Deep Learning

Summer Semester 2024

Monday, May 13, 2024

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

Network Initialization, Normalization, Regularization

Overview

Topics From Last Time: Optimization

- Model Training using large Data Corpora
- Classical Gradient Descent
- Stochastic Gradient Descent
- Mini-Batch Stochastic Gradient Descent

Topics of Today: Parameter Initialization and Batch Normalization

- Initialization of Weights during Model Training
- Vanishing and Exploding Gradients
- Batch Normalization

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Init-, Norm- & Regularization

Network Initialization, Normalization, Regularization

Parameter Initialization

- The parameters for deep learning are determined using an iteration method (e.g. gradient descent). This requires a suitable initialization of the weight matrices and bias vectors.

Question:

What is the best way to initialize the parameters?

Ideas:

- Initialization with zero/static (constant) initialization
- Initialization with random values
  - From a uniform distribution $U(-r, r)$ with r to be specified
  - From a normal distribution $\mathcal{N}(\mu; \sigma^2)$ with $\mu = 0$ and σ to be determined

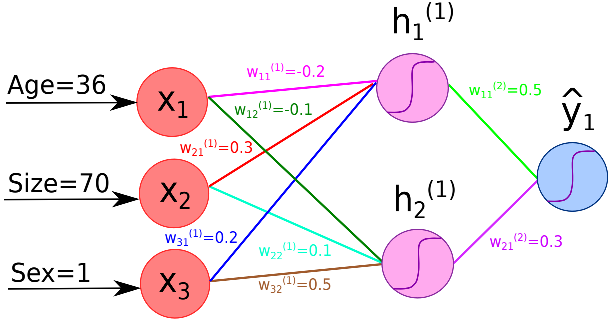
Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Init-, Norm- & Regularization

Network Initialization, Normalization, Regularization

Parameter Initialization

Static Initialization Example:

We consider the following network for modelling a binary classification problem. The sigmoid activation function is used as the activation function in the hidden layer and the output layer.



Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Init-, Norm- & Regularization

Network Initialization, Normalization, Regularization

Static (Constant) Initialization – Forward Propagation

Input $\mathbf{x} = (x_1, x_2, x_3)^T$:

$$\begin{aligned}
\mathbf{z}^{[1]} &= \begin{pmatrix} w_{11}^{[1]} x_1 + w_{21}^{[1]} x_2 + w_{31}^{[1]} x_3 + b_1^{[1]} \\ w_{12}^{[1]} x_1 + w_{22}^{[1]} x_2 + w_{32}^{[1]} x_3 + b_2^{[1]} \end{pmatrix} \\
&= \begin{pmatrix} w_{11}^{[1]} & w_{21}^{[1]} & w_{31}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} & w_{32}^{[1]} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} b_1^{[1]} \\ b_2^{[1]} \end{pmatrix} \\
&= \mathbf{W}^{[1]^T} \mathbf{x} + \mathbf{b}^{[1]}
\end{aligned}$$

$$\mathbf{h}^{[1]} = \sigma(\mathbf{z}^{[1]})$$

$$\begin{aligned}
z^{[2]} &= w_{11}^{[2]} h_1^{[1]} + w_{21}^{[2]} h_2^{[1]} + b_1^{[2]} = \begin{pmatrix} w_{11}^{[2]} & w_{21}^{[2]} \end{pmatrix} \begin{pmatrix} h_1^{[1]} \\ h_2^{[1]} \end{pmatrix} + b_1^{[2]} \\
&= (\mathbf{W}^{[2]})^T \mathbf{h}^{[1]} + b^{[2]}
\end{aligned}$$

$$\hat{y} = \sigma(z^{[2]})$$

OTH Lecture – Prof. Dr.-Ing. Christian Bergler

Additional Notes

OTH Lecture – Prof. Dr.-Ing. Christian Bergler

Additional Notes

OTH Lecture – Prof. Dr.-Ing. Christian Bergler

Additional Notes

Network Initialization, Normalization, Regularization

Static (Constant) Initialization

Forward Propagation Single Sample: Observation: If all weights are initialised with constant values, then

$$z_1^{[1]} = z_2^{[2]} \quad \text{und} \quad h_1^{[1]} = h_2^{[2]}$$

Backward Propagation of the Loss

- In the following, we consider the cost $L(\hat{y}, y)$, which denotes the sample $\mathbf{x} = (x_1, x_2, x_3)^T$, where L is a differentiable cost function, $\hat{y}$ is the model output associated with *textbfx* and y is the label associated with $\mathbf{x}$
- Task: Calculate the partial derivatives of L according to the weights $\mathbf{W}^{[1]}$, $\mathbf{b}^{[1]}$, $\mathbf{W}^{[2]}$, $\mathbf{b}^{[2]}$ and interpret the result

Conclusion: Avoid Constant Initialization!

