Deep Learning

Summer Semester 2024

Monday, May 6, 2024

Prof. Dr.-Ing. Christian Bergler | OTH Amberg-Weiden

Optimization
Overview

Osthayerische Technische Hochschule Amberg-Weiden

Topics From Last Time: Multi-Layer Perceptron

- Multi-Layer fully connected neural networks (Multi-Layer Perceptron)
- Activation functions
- Universal approximation theorem
- Forward propagation and backward propagation

Topics of Today: Optimization

- Model training with large data sets
- Variants of the gradient (descent) method

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Optimization

Osthayerische Technische Hochschule Amberg-Weiden

Linear Regression, Logistic Regression, Softmax Regression

- Mathematically “friendly” problems
- Convexity of the loss function
- No local minima of the loss function

Deep Learning

- Highly complex model functions
- No convexity regarding the loss functions
- Local minima
- Model training is based on very big data corpora

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Optimization

Gradient (Descent) Method (Recap)

The iteration rule for approximating a minimum of a function $f : \mathbb{R}^n \to \mathbb{R}$ using the gradient (descent) method is

$$\mathbf{x}^k = \mathbf{x}^{k-1} - \alpha \nabla f(\mathbf{x}^{k-1}), \quad k = 1, 2, \dots,$$

where $\alpha > 0$ denotes the learning rate.

Questions:

- Which function is minimized in connection with the training of (parameterized) machine learning models?
- Which special shape does it have?

Optimization

Gradient (Descent) Method for Machine/Deep Learning

- The function to be minimized is a loss function which represents the error as a function of the parameters.
- The objective/error/cost/loss function has additive form

$$L(\theta) = \frac{1}{m} \sum_{i=1}^{m} L^{(i)}(\theta)$$

where $L^{(i)}(\theta)$ denotes the error of the ith sample of the training dataset

- Least Squares Functional:

$$L(\theta) = \frac{1}{m} \sum_{i=1}^{m} \underbrace{\frac{1}{2}(f_\theta(\mathbf{x}^{(i)}) - y^{(i)})^2}_{=L^{(i)}(\theta)}$$

#samples

mittlerer Loss aller Samples

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Optimization

Taking into account the special shape of the function L, the gradient (descent) method has the following formulation:

Gradient (Descent) Method for Machine Learning/Deep Learning

Be

$$L(\theta) = \frac{1}{m} \sum_{i=1}^{m} L^{(i)}(\theta)$$

a cost function for training a parameterized machine learning method, where $L^{(i)}$ denotes the loss given by the ith sample. Then the iteration rule of the gradient (descent) method is

$$\theta^k = \theta^{k-1} - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} \nabla L^{(i)}(\theta^{k-1}) \,, \quad k = 1, 2, \ldots$$

gebildeter Gradient über alle Samples

This method is also referred to in the following as **classical gradient (descent) method** or **batch gradient (descent) method**

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Optimization

Optimization

Gradient (Descent) Method for Machine/Deep Learning

Example Linear Regression: $f_\theta(\mathbf{x}) = \sum_{i=0}^{p} \theta_i x_i$, where $x_0 = 1$, and the least squares cost functional is

$$L(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} (f_\theta(\mathbf{x}^{(i)}) - y^{(i)})^2$$

ableiten nach $f_\theta(x^{(i)})$

The partial derivatives of L according to the weights result in

$$\frac{\partial L(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (f_\theta(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)}, \quad j = 0, \ldots, p$$

We had already derived the vectorized form:

$$\nabla L(\theta) = \frac{1}{m} (X^T X \theta - X^T \mathbf{y})$$

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Optimization

Optimization

Gradient (Descent) Method for Machine/Deep Learning

Gradient (Descent) Method Linear Regression (Non-Vectorized)

for $k = 1, 2, \ldots$ **do** - # iterations

  for $j = 0, \ldots, p$ **do** # Neuronen/Gewichte

$$\theta_j^k := \theta_j^{k-1} - \frac{\alpha}{m} \sum_{i=1}^{m} (f_\theta(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)}$$ i-tes Sample

  end for

end for

Samples

Gradient (Descent) Method Linear Regression (Vectorized)

for $k = 1, 2, \ldots$ **do**

$$\theta^k = \theta^{k-1} - \frac{\alpha}{m}(X^T X \theta^{k-1} - X^T \mathbf{y})$$

end for

Matrix ALLER Gewichte

Question: What are the advantages and disadvantages of these two variants in terms of algorithmic implementation on the computer?

