Deep Feedforward Networks - Optimization

Paolo Favaro

Workshop on Machine Learning - Observatoire de Geneve

Contents

- Optimization in Feedforward Neural Networks
  - Batch and mini batch algorithms, stochastic gradient descent, weight initialization
- Based on **Chapter 8** of Deep Learning by Goodfellow, Bengio, Courville

Batch and Minibatch Algorithms

- **Batch** or **deterministic** gradient methods use the whole training set at each iteration

- **Minibatch stochastic** gradient methods use a batch of samples at each iteration

- **Stochastic** gradient methods use only one sample at each iteration

- Today, it is common practice to call minibatch stochastic simply stochastic

Choice of the Batch Size

- Larger batches give better gradients, but the estimate improvement is low
- Small batches might underutilize multicore architectures
- Examples in a batch are processed in parallel; amount of memory defines the maximum size
- GPUs may prefer sizes that are a power of 2
- Small batches may have a regularization effect

Choice of the Batch Size

- The size depends also on the gradient method
  - Methods based on only the loss gradient require small batch sizes
  - Methods based on higher order derivatives (e.g., Hessian) require large batch sizes (to compensate for the larger approximation error)

Shuffling

- An unbiased estimate of the expected gradient requires independent samples

- Using data where the order is fixed might lead to batches where all samples are highly correlated

- Common practice is to randomly visit the training set

- Can save a dataset where the data has been randomly permuted (**data shuffling**)

Basic Algorithms

Stochastic Gradient Descent

- Learning rate ϵ_k and initial parameter θ
- **while** (stopping criterion not met) **do**
  - Sample a minibatch of m examples from the training set with the corresponding targets
  - Compute gradient estimate $\hat{g} \leftarrow \frac{1}{m} \nabla_\theta \sum_i L(f(x_i;\theta), y_i)$
  - Apply update $\theta \leftarrow \theta - \epsilon_k \hat{g}$
- **end while**

Stochastic Gradient Descent

- Probably the most used algorithm in deep learning
- Main setting is the learning rate ϵ_k
  - It is necessary to gradually decrease the learning rate over iteration time k
  - Sufficient conditions (in addition to others on the cost) to guarantee convergence of SGD are that

$$\sum_{k=1}^{\infty} \epsilon_k = \infty \qquad \sum_{k=1}^{\infty} \epsilon_k^2 < \infty$$

Weight Initialization Strategies

- Since the optimization problem is non convex, initialization determines the quality of the solution
- Current initialization strategies are simple and heuristic
- Some initial points may be beneficial to the optimization task, but not to generalization
- One criterion is that the initial parameters need to **break the symmetry** between different units

Weight Initialization Strategies

- Two hidden units with the same activation function and inputs should have different initial parameters
- Otherwise a deterministic learning algorithm will update both of these units in the same way
- The goal of diversifying the computed functions motivates random initialization
- Random weights can be obtained from a Gaussian or Uniform distribution

Weight Initialization Strategies

- The magnitude of the random variable matters
  - Large weights may be more effective in breaking symmetry and in preserving gradients through back-propagation
  - Large weights may also lead to exploding gradients, saturation of nonlinear units, or unstable behavior and chaos (in recurrent networks)

Learning Based Initialization Strategies

- Another strategy is to initialize weights by **transferring weights** learned via an unsupervised learning method

- This is also a technique called **fine-tuning** which aims at exploiting small annotated datasets by combining them with large unlabeled ones

Adaptive Learning Rates

- The learning rate is one of the most difficult parameters to set

- It has a significant impact on the model performance

- It is therefore treated as a hyperparameter that requires adjustment during training

Choosing the Optimization Algorithm

- Currently there is no consensus on what algorithm performs best
- Most popular choices are: SGD, SGD+Momentum, RMSProp, RMSProp+Momentum, AdaDelta, Adam
- Strategy: Pick one and get familiar with the tuning