Gradient Optimization

Algorithms Comparison

Ali Saleh

Universität Hamburg

Jul 25, 2017

What is a Gradient

The gradient is a multi-variable generalization of the derivative

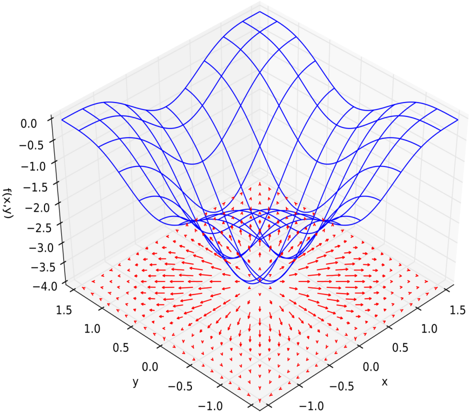

Figure : Gradient of a 2 Variable function projected as vector field [1]

Gradient in Neural Networks

- Used in minimizing model's objective function
- Parameters are updated in the opposite direction of objective function gradient
- The learning rate η determines the size of the steps taken to reach minimum

Computing The Gradient

- Using finite difference
  1. Calculate the partial derivative in each direction
  2. Slow and computationally expensive
- Use calculus to find closed formulas for gradient
  1. Gives faster results
  2. Open for method tweaking
  3. Can be harder to implement

Basic Algorithms

- Stochastic gradient descent (SGD)
  1. Train a model
  2. Evaluate the gradient on a small batch of data
  3. Update the model parameters accordingly
  4. Slowly decreasing learning rate guarantee convergence
- Momentum [2]
  1. Use adaptive learning rate for different dimensions

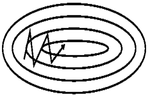

Figure : SGD with momentum [3]

Adaptive Algorithms

- Adagrad [4]
  1. Use adaptive learning based on parameter update frequency
  2. Keeps a matrix of the squares of all past gradients
  3. Divide learning rate by the sum of all past square gradients
  4. Over time learning rate gets infinitesimally small
- Adam [5]
  1. Use adaptive learning based on parameter update frequency
  2. Keeps track of the average of all past gradients
  3. Keeps track of the average of all past square of gradients
  4. Avoid the weakness of Adagrad

Both are more suitable for sparse datasets

Experiment Setup

- Build a convolution neural network using keras
- Use 3 different optimization algorithms (SGD, Adam, Adagrad)
- Optimize network parameters using hyperas (Keras + Hyperopt)
- Use CIFAR10 dataset

b

CIFAR 10

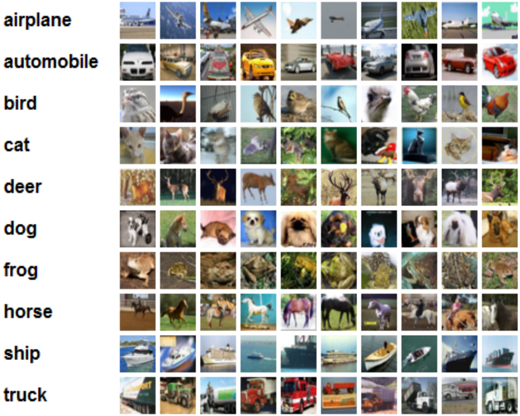

Figure : overview of the CIFAR 10 dataset (courtesy of Alex Krizhevsky)