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Optimization

Optimization

Vecotrization vs. Non-Vectorization

Advantages and Disadvantages of the Non-Vectorized Implementation

+ Good traceability of the formulation
+ Low memory requirement even with large data sets
- Nested loops
- No ideal utilization of the hardware (e.g. GPUs)

Advantages and Disadvantages of the Vectorized Formulation

- Derivation of the formulation required
+ Easy to implement
- High memory requirement, especially for large data sets
+ Use of efficient libraries for linear algebra utilizing the hardware

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Optimization

Osthayerische Technische Hochschule Amberg-Weiden

Dataset Size for Deep Learning

- Complex models with high-dimensional data require a lot of training data (overfitting, curse of dimensionality)
- Deep learning particularly shows its strength when using very large datasets
- Applications: Computer Vision, Acoustic Analysis, Natural Language Processing
- Available Data Corpora: HuggingFace, Kaggle, PapersWithCode

Consequences Model Training

- Vectorization (entire data corpus) is not possible due to high memory requirements
- Summation across all samples for gradient calculation can be (very) expensive

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Optimization

Optimization

Stochastic Gradient (Descent) Method

Idea: Approximation of the gradient $\nabla L(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla L^{(i)}(\theta)$ using the gradient of the costs of a randomly selected sample $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$:

$$\nabla L(\theta) \approx \nabla L^{(i)}(\theta)$$

where $i \in \{1, \ldots, m\}$ is chosen at random

Stochastic Gradient (Descent) Method

for $k = 1, 2, \ldots$ **do** - # Iterations
  Randomly shuffle the dataset („shuffling") → $(x^{(i)}, y^{(i)})$ → feature, Label
  for $i = 1, 2, \ldots, m$ **do** - #samples
    for $j = 0, \ldots, p$ **do** - #weights / Weights
      $\theta_j := \theta_j - \alpha \frac{\partial L^{(i)}(\theta)}{\partial \theta_j}$
    end for
  end for
end for

Note: The entire dataset is typically run through several times (corresponds to the run variable k in the pseudo code above, known as epochs)

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Optimization

Optimization

Traditional vs. Stochastic Gradient (Descent) in Deep Learning

- The parameter update for the stochastic gradient (descent) method is performed using a single sample, for the classical gradient method using all samples
- The individual updates can be carried out very efficiently with the stochastic gradient (descent) method and require little memory
- The parameter update in the classical gradient method requires a summation over all samples (expensive) or a vectorized implementation (memory-intensive)
- The trajectories of the iterates are generally noisier with the stochastic than with the classical gradient method. The approximation quality is usually worse than with the classical gradient method.
- In order for the method to converge, the learning rate is typically reduced dynamically during the training process ($\rightarrow$ noise reduction)
- Question: Why it can be okay to accept the supposedly poorer approximation quality of the stochastic gradient method?

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Optimization

Optimization

Mini-Batch Gradient (Descent)

„Average Solution“ between Classical and Stochastic Gradient (Descent) Method

- Approximation of the gradient of L using several samples that originate from a randomly selected subset $B \subset \{1, \ldots, m\}$:

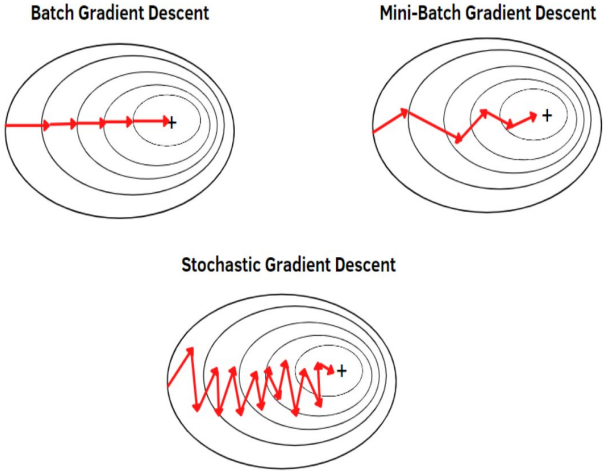
$$\nabla L(\theta) \approx \frac{1}{|B|} \sum_{i \in B} \nabla L^{(i)}(\theta)$$

- Such a subset $B \subset \{1, \ldots, m\}$ of the samples is called mini-batch
- Typically, $b := |B| << m$ applies
- The dataset is first shuffled and then split into disjoint minibatches. A parameter update is then calculated for each minibatch
- Vectorization via mini-batch possible
- Number of Iterations with a given batch-size (mini-batch) leads to an entire epoch (whole data corpus)

