menu

› Course Shortcuts

› Student Lounge

› Q&A

› Explore a Neural Network

- Module Introduction: Explore a Neural Network
- Notations for Machine Learning
- Approximate Nonlinear Functions
- Formalize Hinge Functions
- Deep Learning with Neural Networks
- Formalize Deep Learning
- Define Transition Functions
- Formalize Transition Functions
- Design a Neural Network
- Implement Cross Entropy Loss Function
- Apply Loss Functions
- Implement Forward Propagation
- Explore Backpropagation
- Formalize Forward and Backpropagation
- Speed Up Training with SGD
- Stochastic Gradient Descent Tips and Tricks
- Formalize Stochastic Gradient Descent
- Mitigate Overfitting
- Formalize Weight Decay and Dropout
- Explore Neural Networks
- Neural Network Cheat Sheet
- Implement a Neural Network
- Module Wrap-up: Explore a Neural Network

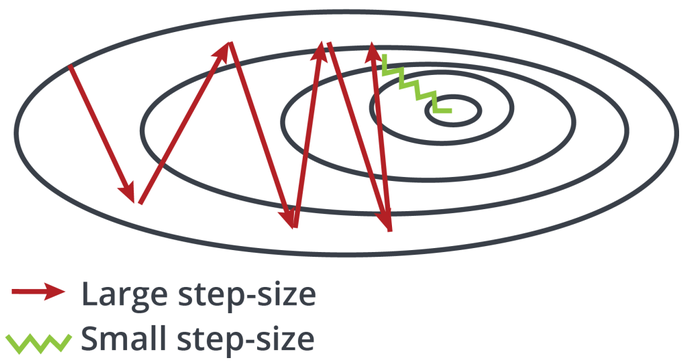
› Use Convolutional Neural Networks for Image Analysis

› Analyze Sequence Data with Neural Sequence Models

› Course Resources

Formalize Stochastic Gradient Descent

Stochastic gradient descent (SGD) is a variant of gradient descent (GD) that is widely used to train different machine learning models, especially a neural network.



☆ Key Points

Stochastic gradient descent is commonly used to train neural networks.

The modification for training deep neural networks is simple because the update is based on small batches instead of all training examples.

Suppose we want to train our neural network using GD. To do so we:

1. Evaluate the output of all training examples
2. Calculate the loss: $\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{x}_i, y_i)$
3. Calculate the gradient of the loss with respect to the weights $\nabla\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \nabla\ell(\mathbf{x}_i, y_i)$
4. Take a gradient step to update the weights
5. Repeat steps 1 to 4 until the training loss does not decrease

However, with deep neural networks, evaluating and calculating the loss and gradients of every single example is very expensive. Thus, researchers came up with SGD:

1. Sample a minibatch of m training examples without replacement
2. Evaluate the output of these m training examples
3. Calculate the loss: $\mathcal{L} = \frac{1}{m}\sum_{i=1}^{m} \ell(\mathbf{x}_i, y_i)$
4. Calculate the gradient of the loss with respect to the weights $\nabla\mathcal{L} = \frac{1}{m}\sum_{i=1}^{m} \nabla\ell(\mathbf{x}_i, y_i)$
5. Take a gradient step to update the weights
6. Repeat steps 1 to 4 until the training loss does not decrease

The modification is rather simple. Instead of each update being based on all training examples, they are based on a small batch of size $m = 64, 128, 256$, etc. We can do this because the loss and gradient can be expressed as a sum of terms. Sampling a small set of examples from the training set and then evaluating the loss and gradient on the small batch can be viewed as estimating the true training loss and gradient. This is where the term stochastic is coming from.