Parameter Initialization Strategies

Sargur N. Srihari srihari@cedar.buffalo.edu

Topics

- Importance of Optimization in machine learning
- How learning differs from optimization
- Challenges in neural network optimization
- Basic Optimization Algorithms
 - SGD, Momentum, Nesterov Momentum
- Parameter initialization strategies
- Algorithms with adaptive learning rates
- Approximate second-order methods
- Optimization strategies and meta-algorithms

Role of Initialization

- Non-iterative optimization requires no initialization
 - Simply solve for solution point
- 2. Iterative but converge regardless of initialization
 - Acceptable solutions in acceptable time
- 3. Iterative but affected by choice of Initialization
 - Deep learning training algorithms are iterative
 - Initialization determines whether it converges at all
 - Can determine how quickly learning converges

Keras Initialization

- Initializations define the way to set the initial random weights of Keras layers
- The keyword arguments used for passing initializers to layers will depend on the layer
- Usually it is simply kernel_initializer and bias initializer:

Available Initializers in Keras

Zeros

keras.initializers.Zeros()

Ones

keras.initializers.Ones()

Constant

keras.initializers.Constant(value=0)

Random Normal

keras.initializers.RandomNormal(mean=0.0, stddev=0.05, seed=None)

Random Uniform

keras.initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None)

Truncated Normal

keras.initializers.TruncatedNormal(mean=0.0, stddev=0.05, seed=None)

• Variance Scaing keras.initializers.VarianceScaling(scale=1.0, mode='fan_in', distribution='normal', seed=None)

Orthogonal

keras.initializers.Orthogonal(gain=1.0, seed=None)

Identity

keras.initializers.Identity(gain=1.0)

Lecun uniform

keras.initializers.lecun uniform(seed=None)

Modern Initialization Strategies

- They are simple and heuristic
- Based on achieving nice properties
- But problem is a difficult one
 - Some initial points are beneficial for optimization but detrimental to generalization

Known property: Break Symmetry

- Only property known with certainty: Initial parameters must be chosen to break symmetry
- If two hidden units have the same inputs and same activation function then they must have different initial parameters
- Usually best to initialize each unit to compute a different function
- This motivates use random initialization of parameters

Choice of biases

- Biases for each unit are heuristically chosen constants
- Only the weights are initialized randomly
- Extra parameters such as conditional variance of a prediction are constants like biases

Weights drawn from Gaussian

- Weights are almost always drawn from a Gaussian or uniform distribution
 - Choice of Gaussian or uniform does not seem to matter much but not studied exhaustively
- Scale of the initial distribution does have an effect on outcome of optimization and ability to generalize
 - Larger initial weights will yield stronger symmetrybreaking effect, helping avoid redundant units
 - Too large may result in exploding values

Heuristics for initial scale of weights

• One heuristic is to initialize the weights of a fully connected layer with N_{in} inputs and N_{out} outputs by sampling each weights from Uniform(-r, r) where $r = \frac{1}{\sqrt{N_i}}$

• Another heuristic is normalized initiation with

$$r = \sqrt{\frac{6}{N_{in} + N_{out}}}$$

 Which is a compromise between the goal of initializing all layers to have the same activation variance and the goal of having all layers having the same gradient variance

Initialization for the biases

- Bias settings must be coordinated with setting weights
- Setting biases to zero is compatible with most weight initialization schemes
- Situations for nonzero biases:
 - Bias for an output unit: initialize to obtain right marginal statistics for output
 - Set bias to inverse of activation function applied to the marginal statistics of the output in the training set
 - Choose bias to causing too much saturation at initialization