Deep Learning Reviews

Ao Zhang

July 2, 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.

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

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

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

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

- **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 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**:

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

3 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.

4 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}

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

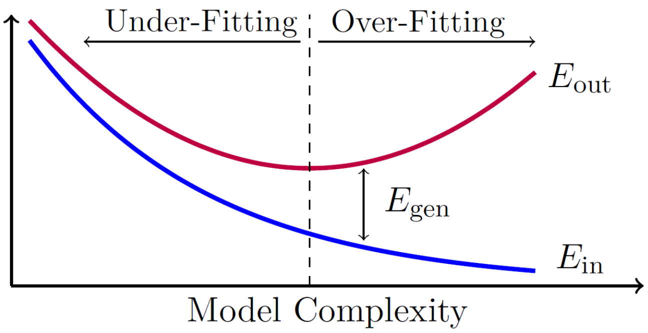

Figure 1: Overfitting and Underfitting

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**:

Algorithm 1: How to do dropout

```
Input: mini-batch B
for X ∈ 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.
```

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

5 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**

6 Backpropagation Method

7 Optimization Methods

8 Convolutional Neural Networks (CNN)

8.1 Convolutional Layers

8.2 Pooling Layers

8.3 Zero-paddings

9 Object Detection

9.1 Backbone

9.2 Neck Layers

9.3 Detection Head

9.4 Loss Functions

9.5 Matrics

10 Image Segmentation

10.1 Backbone

10.2 Neck Layers

10.3 Detection Head

10.4 Loss Functions

10.5 Matrics

References