Deep Learning Reviews

Ao Zhang

July 3, 2021

1 Some Definitions

- **Supervised Learning**: Given $\mathcal{D} = \{(x_i, y_i) : i = 1, 2, ..., n\}$, develop a program that predicts Y from X, or finds how Y depends on X.
- **Unsupervised Learning**: Given $\mathcal{D} = \{x_i : i = 1, 2, ..., n\}$, 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.

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

- **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 Θ of model parameters θ. An example using Mean Square Error (MSE) is shown as below.

$$\hat{\theta} := \underset{\theta \in \Theta}{\arg\min}\, \mathcal{L}(\theta) = ||Y - X\theta||^2 \tag{2}$$

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

$$\begin{aligned} \theta^{new} &= \theta^{old} - \lambda \frac{d\mathcal{L}}{d\theta}(\theta^{old}) && (3) \\ &= \theta^{old} + \lambda \frac{1}{N} \sum_{i=1}^{N} 2(y_i - \theta^{old} x_i) x_i && (4) \end{aligned}$$

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

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

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

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

2 Initalization

Good intialization 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 intialization:

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

Typical intialization methods:

- Ramdom Norm (Gaussian) Distribution:
- **Xavier Intialization (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**:

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

$$\sigma'(x) = \sigma(x)(1-\sigma(x)) \tag{8}$$

- **Softmax Function**:

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

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

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

- **Leaky ReLU**:

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

- **Hyperbolic Tangent Function**:

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

- **Softplus**:

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

4 Normalization

Batch Normalization:

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

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

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

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 Marcov Chain mode,

$$X^{(0)} \leftarrow X^{(1)} \leftarrow X^{(2)} \leftarrow X^{(3)} \leftarrow ... \leftarrow X^{(n)} \tag{17}$$

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 θ^{old} to θ^{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}

$$E_{gen} = E_{out} - E_{in} \tag{18}$$

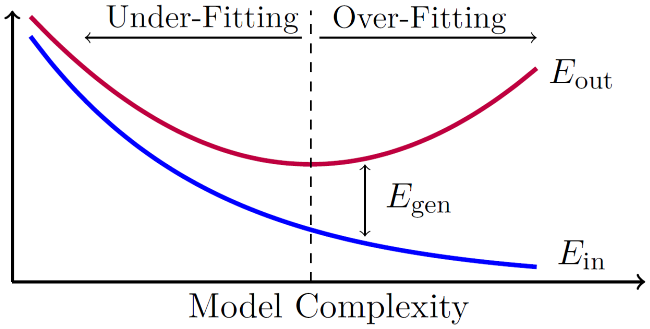


Figure 1: Overfitting and Underfitting

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

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

Popular regularziation methods can be listed as:

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

Algorithm 1: How to do dropout

```
Input: mini-batch $\mathcal{B}$
for $X \in \mathcal{B}$ do
    for earch layer, delete nodes with probability $1 - p$;
    derive gradient with backpropagation and set the gradient of droped nodes to 0;
end
average the gradients derived above and update the parameters.
```

6 Loss Functions

- **Mean Square Error**:

$$\hat{\theta} = \arg\min_{\theta}(Y - X\theta)^2 \tag{20}$$

- **Cross Entropy**: Minimize Cross Entropy = Maximize Likelihood

$$\begin{aligned} CE(\tilde{p}; p) &= -\sum_{y \in Y} \tilde{p}(y) \log p(y) && (21) \\ &= -\mathbb{1}_{y_i=1} \log p_{Y|X}(1|x_i) - \mathbb{1}_{y_i=0} \log p_{Y|X}(0|x_i) && (22) \end{aligned}$$

- **Focal Loss function**
- **IoU Loss function**
- **Dice Loss function**

7 Practical Backpropagation Method

7.1 Symbolic Differentiation

This is basically the method TensorFlow uses, a.k.a computational graph.