Time

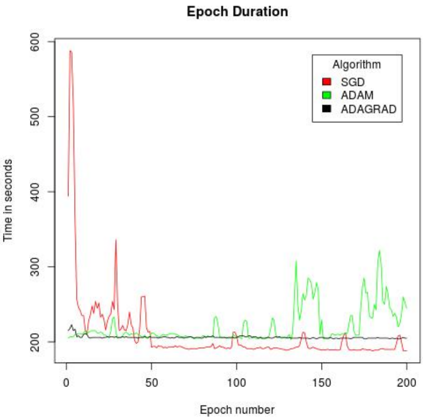

Figure : Total execution time per epoch over time for each optimizer

Loss

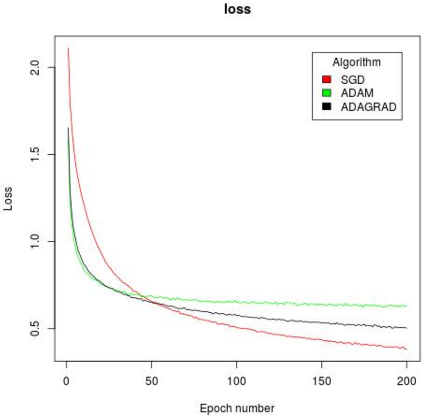

Figure : Loss per epoch on test set

Validation Loss

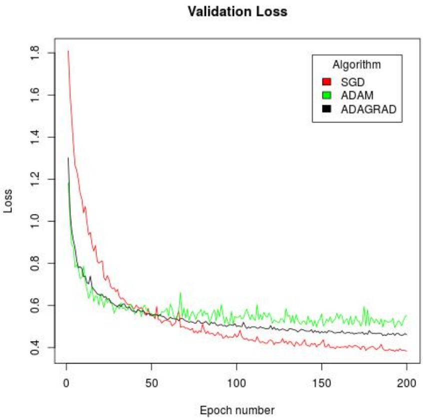

Figure : Loss per epoch for on validation set

Accuracy

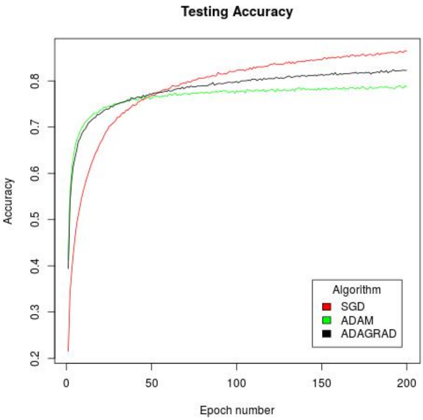

Figure : Accuracy on test set

Validation Accuracy

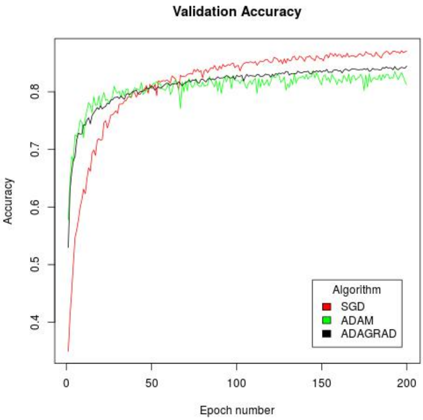

Figure : Accuracy on validation set

Statistics

Metric	SGD min	SGD mean	SGD max	Adam min	Adam mean	Adam max
Time	188.0	209.2	588.0	203.0	220.6	322.0
Loss	0.3822	0.6041	2.1094	0.6224	0.6836	1.5892
Accuracy	0.2151	0.7865	0.8654	0.4132	0.7655	0.7899
Validation Loss	0.3828	0.5363	1.8103	0.4953	0.5708	1.1835
Validation Accuracy	0.3496	0.8150	0.8715	0.5782	0.8075	0.8353

Metric	Adagrad min	Adagrad mean	Adagrad max
Time	205.0	206.1	223.0
Loss	0.5027	0.6187	1.6527
Accuracy	0.3942	0.7827	0.8242
Validation Loss	0.4577	0.5391	1.3018
Validation Loss	0.5298	0.8131	0.8449

This is the last slide.

Any questions?

References

1. https://en.wikipedia.org/wiki/Gradient
2. Qian, N. (1999). On the momentum term in gradient descent learning algorithms. Neural Networks : The Official Journal of the International Neural Network Society
3. http://ruder.io/optimizing-gradient-descent/index.html
4. Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method
5. Kingma, D. P., Ba, J. L. (2015). Adam: a Method for Stochastic Optimization. International Conference on Learning Representations,