Deep Learning

4DV661

Optimization for Training Deep Models

Welf Löwe

Course structure

1. Introduction
2. Applied Math Basics
3. Deep Feedforward Networks
4. Regularization for Deep Learning
5. Optimization for Training Deep Models
6. Convolutional Networks
7. Sequence Modeling: Recurrent and Recursive Nets
8. Practical Methodology
9. Applications

Agenda for today

- How Learning Differs from Pure Optimization
- Challenges in Neural Network Optimization
- Notebook
    - Basic Algorithms
    - Parameter Initialization Strategies
    - Algorithms with Adaptive Learning Rates
    - Approximate Second-Order Methods
- Optimization Strategies and Meta-Algorithms

How Learning Differs from Pure Optimization

- Learning optimizes a surrogate loss function on training data (sample distribution of real-world data) hoping that, in general, the actual goal function gets optimized on the real-world data
  - the actual goal function, in turn, is often not identical but only similarly coupled to the utility for the end user
- Gradient-based approaches exploit that the loss function decomposes on training data points allowing for stochastic estimations of the gradient (minibatch approaches)
  - Early stopping does not refer to the surrogate loss function but to the actual goal function on the validation set (hence, separate validation and test set)

What we want vs what we get

- Actual utility, users' appreciation of model accuracy, could be hard to formalize, expensive to measurable
- Loss function, our modelling of utility, should be minimized for the distribution of data, which is not known (in general)
- Surrogate loss as the actual loss is not differentiable (e.g., class loss vs negative log likelihood of the correct class)
- Empirical loss using training data as sample of the distribution
- For computational efficiency, we can estimate gradient by computing them for a random subsample (minibatch)
  - The size of the minibatch (batch size) is an important hyperparameter

Why is this still challenging?

- Some functions are easier to optimize than others
- The “best” ones are the convex functions:
  - A *convex* function is a function where the line segment between any two points on its graph lies above or on the graph itself.
- We know neural networks are not convex
- Therefore, we need to address the challenges (next slides) with optimization algorithm variants (notebook)

Challenges in Neural Network Optimization

- Overflow and underflow problems
- Ill-conditioning
    - Gradients can be “steep” in some directions and very flat in others.
    - Happens when the Hessian matrix H (second derivative) of the loss function has eigenvalues vary greatly in magnitude:
        - it means the *curvature* of the loss function is very different depending on the direction you move in the parameter space
            - Large eigenvalues mean very *steep* curvature in some directions.
            - Small eigenvalues mean very *flat* curvature in other directions.
        - the gradient points mostly along the steep direction.
            - large moves along the steep direction (where you don't need to move much) and
            - tiny, slow moves along the flat direction (where you need bigger moves to reach the minimum).
        - causes slow zig-zagging towards the minimum or not even converging
    - Can be checked if the condition number ($\lambda_{max}/\lambda_{min}$) of matrix H is high

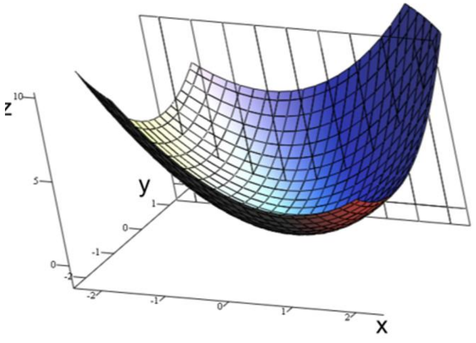
Challenges in Neural Network Optimization (cont'd)

- Vanishing gradients:
  - Gradient based optimization: each of the NN's weights receives an update proportional to the gradients (partial derivative of the error function) with respect to the current weight
  - As the NN depth increases, the gradient magnitude typically is expected to decrease for the earlier layers, especially, if output layers have (close to) zero error gradients
  - Why:
    - traditional activation functions and backpropagation computes gradients by the chain rule.
    - Multiplying of small numbers to compute gradients of the early layers
    - Gradient (error signal) decreases exponentially with the depth while the early layers train very slowly.
  - Slows down training process and, in the worst case, may completely stop the NN from further training
- Exploding gradient: When activation functions are used whose derivatives can take on larger values, one risks encountering the oposite problem.