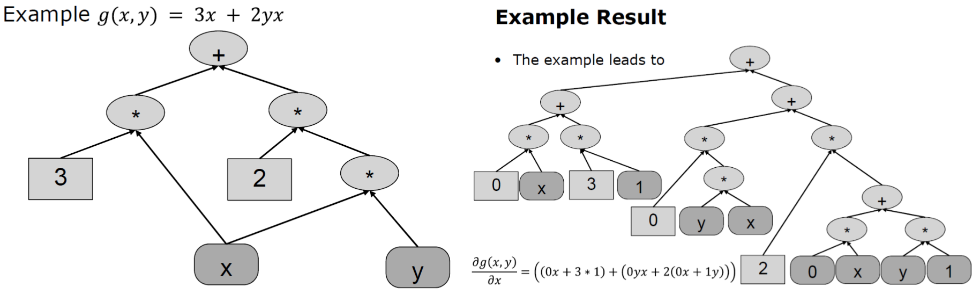

Example $g(x,y) = 3x + 2yx$
+
*
*
*
3
2
x
y
Example Result
• The example leads to
+
+
+
*
*
*
*
*
+
*
*
0
x
3
1
0
y
x
$\frac{\partial g(x,y)}{\partial x} = \left((0x + 3 * 1) + \left(0yx + 2(0x + 1y)\right)\right)$
2
0
x
y
1

Figure 2: Symbolic Differentiation

7.2 Numerical Differentiation

$$\frac{\partial f(x)}{\partial x} \approx \frac{f(x+\epsilon) - f(x)}{\epsilon} \tag{23}$$

7.3 AutoDiff

Dual Numbers are similar to complex numbers but replace the imaginary part with infinitesimal number $a + \epsilon b$ such that $\epsilon \neq 0$ but $\epsilon^2 = 0$.

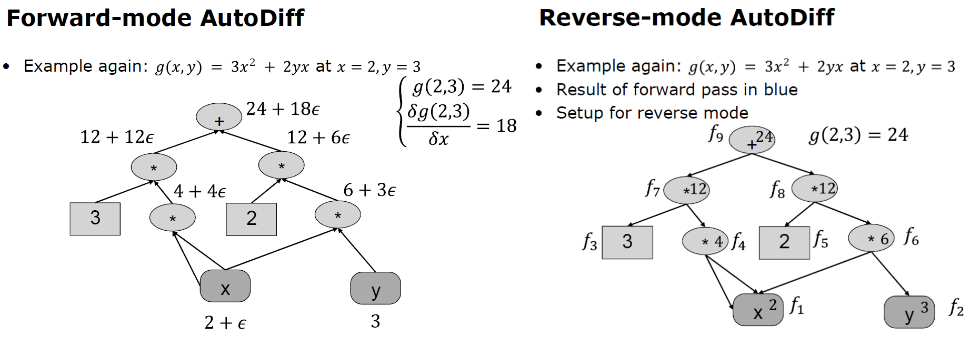
$$(a + \epsilon b) + (c + \epsilon d) = (a + c) + \epsilon(b + d) \tag{24}$$

$$(a + \epsilon b)(c + \epsilon d) = ac + \epsilon(bc + ad) \tag{25}$$

Combined with Taylor Expansion.

$$
\begin{aligned}
f(a+\epsilon b) &= f(a) + \frac{f'(a)}{1!}\epsilon b + \frac{f''(a)}{2!}\epsilon^2 b^2 + ... &(26)\\
&= f(a) + \epsilon b f'(a) &(27)
\end{aligned}
$$

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 $\mathcal{D} = \{x_1, x_2, ..., x_n\}$, where $n \in \mathbb{R}$.

$$
\begin{aligned}
g_{t,t-1} &= \partial_\theta \sum_{i=1}^{n} f(x_i, \theta_{t-1}) &(28)\\
\theta_t &\leftarrow \theta_{t-1} - \eta \cdot g_{t,t-1} &(29)
\end{aligned}
$$

- **Stochastic Gradient Descent (SGD)**

$$
\begin{aligned}
g_{t,t-1} &= \partial_\theta f(x_i, \theta_{t-1}) &(30)\\
\theta_t &\leftarrow \theta_{t-1} - \eta \cdot g_{t,t-1} &(31)
\end{aligned}
$$

- **Mini-batched SGD**

$$
\begin{aligned}
g_{t,t-1} &= \partial_\theta \frac{1}{|\mathcal{B}|} \sum_{i\in\mathcal{B}} f(x_i, \theta_{t-1}) &(32)\\
\theta_t &\leftarrow \theta_{t-1} - \eta \cdot g_{t,t-1} &(33)
\end{aligned}
$$

- **Momentum**

$$
\begin{aligned}
g_{t,t-1} &= \partial_\theta \frac{1}{|\mathcal{B}|} \sum_{i\in\mathcal{B}} f(x_i, \theta_{t-1}) &(34)\\
v_t &\leftarrow \beta v_{t-1} + g_{t,t-1} &(35)\\
\theta_t &\leftarrow \theta_{t-1} - \eta \cdot v_t &(36)
\end{aligned}
$$