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Optimization

Optimization

Traditional vs. Stochastic vs. Mini-Batch Gradient Descent Trajectory



Source: https://statusneo.com/efficientdl-mini-batch-gradient-descent-explained/
Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Optimization

Mini-Batch Gradient Descent (Vectorized)

for $k = 1, 2, \ldots$ **do**
  Shuffle the dataset randomly („shuffling“)
  Split the dataset into disjoint batches
  $B_1 = \{1, \ldots, b\}, B_2 = \{b+1, \ldots, 2b\}, \ldots$
  for $j = 1, 2, \ldots, \#Batches$ **do**
    $\theta := \theta - \frac{\alpha}{|B_j|} \sum_{i \in B_j} \nabla L^{(i)}(\theta)$
  end for
end for

Remarks:

- If m (length of the training dataset) is not a multiple of the batch-size b, the last batch may consist of fewer than b elements ($\rightarrow$ PyTorch allows to drop or keep it!)
- For $b = 1$ the stochastic gradient method (SGD) is used, for $b = m$ the classical method (BGD)
- The parameter updates are generally less noisy than with the stochastic gradient method
- The batch-size b is a hyperparameter whose choice depends on the problem. Often only trial and error helps to determine a suitable batch-size

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Lecture Deep Learning, Optimization

Additional Notes

Optimization

Gradient (Descent) Method Using Momentum-Update

Idea

- Provide the gradient (descent) method with a "memory" and consider not only the current gradient but also historical updates when calculating the current update
- Goals: Increase convergence speed, reduce noise & convergence issue of "local minima"
- Can be combined with mini-batch gradient (descent) method

Gradient (Descent) Method Using Momentum-Update

Iteration rule for approximating a minimum of the function $f : \mathbb{R}^n \to \mathbb{R}$:

$$\mathbf{z}^k = \beta \mathbf{z}^{k-1} + \nabla f(\mathbf{x}^{k-1}) \,,$$

$$\mathbf{x}^k = \mathbf{x}^{k-1} - \alpha \mathbf{z}^k$$

where α is the learning rate and $\beta \in (0, 1)$ is a weighting parameter for the momentum ($\beta = 0$ corresponds to the classic gradient method)

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Lecture Deep Learning, Optimization

Optimization

Epochs

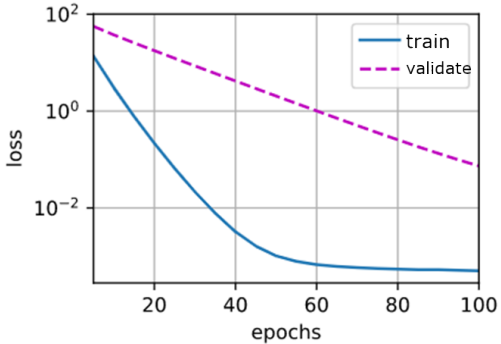
- In the stochastic gradient (descent) method, the parameter updates are carried out using a single training sample, while in the mini-batch gradient (descent) method, they are carried out using batches that form a decomposition of the training dataset
- A complete run of the entire training data set is referred to as epoch
- In the classical gradient method, each iteration/parameter update corresponds to an epoch.
- In the stochastic gradient method, an epoch ends after m parameter updates.
- In the stochastic gradient method with minibatches, an epoch is completed when all batches have been processed once.

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Optimization

Optimization

Training Process Evaluation

- To assess the progress of the model training and its quality, the temporal development of the loss (e.g. on the training and validation dataset) can be plotted in a diagram
- In such diagrams, the number of epochs is plotted on the x axis



Question: What would you recommend in the above situation and why?

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Optimization

Optimization

Summary and Outlookk

Osthayerische Technische Hochschule Amberg-Weiden

Summary

- Classical gradient method
- Stochastic gradient method
- Stochastic gradient method with minibatches
- Gradient method with momentum
- Epochs, visualisation of the learning progress

Outlook

- Exercise: Implementation of the variants for univariate linear regression
- Parameter initialization for MLPs
- MLPs in PyTorch
- Regularization

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Optimization

Further Questions?

https://www.oth-aw.de/hochschule/ueber-uns/personen/bergler-christian/

Source: https://emekaboris.medium.com/the-intuition-behind-100-days-of-data-science-code-c98402cdc92c