Initializing the weights with a constant value can lead to undesirable symmetries in the network and should be avoided

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Init-, Norm- & Regularization

Network Initialization, Normalization, Regularization

Random Initialization

Since initialization with constant values is not expedient, we investigate strategies for initializing the weights with random values in the following.

Desirable Properties of the Parameter Initialization

- Avoidance of unwanted symmetries (not fulfilled with constant initialization)
- Avoidance of disappearing or exploding activations during forward propagation
- Avoidance of disappearing or exploding gradients during backward propagation

Consideration of the Following Cases

- Type of random initialization: normal distribution or uniform distribution
- Activation function used: hyperbolic tangent, sigmoid, ReLU

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Init-, Norm- & Regularization

Network Initialization, Normalization, Regularization

Vanishing and Exploding Gradient Descent Problem

Vanishing Gradient Problem

- With backward propagation, derivatives of the activation functions are successively multiplied (from the output layer to the input layer) (chain rule!). As a result, there is a risk that the partial derivatives become very small after the weights in the vicinity of the input layer and that model training comes to a standstill there. This effect is called "vanishing gradient problem"
- The effect is particularly noticeable when the hyperbolic tangent or the sigmoid function are used as activation functions

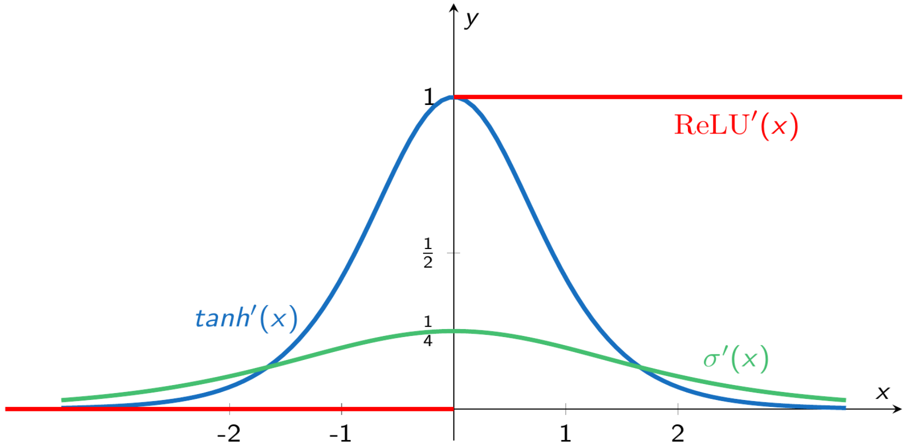
Exploding Gradient Problem

- Analogue to the "vanishing gradient problem" there is the reverse effect called "exploding gradient problem"
- This is caused by exploding values for the activations

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Init-, Norm- & Regularization

Network Initialization, Normalization, Regularization

Derivative Functions of Common Activation Functions



Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Init-, Norm- & Regularization

Network Initialization, Normalization, Regularization

Backpropagation – Recap

Recursive Calculation of Variables $\delta^{(l)} = \frac{\partial L}{\partial z^{(l)}}$:

$$\delta^{(L)} = \nabla_{\hat{y}} L \odot (f^{(L)})'(z^{(L)}) \ ,$$

$$\delta^{(l)} = \mathbf{W}^{(l+1)} \delta^{(l+1)} \odot (f^{(l)})'(z^{(l)}) \ , \quad l = L-1, \ldots, 1$$

Note: where $\odot$ denotes the dot product between matrices.

Calculation of the Partial Derivatives with respect to the Variables $\delta^{(l)}$

$$\frac{\partial L}{\partial b_j^{(l)}} = \delta_j^{(l)} \ , \quad l = 1, \ldots, L$$

$$\frac{\partial L}{\partial w_{kj}^{(l)}} = h_k^{(l-1)} \delta_j^{(l)} \ , \quad l = 1, \ldots, L$$

Network Initialization, Normalization, Regularization

Xavier Initialization

Assumptions:

- The inputs are independent and identically standard-normal distributed (standardization)
- Initialization of the weight matrices $\mathbf{W}^{(l)}$ with independent and identically distributed random values with expected value zero (e.g. from a normal distribution or uniform distribution) wegen sigmoid und tanh
- Initialization of the bias vectors $\mathbf{b}^{(l)}$ with zero
- Use of the hyperbolic tangent as an activation function

