

# Deep Learning Reviews

Ao Zhang

July 3, 2021

## 1 Some Definitions

- **Supervised Learning:** Given  $\mathcal{D} = \{(x_i, y_i) : i = 1, 2, \dots, n\}$ , develop a program that predicts  $Y$  from  $X$ , or finds how  $Y$  depends on  $X$ .
- **Unsupervised Learning:** Given  $\mathcal{D} = \{x_i : i = 1, 2, \dots, n\}$ , develop a program that finds the structure in  $X$ , or generates an (new)  $X$  that conforms to the structure.
- **Reinforcement Learning:** Suppose there is an “environment” which can interact with an agent by changing the “state” and generating a reward for the agent. Develop an agent program that maximize some accumulated reward it receives.
- **Model:** A restricted family  $\mathcal{H}$  of hypotheses.

$$Y = f(X; \hat{\theta}) \quad (1)$$

- **Parametric Models:** Models have a fixed number of parameters, independent of sample size.
- **Non-Parametric Models:** The number of parameters increases with sample size. (usually not considered.)
- **Loss Functions:** A model is usually characterized by Loss Function  $\mathcal{L}(\theta)$  over the space  $\Theta$  of model parameters  $\theta$ . An example using Mean Square Error (MSE) is shown as below.

$$\hat{\theta} := \arg \min_{\theta \in \Theta} \mathcal{L}(\theta) = \|Y - X\theta\|^2 \quad (2)$$

- **Gradient Descent (GD):** Based on Equation 2,

$$\theta^{new} = \theta^{old} - \lambda \frac{d\mathcal{L}}{d\theta}(\theta^{old}) \quad (3)$$

$$= \theta^{old} + \lambda \frac{1}{N} \sum_{i=1}^N 2(y_i - \theta^{old} x_i) x_i \quad (4)$$

- **Stochastic Gradient Descent (SGD):** Based on Equation 2,

$$\theta^{new} = \theta^{old} + \lambda 2(y_i - \theta^{old} x_i) x_i \quad (5)$$

- **Mini-Batched SGD:** Based on Equation 2,

$$\theta^{new} = \theta^{old} + \lambda \frac{1}{|\mathcal{B}|} \sum_{(x,y) \in \mathcal{B}} 2(y - \theta^{old} x) x \quad (6)$$

## 2 Initialization

Good initialization of the model parameters will prevent exploding or vanishing gradients. A too-large initialization leads to exploding gradients and a too-small initialization leads to vanishing gradients.

Rules of appropriate initialization:

- The mean of the activations should be zero.
- The variance of the activations should stay the same across every layer.

Typical initialization methods:

- Random Norm (Gaussian) Distribution:
- **Xavier Initialization (most popular)**: Use a normal distribution with  $\sigma = \sqrt{\frac{2}{n_{in} + n_{out}}}$ ;  
Or use a uniform distribution with a range  $r = \sqrt{\frac{6}{n_{in} + n_{out}}}$
- Bias Initialization: often assign them to 0.

## 3 Activation Functions

- **Sigmoid Function**:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (7)$$

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)) \quad (8)$$

- **Softmax Function**:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}}, \quad x_i \in \mathbb{R}^K \quad (9)$$

- **Rectified Linear Unit (ReLU)**:

$$\text{ReLU}(x) = \max\{x, 0\} \quad (10)$$

- **Leaky ReLU**:

$$\text{ReLU}(x) = \max\{x, 0\} + \alpha \min\{x, 0\}, \quad \alpha > 0 \quad (11)$$

- **Hyperbolic Tangent Function**:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (12)$$

