A Pragmatic Approach for Machine Learning in D

Jens Mueller
jens.mueller@dunnhumby.com

DConf 2019

Wednesday 8th May, 2019
Logistic Regression
\[ \frac{dN}{dt} = \frac{\lambda}{\sqrt{\pi}} - \beta_0 (N - N_0) (1 - S)^3 - \frac{N - N_0}{t_0} \]

\[ \frac{dS}{dt} = \Gamma_0 \rho_0 (N - N_0) (1 - S)^3 - \frac{S}{t_0} \]

\[ \rho = \frac{\Gamma_0 \rho_0}{V_0 + \rho_0 c} \]

\[ \beta_0 = \frac{1}{S} \]

\[ N = N_0 \]

\[ P = (m \rho)^{1/3} \]
$x_i^T \theta$
Logistic Function

\[ m(x) = \frac{1}{1 + \exp(-x)} \]
Logistic Function
\[ m(x_i; \theta) = \frac{1}{1 + \exp(-x_i^T \theta)} \]
Cross Entropy Loss

\[ H(p, q) = - \sum_x p(x) \log q(x) \]
Cross Entropy Loss

\[-\log(x)\]  
\[-\log(1-x)\]
Cross Entropy Loss

Let \( y_i \in \{y_1, y_2\} \) the set of label classes.

\[
- \sum_x p(x) \log q(x) = - \sum_x 1_{x=y_i} \log q(x)
\]

\[
= - (1_{x=y_1} \log q(x) + 1_{x=y_2} \log(1 - q(x)))
\]

\[
= - (1_{x=y_1} \log m(x_i; \theta) + 1_{x=y_2} \log(1 - m(x_i; \theta)))
\]

\[
= - (1_{x=y_1} \log m(x_i; \theta) + 1_{x=y_2} \log(m(-x_i; \theta)))
\]

\[
= - \log m(y_i; x_i; \theta) \text{ with setting } y_1 = 1 \text{ and } y_2 = -1
\]

\[
= - \log \left( \frac{1}{1 + \exp(-y_i x_i^T \theta)} \right)
\]

\[
= \log \left( 1 + \exp(-y_i x_i^T \theta) \right)
\]
\[ \log \left( 1 + \exp(-y_i \ x_i^T \theta) \right) \]
\[
\sum_{i=1}^{m} \log \left(1 + \exp\left(-y_i \mathbf{x}_i^\top \mathbf{\theta}\right)\right)
\]
\[ \sum_{i=1}^{m} \left[ \log \left( 1 + \exp(-y \odot X \theta) \right) \right]_i \]
Data

» Numeric
» Scalars, vectors, matrices, ...
» Quality vs Quantity
Models

- Defines mapping between data and model parameters
- Classification and regression models
- Factorization methods
- Deep Neural Networks
- ...

...
Losses

- Squared loss
- Huber loss
- Hinge loss
- Absolute loss
- Cross Entropy loss
- Squared hinge loss
Model Parameters

- Updated by minimizing the modeled loss
- Distribute parameters for large models
Optimization

» (Stochastic) Gradient Descent Step

$$\theta_{i+1} = \theta_i - \alpha g$$

» Variations for faster convergence
» Learning rate is important
MXNet

- Mix symbolic and imperative programming
- Dependency scheduler
- Memory efficient and fast execution of symbolic graphs
Why Not MXNet

Trending on Google
Why MXNet

- Exposes C API to access all functionality (not only prediction)
- Both prototyping and production code in D
- Use your D language skills
MXNet Backend

- Basic Linear Algebra Subprograms (Intel MKL, Apple Accelerate, ATLAS, OpenBLAS)
- CUDA and cuDNN
- Open Neural Network Exchange
Symbolic vs Imperative

- Define symbolic network, bind values to it and execute it
- Imperative scheduled and evaluated as you go
MXNet API

- AtomicSymbol
- Symbol
- NDArray
- Executor
- Autograd
- Key-value store
- ...

...
// creating the network
// data arguments
auto x_symbol = new Variable("X"); // feature matrix
scope(exit) x_symbol.freeHandle();
auto y_symbol = new Variable("y"); // label vector
scope(exit) y_symbol.freeHandle();

// network architecture and model parameters
auto w_symbol = new Variable("W");
scope(exit) w_symbol.freeHandle();
auto fc = new FullyConnected(x_symbol, num_classes, w_symbol);
scope(exit) fc.freeHandle();
auto softmax = new SoftmaxOutput(fc, y_symbol);
scope(exit) softmax.freeHandle();
// setup context where computations should happen
auto context = cpuContext();
// size of a training batch
auto batch_size = 100;
// data variable X
auto matrix_x = new NDArray<float>(context, [batch_size, num_pixels]);
scope(exit) matrix_x.freeHandle();
// data variable y
auto vector_y = new NDArray<float>(context, [batch_size]);
scope(exit) vector_y.freeHandle();
Initialize Model Parameters

// initialize parameter W to zero
auto matrix_w = new NDArray!(float)(context, [num_classes, num_pixels], 0f);
scope(exit) matrix_w.freeHandle();

// holds gradient w.r.t. W
auto gradient_w = new NDArray!(float)(context, matrix_w.shape());
scope(exit) gradient_w.freeHandle();
// verify that all arguments are provided in proper order
assert(softmax.arguments == ["X", "W", "y"]);
// define the executor binding the model with data and model parameters
auto executor = new Executor<float>(context, softmax,
    arguments, gradients, gradients_req_type, []);
scope(exit) executor.freeHandle();
Training in Batches

// set batch and ...  
matrix_x.copyFrom(images_batch);  
vector_y.copyFrom(labels_batch);

// make a forward and a backward pass  
executor.forward();  
executor.backward();  
auto step_length = 5e-1f;  
// and a gradient descent step  
gradient_w *= step_length;  
matrix_w -= gradient_w;
Still in 0.x
Targeted platform Linux
Unit- and integration tested
Documented
Demo
Outlook

- Cleanup D1 artifacts
- More examples to learn from
- Automatic generating of Symbols/NDArrays
- Tape-based gradient calculation
- Use uniform function call syntax

```python
auto model = symbol("image").convolve()
   .activate()
   .pool()
   .fullyConnected()
   .softmax();
```
Conclusions
A Pragmatic Approach for Machine Learning in D

Jens Mueller
jens.mueller@dunnhumby.com

DConf 2019

Wednesday 8th May, 2019
Appendix
Data Analysis Cycle
VGG19