Target: "Controlled" Forward and Backward Propagation

- Initialization of the weight matrices $\mathbf{W}^{(l)}$ in such a way that
  (1) the variances of the activations $\mathbf{h}^{(l)}$ are the same in all layers,
  (2) the variances of the errors $\delta^{(l)}$ are the same in all strata.
- Thus: Containment of the problem of exploding and vanishing values in the activations and the exploding and vanishing gradients

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Init-, Norm- & Regularization

Network Initialization, Normalization, Regularization

Xavier Initialization

Resulting Conditions:

Let n_l be the number of activations in layer l. Then the following two conditions result from requirements (1) and (2):

$$n_{l-1}\mathrm{Var}(\mathbf{W}^{(l)}) = 1 \qquad \text{(aus Vorwärts-Propagation)}$$

$$n_{l}\mathrm{Var}(\mathbf{W}^{(l)}) = 1 \qquad \text{(from backward propagation)}$$

If all layers have the same length, they are fulfilled if as variance $\mathrm{Var}(\mathbf{W}^{(l)}) = \frac{1}{n_l}$ is selected. Otherwise, the following condition is proposed as „mean path“

$$\mathrm{Var}(\mathbf{W}^{(l)}) = \frac{1}{\frac{1}{2}(n_{l-1} + n_l)} = \frac{2}{n_{l-1} + n_l}$$

Comments:

- The above conditions have not yet been derived using a specific distribution assumption
- In the following, we will examine what these conditions look like for the case of normal distribution and uniform distribution

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Init-, Norm- & Regularization

Network Initialization, Normalization, Regularization

Xavier Initialization

For a normally distributed random variable $X \sim \mathcal{N}(\mu; \sigma^2)$, $\mathrm{Var}(X) = \sigma^2$ applies, in order to obtain the following initialization scheme:

Xavier Initialization with Normally Distributed Weights

$$\mathbf{W}^{(l)} \sim \mathcal{N}\left(0, \frac{2}{n_{l-1} + n_l}\right)$$

For an equally distributed random variable $X \sim U(-r, r)$, $\mathrm{Var}(X) = \frac{r^2}{3}$, resulting in the following initialization scheme:

gleichsetzen
r auflösen

Xavier Initialisation with Equally Distributed Weights

$$\mathbf{W}^{(l)} \sim U(-r_l, r_l) \text{ mit } r_l = \sqrt{\frac{6}{n_{l-1} + n_l}}$$

Note: These initialization schemes were derived for the hyperbolic tangent as an activation function. For the sigmoid function, slightly modified distribution parameters

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Init-, Norm- & Regularization

Network Initialization, Normalization, Regularization

He-Initialization – Usage of the ReLU Activiation Function

- The Xavier initialization is suitable for activation functions that are approximately linear in $x = 0$. This does not apply to the ReLU function, which is not differentiable there
- For the ReLU function, the following condition for the „control" of the gradients and the activation functions was derived in He et. al. (2015):

$$\mathrm{Var}(\mathbf{W}^{(l)}) = \frac{2}{n_{l-1}}$$

He-Initialization with Normally Distributed Weights

$$\mathbf{W}^{(l)} \sim \mathcal{N}\left(0, \frac{2}{n_{l-1}}\right)$$

He-Initialization with Equally Distributed Weights

$$\mathbf{W}^{(l)} \sim U(-r_l, r_l) \;\; \text{mit } r_l = \sqrt{\frac{6}{n_{l-1}}}$$

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Init-, Norm- & Regularization

Network Initialization, Normalization, Regularization

Summary: Parameter Initialization

Ostbayerische Technische Hochschule Amberg-Weiden

Parameter Initialization

Activation Function	**Uniform Distribution** $\mathbf{W}^{(l)} \sim U(-r_l, r_l)$	**Normal Distribution** $\mathbf{W}^{(l)} \sim \mathcal{N}(0, \sigma_l^2)$
$\tanh$	$r_l = \sqrt{\frac{6}{n_{l-1}+n_l}}$	$\sigma_l = \sqrt{\frac{2}{n_{l-1}+n_l}}$
Sigmoid	$r_l = 4\sqrt{\frac{6}{n_{l-1}+n_l}}$	$\sigma_l = 4\sqrt{\frac{2}{n_{l-1}+n_l}}$
ReLU	$r_l = \sqrt{\frac{6}{n_{l-1}}}$	$\sigma_l = \sqrt{\frac{2}{n_{l-1}}}$

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Init-, Norm- & Regularization

