tan^{-1}(x) \]

- Softmax
  
  \[ g_k(x_1, x_2, ..., x_N) = \frac{e^{x_k}}{\sum e^{x_i}} \]
Neural Networks

- Parameterized function
  - $Z_M = \sigma(\alpha_{0m} + \alpha_m X)$
  - $T_K = \beta_{0k} + \beta_k Z$
  - $f_K(X) = g_k(T)$
- Linear Transformations with pointwise evaluation of nonlinear function, $\sigma$
  - $\beta_{0i}, \beta_i, \alpha_{0m}, \alpha_m$
  - Weights to be optimized
Faux Model Example

Trainable Weights

$$\{\Theta_i : i \in [0, 1, 2, 3, 4]\}$$
Distributed Training, data distributed
Distributed Training, data distributed
Distributed Training, All Reduce Collective

Node (Worker 0)
- CPU
- GPU

Node (Worker 1)
- CPU
- GPU

Node (Worker 2)
- CPU
- GPU

Node (Worker 3)
- CPU
- GPU

All Reduce
\[ \sum \]

\[ \nabla R_{Tot} \]
\[ \nabla R_{N,0} \]
\[ \nabla R_{N,1} \]
\[ \nabla R_{N,3} \]

\[ \nabla R_{Tot} \]
\[ \nabla R_{Tot} \]
\[ \nabla R_{Tot} \]
Distributed TensorFlow: Parameter Sever/Worker Default, Bad Way on HPC

ps:0
Aggregate
Update Parameters

worker:0
Model
Loss (Cross Entropy)
Optimize (Gradient Decent)

ps:1
Aggregate
Update Parameters

worker:1
Model
Loss (Cross Entropy)
Optimize (Gradient Decent)

worker:2
Model
Loss (Cross Entropy)
Optimize (Gradient Decent)
Other models: Sequence Modeling

- Autoregression
  \[ X_t = c + \sum_{i=1}^{p} \phi_i B^i X_t + \epsilon_t \]
  
  *Back Shift Operator:* \( B^i \)

- Autocorrelation
  \[ R_{XX}(t_1, t_2) = E[X_{t_1} X_{t_2}] \]

- Other tasks
  - Semantic Labeling

The quick red fox jumps over the lazy brown dog
Recurrent Neural Networks: Sequence Modeling

- Few projects use pure RNNs, this example is only for pedagogy
- RNN is a model that is as “deep” as the modeled sequence is long
- LSTM’s, Gated recurrent unit,
- No Model Parallel distributed training on the market (June 2019)