- **Softplus**:

$$\text{softplus}(x) = \log(1 + e^x) \quad (13)$$

## 4 Normalization

**Batch Normalization**:

$$\text{bn}_{k,j}(a) = \frac{a - \mathbb{E}(X^{(k)}[j])}{\text{VAR}(X^{(k)}[j])} \quad (14)$$

$$\mathbb{E}(X^{(k)}[j]) \approx \mu_{\mathcal{B},j}^{(k)} = \frac{1}{|\mathcal{B}|} \sum_{j \in \mathcal{B}} x^{(k)}[j] \quad (15)$$

$$\text{VAR}(X^{(k)}[j]) \approx s_{\mathcal{B},j}^{(k)} = \frac{1}{|\mathcal{B}|} \sum_{j \in \mathcal{B}} (x^{(k)}[j] - \frac{1}{|\mathcal{B}|})^2 \quad (16)$$

where,  $\mathcal{B}$  means batched data;  $X^{(k)}$  stands for input data to the  $k$ -th layer.

Reason why Batch Normalization works: Avoiding Internal Covariate Shift. Imagine the model is described in a Markov Chain mode,

$$X^{(0)} \leftarrow X^{(1)} \leftarrow X^{(2)} \leftarrow X^{(3)} \leftarrow \dots \leftarrow X^{(n)} \quad (17)$$

Therefore, the output probability of the  $k$ -th layer is a conditional distribution from previous prediction probability. As the input to the 1-st layer is already a batched data, the conditional distribution learning starts from the beginning. When using SGD to update parameters from  $\theta^{old}$  to  $\theta^{new}$ , the covariance of the  $k$ -th layer parameters will be shifted. This phenomenon is called “Internal Covariate Shift”. It can sometimes cause vanishing gradient. Using Batch Normalization can pretty much tackle the problem, as it re-scale the vanishing predictions to  $[-1, 1]$ .

Another reason why Batch Normalization works: It just be smoothing the loss landscape.

## 5 Regularization

First, we need to review what is **Overfitting** and what is **Underfitting**. Some definitions need to be introduced at the beginning.

- **In-Sample Error:** Also known as **training error**, notation  $E_{in}$ .
- **Out-Sample Error:** Also known as **testing error**, notation  $E_{out}$ .
- **Generalization Gap:** Notation  $E_{gen}$

$$E_{gen} = E_{out} - E_{in} \quad (18)$$

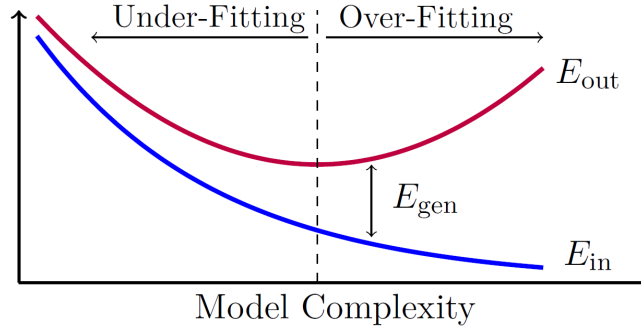


Figure 1: Overfitting and Underfitting

**Regularization:** It refers to techniques that reduce over-fitting when learning with complex models.

$$\mathcal{L}_{Reg}(\theta) = \mathcal{L} + \Omega(\theta) \quad (19)$$

Popular regularization methods can be listed as:

- **Data Augmentation**
- **Early Stop**
- **Dropout:**
- **L1 Regularizer:**  $\Omega(\theta) := \lambda_{Reg} \|\theta\|_1$
- **L2 Regularizer:**  $\Omega(\theta) := \lambda_{Reg} \|\theta\|_2^2$



Combined with Taylor Expansion.

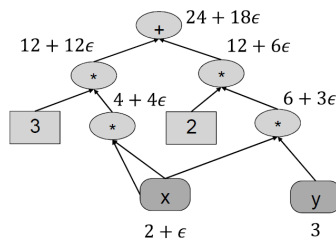
$$f(a + \epsilon b) = f(a) + \frac{f'(a)}{1!} \epsilon b + \frac{f''(a)}{2!} \epsilon^2 b^2 + \dots \quad (26)$$

$$= f(a) + \epsilon b f'(a) \quad (27)$$

Reason:  $\epsilon^2 = 0$ .