Network Initialisation, Normalization, Regularisation

Batch Normalization

Ostbayerische Technische Hochschule Amberg-Weiden

Challenges in the Course of the Training Process

- Suitable parameter initialization required
- Disappearing and exploding gradients
- Instability: small changes in the parameters have a strong effect on the activation of layers further to the right in the network

$\rightarrow$ Consequences: numerical problems, small learning rates required, slow training process

Basic Idea & Goal of Batch Normalization

- Perform a normalization over the dataset for each node of a layer
- Instead of $\mathbf{z}^{(l)}$, use a normalized vector $\tilde{\mathbf{z}}^{(l)} = \mathrm{BN}(\mathbf{z}^{(l)})$ as input to the activation function of layer l
- Reduce the dependency on weights of previous layers and stabilise or accelerate the training process
- S. Ioffe and C. Szegedy, *Batch Normalization: Accerlerating Deep Network Training by Reducing Internal Covariate Shift*, International conference on machine learning, 2015

OTH Lecture – Prof. Dr.-Ing. Christian Bergler

Additional Notes

Network Initialization, Normalization, Regularization

Ostbayerische Technische Hochschule Amberg-Weiden

Batch Normalization

n: #Features
m: Batchsize
k: #Neuronen im Layer

Procedure Model Training

Let $\mathbf{z}^{(1)}, \dots \mathbf{z}^{(m)} \in \mathbb{R}^n$ be the inputs to a hidden layer when the training data $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}$ is sent through the network.

- Calculate the mean values and the standard deviations of the inputs $z_k^{(i)}$ in the activation function for each node $k \in \{1, \dots, n\}$ of the hidden layer:

vektor

vektor

$$\mu_k = \frac{1}{m} \sum_{i=1}^{m} z_k^{(i)} \quad \text{and} \quad \sigma_k^2 = \frac{1}{m} \sum_{i=1}^{m} (z_k^{(i)} - \mu_k)^2$$

- Normalization:

durch 0 teilen vermeiden

$$z_{\text{norm},k}^{(i)} = \frac{z_k^{(i)} - \mu_k}{\sigma_k + \varepsilon} , \quad i = 1, \dots, m , \ k = 1, \dots, n , \quad \varepsilon > 0$$

- Define:

$$\text{BN}_{\gamma_k, \beta_k}(z_k^{(i)}) := \tilde{z}_k^{(i)} := \gamma_k z_{\text{norm},k}^{(i)} + \beta_k , \quad i = 1, \dots, m , \ k = 1, \dots, n$$

with new (trainable) parameters γ_k and β_k.

- Use $\tilde{\mathbf{z}}^{(i)}$ instead of $\mathbf{z}^{(i)}$ to calculate the activations of the layer

Network Initialization, Normalization, Regularization

Batch Normalization – Comprehensive Questions

- How many additional parameters are created if batch normalization is applied in a layer with n nodes? 2 * n
- For which values of γ_k and β_k does $z_k^{(i)} = \tilde{z}_k^{(i)}$? 1, 0

Batch Normalization in Practice

- If batch normalization is applied to a layer, no bias values are required for its nodes (constant additive contributions are omitted when averaging)
- If the gradient method with minibatches is used for training, normalization is performed on each minibatch, not on the entire training dataset
- When the model is later applied (possibly to individual samples), the mean values and standard deviations calculated for the individual batches during the training phase are averaged and used for normalization
- In software packages, batch normalization is usually available as a separate layer that can be inserted as required

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Init-, Norm- & Regularization

Network Initialization, Normalization, Regularization

Effect of Batch Normalization

- The inputs into the activations of a layer (or their distributions) are less sensitive to changes in the weights of previous layers when using batch normalization (reduction of the so-called “covariate shift”)
- Experience has shown that the effects of exploding and disappearing gradients are reduced
- Experience has shown that the training process is more stable and faster
- If training is carried out using the stochastic gradient method with minibatches, batch normalization also has a (minor) regularizing effect

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Init-, Norm- & Regularization

Network Initialization, Normalization, Regularization

Summary and Outlook

Summary

- Parameter initialization for neural networks
- Static initialization vs. random initialization
- Vanishing gradient problem, exploding gradient problem
- Xavier initialization and He initialization
- Batch normalization as a technique to speed up model training

Outlook

- Regularisation of MLPs
- Definition of neural networks in software packages (e.g. PyTorch)

Source: OTH-AW, Electrical Engineering, Media and Computer Science, Fabian Brunner – Vorlesung Deep Learning, Init-, Norm- & Regularization