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

$$
\begin{aligned}
g_{t,t-1} &= \partial_\theta \frac{1}{|\mathcal{B}|} \sum_{i\in\mathcal{B}} f(x_i, \theta_{t-1}) &(37)\\
s_t &\leftarrow \beta s_{t-1} + g_{t,t-1}^2 &(38)\\
\theta_t &\leftarrow \theta_{t-1} - \frac{\eta}{\sqrt{s_t + \epsilon}} \odot g_{t,t-1} &(39)
\end{aligned}
$$

- **Adadelta**

$$
\begin{aligned}
g_{t,t-1} &= \partial_\theta \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} f(x_i, \theta_{t-1}) \qquad (40)\\
g'_t &= \frac{\sqrt{\Delta x_{t-1} + \epsilon}}{\sqrt{s_t + \epsilon}} \odot g_{t,t-1} \qquad (41)\\
s_t &\leftarrow \gamma s_{t-1} + (1-\gamma) g^2_{t,t-1} \qquad (42)\\
\Delta x_t &= \rho \Delta x_{t-1} + (1-\rho) {g'_t}^2 \qquad (43)
\end{aligned}
$$

- **RMSProp**

$$
\begin{aligned}
g_{t,t-1} &= \partial_\theta \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} f(x_i, \theta_{t-1}) \qquad (44)\\
s_t &\leftarrow \gamma s_{t-1} + (1-\gamma) g^2_{t,t-1} \qquad (45)\\
\theta_t &\leftarrow \theta_{t-1} - \frac{\eta}{\sqrt{s_t + \epsilon}} \odot g_{t,t-1} \qquad (46)
\end{aligned}
$$

- **Adam**

$$
\begin{aligned}
g_{t,t-1} &= \partial_\theta \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} f(x_i, \theta_{t-1}) \qquad (47)\\
v_t &\leftarrow \beta_1 v_{t-1} + (1-\beta_1) g_{t,t-1} \qquad (48)\\
s_t &\leftarrow \beta_2 s_{t-1} + (1-\beta_2) g^2_{t,t-1} \qquad (49)\\
\hat{v}_t &= \frac{v_t}{1-\beta_1^t} \qquad (50)\\
\hat{s}_t &= \frac{s_t}{1-\beta_2^t} \qquad (51)\\
\theta_t &\leftarrow \theta_{t-1} - \frac{\eta \hat{v}_t}{\sqrt{\hat{s}_t} + \epsilon} \qquad (52)
\end{aligned}
$$

- **Newton's Method**:

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

9 Convolutional Neural Networks (CNN)

9.1 Convolutional Layers

How to calculate output size regarding input size.

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

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.

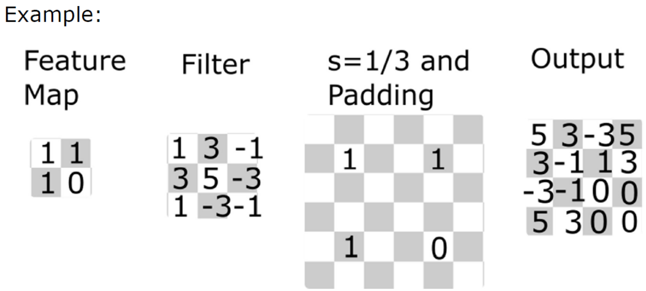


Figure 4: AutoDiff

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 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 Backbone

10.2 Neck Layers

10.3 Detection Head

10.4 Loss Functions

10.5 Matrics

- **Precision**:
$$\text{precision} = \frac{TP}{TP +_F P} \tag{55}$$
- **Recall**:
$$\text{recall} = \frac{TP}{TP + FN} \tag{56}$$
- **Average Precision (AP)**: the area of precision-recall curves.
- **F1 Score**:
$$F1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = \frac{TP}{TP + \frac{FN+FP}{2}} \tag{57}$$
- **Receiver Operating Characteristics (ROC) Curves**:
$$TPR = \frac{TP}{TP + FN} \tag{58}$$
$$FPR = \frac{FP}{TN + FP} \tag{59}$$
- **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.
$$C = \begin{bmatrix} 21 & 2 & 7 \\ 10 & 11 & 9 \\ 3 & 1 & 26 \end{bmatrix} \tag{60}$$
Where, the perfect classifier would have results like,
$$C = \begin{bmatrix} 30 & 0 & 0 \\ 0 & 30 & 0 \\ 0 & 0 & 30 \end{bmatrix} \tag{61}$$

11 Image Segmentation

11.1 Backbone

11.2 Neck Layers

11.3 Detection Head

11.4 Loss Functions

11.5 Matrics

References