#### Forward-mode AutoDiff

- Example again:  $g(x, y) = 3x^2 + 2yx$  at  $x=2, y=3$



#### Reverse-mode AutoDiff

- Example again:  $g(x, y) = 3x^2 + 2yx$  at  $x=2, y=3$
- Result of forward pass in blue
- Setup for reverse mode

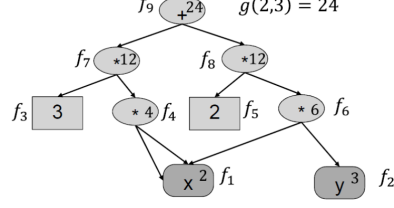


Figure 3: AutoDiff

## 8 Optimization Methods

- **Gradient Descent (GD)**: Assume Dataset  $\mathcal{D} = \{x_1, x_2, \dots, x_n\}$ , where  $n \in \mathbb{R}$ .

$$g_{t,t-1} = \partial_{\theta} \sum_{i=1}^n f(x_i, \theta_{t-1}) \quad (28)$$

$$\theta_t \leftarrow \theta_{t-1} - \eta \cdot g_{t,t-1} \quad (29)$$

- **Stochastic Gradient Descent (SGD)**

$$g_{t,t-1} = \partial_{\theta} f(x_i, \theta_{t-1}) \quad (30)$$

$$\theta_t \leftarrow \theta_{t-1} - \eta \cdot g_{t,t-1} \quad (31)$$

- **Mini-batched SGD**

$$g_{t,t-1} = \partial_{\theta} \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} f(x_i, \theta_{t-1}) \quad (32)$$

$$\theta_t \leftarrow \theta_{t-1} - \eta \cdot g_{t,t-1} \quad (33)$$

- **Momentum**

$$g_{t,t-1} = \partial_{\theta} \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} f(x_i, \theta_{t-1}) \quad (34)$$

$$v_t \leftarrow \beta v_{t-1} + g_{t,t-1} \quad (35)$$

$$\theta_t \leftarrow \theta_{t-1} - \eta \cdot v_t \quad (36)$$

- **AdaGrad**: The  $\odot$  means matrix-vector product.

$$g_{t,t-1} = \partial_{\theta} \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} f(x_i, \theta_{t-1}) \quad (37)$$

$$s_t \leftarrow \beta s_{t-1} + g_{t,t-1}^2 \quad (38)$$

$$\theta_t \leftarrow \theta_{t-1} - \frac{\eta}{\sqrt{s_t + \epsilon}} \odot g_{t,t-1} \quad (39)$$

- **Adadelta**

$$g_{t,t-1} = \partial_{\theta} \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} f(x_i, \theta_{t-1}) \quad (40)$$

$$g'_t = \frac{\sqrt{\Delta x_{t-1} + \epsilon}}{\sqrt{s_t + \epsilon}} \odot g_{t,t-1} \quad (41)$$

$$s_t \leftarrow \gamma s_{t-1} + (1 - \gamma) g_{t,t-1}^2 \quad (42)$$

$$\Delta x_t = \rho \Delta x_{t-1} + (1 - \rho) g_t'^2 \quad (43)$$

- **RMSProp**

$$g_{t,t-1} = \partial_{\theta} \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} f(x_i, \theta_{t-1}) \quad (44)$$

$$s_t \leftarrow \gamma s_{t-1} + (1 - \gamma) g_{t,t-1}^2 \quad (45)$$

$$\theta_t \leftarrow \theta_{t-1} - \frac{\eta}{\sqrt{s_t + \epsilon}} \odot g_{t,t-1} \quad (46)$$

- **Adam**

$$g_{t,t-1} = \partial_{\theta} \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} f(x_i, \theta_{t-1}) \quad (47)$$

$$v_t \leftarrow \beta_1 v_{t-1} + (1 - \beta_1) g_{t,t-1} \quad (48)$$

$$s_t \leftarrow \beta_2 s_{t-1} + (1 - \beta_2) g_{t,t-1}^2 \quad (49)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_1^t} \quad (50)$$

$$\hat{s}_t = \frac{s_t}{1 - \beta_2^t} \quad (51)$$

$$\theta_t \leftarrow \theta_{t-1} - \frac{\eta \hat{v}_t}{\sqrt{\hat{s}_t + \epsilon}} \quad (52)$$