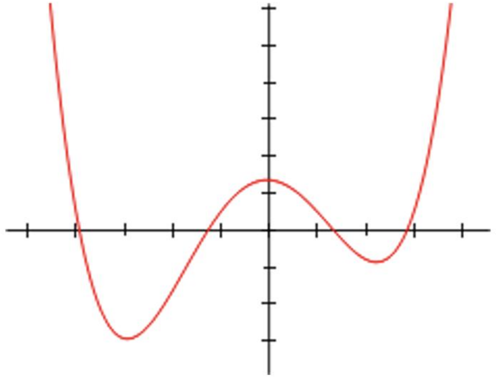
Challenges in Neural Network Optimization (cont’d)

- Local minima (non-convex function)
  - A model is said to be identifiable if a sufficiently large training set can rule out all but one setting of the model’s parameters.
  - NN have a model identifiability problem due to weight space symmetry (and other issues)
  - experts suspect that, for sufficiently large neural networks, most local minima have a low-cost function value, i.e., not a practical problem (active area of research)
- Plateaus, saddle points and other flat regions
  - Points with (almost) zero gradient
  - Too small (zero) step size
- Cliffs
  - exploding gradients
  - too large step size

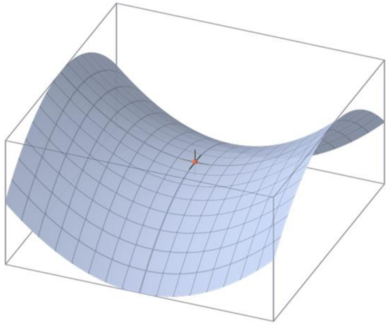
Visualizations of some solution spaces



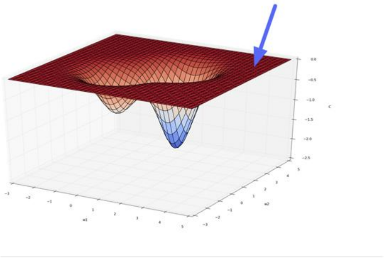
convex functions



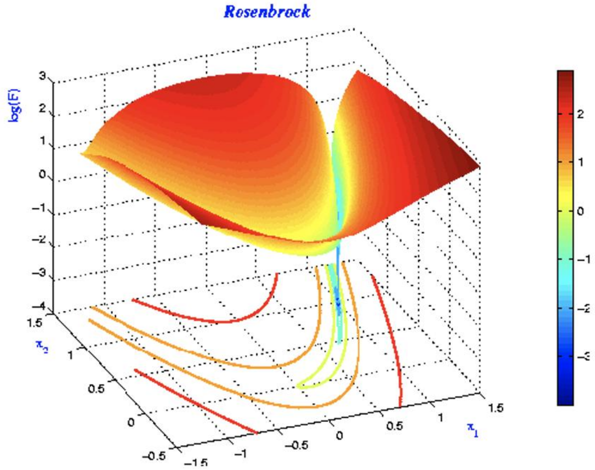
local minima



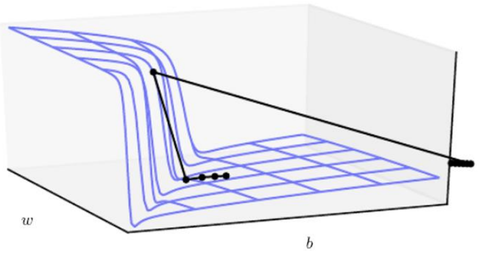
saddle points



plateaux



narrow ravines



cliffs

Challenges in Neural Network Optimization (cont'd)

- Inexact gradients, Hessians
  - Due to sampling (minibatches)
- No second order derivatives (excludes some optimizations, e.g., Newton)
- Poor correspondence between local and global search space structure
- Theoretical limits
  - Exponential or NP hard problems when weights are integers
  - Intractable

Pointers

- Slides of the chapter’s discussion: https://docs.google.com/presentation/d/1TBXYGm2IWQKYIDyeoailxd6R5_xNTN1HXguraRpOKNM
- Demo of some algorithms in the notebook: https://github.com/WelfLowe/Public-ML-Notebooks/blob/master/Variants%20of%20stochastic%20gradient-based%20optimization.ipynb

Assignment 7

Watch the videos of and read:

- Chapter 9: Convolutional Networks (41 pages)
- Chapter 10: Convolutional Networks (21 pages)
- Create a new Jupyter notebook in Python code based on the notebook ”Assignment7-DL-Matlab.pdf” (Variants of stochastic gradient-based optimization)
  - Pick 2 hyper-parameterized optimization approaches
  - Optimize the hyper-parameters (using a grid-based approach) and
  - Learn the parameters with the optimal hyper-parameter setting
  - Plot the loss history for 20 learning steps in one diagram
  - Interpret your results
- Deadline: 2025-05-06 (before the next lecture)