- **Newton's Method:** Hessian Matrix too big for computation.

$$\beta^{t_1} \approx \beta^{t_0} - \mathbb{H}^{-1} \Delta_{\beta} J(\beta^{t_0}) \quad (53)$$

## 9 Convolutional Neural Networks (CNN)

### 9.1 Convolutional Layers

How to calculate output size regarding input size.

$$O = \frac{W - K + 2P}{S} + 1 \quad (54)$$

Where,  $O$  is output size;  $W$  input feature map size;  $K$  kernel size;  $P$  one-sided padding size;  $S$  stride.

### 9.2 “Deconvolution” or Fractionally-Strided Convolution

The term “Deconvolution” is actually fractionally-strided convolution. The stride of this type of convolution can be expressed as fractional numbers, such as  $\frac{1}{2}$ ,  $\frac{1}{3}$ , and etc. An example is shown as below.

Example:

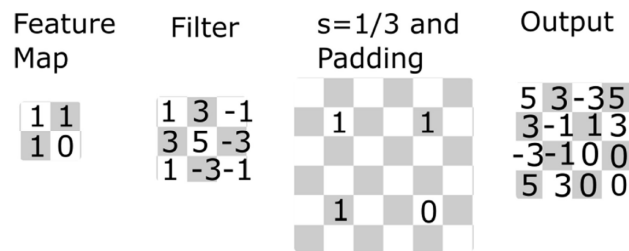


Figure 4: AutoDiff

### 9.3 Pooling Layers

- Median Pooling
- Mean Pooling
- Max Pooling
- $L^2$  Norm Pooling

### 9.4 Backbones

- Xception
- VGG16
- ResNet50
- InceptionV3
- MobileNet
- DenseNet
- NASNet

### 9.5 Neck Layers

- Feature Pyramid Networks (FPN)

### 9.6 Data Augmentation

- **Point Operations:** Change brightness, saturation, hue, gamma, correction, histogram equalization, etc.
- **Geometric Operations:** Shift, flip, rotate or shear.
- **Adding Noise:** Gaussian noise, etc.

## 10 Object Detection

### 10.1 One-Stage Detector

- YOLO
- SSD

## 10.2 Two-Stage Detector

- **Faster R-CNN: Region Proposal Networks + Classification Head;** It is an anchor-based method.

## 10.3 Metrics

- **Precision:**

$$\text{precision} = \frac{TP}{TP + F} \quad (55)$$

- **Recall:**

$$\text{recall} = \frac{TP}{TP + FN} \quad (56)$$

- **Average Precision (AP):** the area of precision-recall curves.

- **F1 Score:**

$$F1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = \frac{2TP}{TP + \frac{FN+FP}{2}} \quad (57)$$

- **Receiver Operating Characteristics (ROC) Curves:**

$$TPR = \frac{TP}{TP + FN} \quad (58)$$

$$FPR = \frac{FP}{TN + FP} \quad (59)$$

- **Confusion Matrix:** assume a classification task, the confusion matrix is computed on the validation set with each entry of the matrix indicating the count for each class predicted by the classifier. Example shown as below.

$$C = \begin{bmatrix} 21 & 2 & 7 \\ 10 & 11 & 9 \\ 3 & 1 & 26 \end{bmatrix} \quad (60)$$

Where, the perfect classifier would have results like,

$$C = \begin{bmatrix} 30 & 0 & 0 \\ 0 & 30 & 0 \\ 0 & 0 & 30 \end{bmatrix} \quad (61)$$

## 11 Image Segmentation

### 11.1 Instance Segmentation

### 11.2 Semantic Segmentation

- SegNet
- FCN
- U-Net

### 11.3 Metrics

## References