Math 690/Convolution neural network

Sciber: Guangshen Ma

11/15/2017

1 Review and guideline

On previous class, we mention the following concepts:

- Classifier(supervised):
- AE:Autoencoder(unsupervised)

We talk about the fundamental analysis in machine learning including approximation as well as generalization and we will focus on the Autoencoder method and stability analysis of CNN(convolution neural network). Autoencoder is used for unsupervised learning(without label) and its main goal(in short) is for reconstruction. The stability analysis is to discuss the relationship between the weight parameters and stability of the CNN method. The main topics for this class is in the following:

- Autoencoder and its architecture
- Stability analysis of Convolution neural network
- Approximation theory
- Generalization analysis of the supervised learning model

2 Autoencoder

2.1 Basic architecture

For the goal of supervised learning(given original data and label), we want to find the function $f(x) = y$ with the probability $p(y) = 1$(We want the model to learn the distribution). For unsupervised learning, we are only given $\overrightarrow{x}$, and we want to learn the distribution of $f(x)$ itself by only given the original data, without label. For the Autoencoder, we are given $\overrightarrow{x}$ and the process is shown in the following:

- Encode: reduce the dimension(we can think it as the process of extracting feature)

- Decode: increase the dimension and reconstruct the data

We can consider the whole process as encoding and decoding or mapping. The final goal is develop an architecture that can extract important features and use these features to represent the data(reconstruction).
The general architecture is shown in the following:

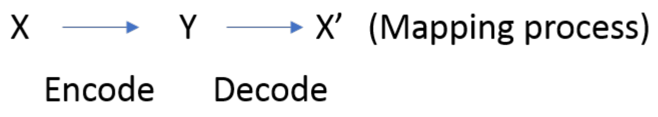

Figure 1: The autoencoder process

More specifically, this is the architecture of autoencoder method:

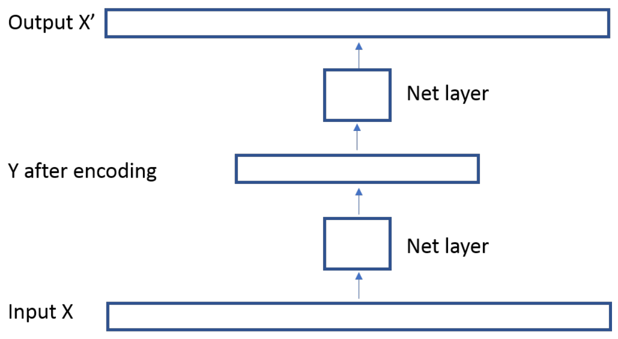

Figure 2: The autoencoder process

For the above architecture network, we generate the parameter basis for encoding and decoding process with E_θ and D_ϕ respectively. The representation for mapping based on the network parameter for each layer is:

$$E_\theta : \mathbb{R}^{dx} \rightarrow \mathbb{R}^{dy};\ D_\phi : \mathbb{R}^{dy} \rightarrow \mathbb{R}^{dx}$$

2.2 Loss function analysis

The basic question for this model is how to analyze the training loss.Suppose we have training data ${x_i}_{i=1}^{n=t_r}$:

$$x' = D_\phi(E_\theta(x)) = D_\phi E_\theta(x)$$

Where x' is the output for the network model, as shown before. Then we define the loss function with square L_2 norm to measure the difference between training data x_i and output x_c of the model;

$$Loss = \sum_{i=1}^{n_{tr}} ||x_i - D_\phi E_\theta(x_c)||_2^2$$

Remark: The denoising problem of Autoencoder.
Suppose we have the noisy data(Gaussian noise) for the input $\overline{x}$:

$$\overline{x} = x_i + \sigma_i$$

The loss function for the noisy data is:

$$Loss_{noise} = \sum_i ||x_i - D_\phi E_\phi(x_i + \sigma_x)||_2^2$$

The main question here may be why we need to analyze the noisy problem for Autoencoder. The main goal is to improve the generalization (prevent over-fitting) of the whole model by adding noise for input data, so that the model can not only adapt to the training data itself, but also the data around, i.e the model can classify the data points and the surrounding data points(That is the goal of generalization).[ref:Bishop 96']

The mathematical analysis of the noisy method(as mentioned above):
Suppose $f(\overline{x})$ is not changing frequently around $x = \overline{x}$, then we have the Taylor expansion:

$$f(x_i + \sigma_i) = f(x_i) + f'(x_i)\sigma_i + \frac{1}{2}f''(x_i)\sigma_i^2 + o(|\sigma_i|^n)$$

Where n denotes the order of higher order term and σ_i is gaussian noise.For the term $f'(x_i)\sigma_i + \frac{1}{2}f''(x_i)\sigma_i^2 + o(|\sigma_i|^n)$, we can consider it as a regularization term for the model(can consider as add a noise term), and so autoencoder by adding noise can be considered as a way to improve the performance of regularization(The reason we add noise for the input sometimes).

2.3 Approximation theory

First, let us start with an simple example in supervised learning to discuss the approximation theory. we want to find the function:

$$y = f(x)$$

We want to find $f(\theta)$(θ is the parameter in the model) to approximate f. We want to talk about the case $y \in \mathbb{R}$(y being in one dimension). And the target is actually mapping from d dimension to one dimension, as shown in the following:

$$R^d \to R,\ d = d_x$$

We can consider the above case as a simple setting for supervised learning(binary classification problem). We want to talk about the shallow network(one hidden layer) for above case.

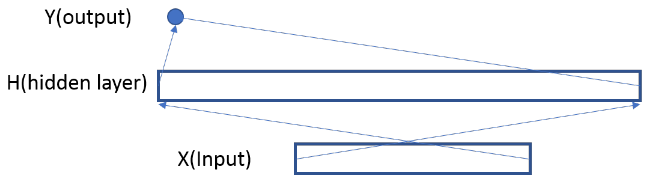

Figure 3: The supervised learning network with one hidden layer

For the approximation analysis, we want to find the best parameters θ that can approximate the function f. The goal is:

$$||f_\theta - f|| < C$$

Where C is the target constant that we want to make it bounded with the conditions from parameters.
Suppose f_m is the approximated function with m neurons, then we have:

$$||f_m - f|| < \epsilon$$

Where m is large enough and ϵ is a small constant.
Based on the history of machine learning, we have two traditional results (Only refer the results without details and proving here):

- ϵ can be arbitrarily small and m is exponentially large to approximate any continues f [ref:Hormit'89,Gylenko'89]
- $||f_m - f||_2^2 \leq \frac{c_f}{m}$ where c_f is the Fourier criterion and m is the number of neurons [ref:Barron 93',94].

Then the question lies in how we determine the coefficients c_f and m.
More specifically, we have the representation for f:

$$c_f = \int_{R^d} ||w|||f'(w)|dw < +\infty$$

Where the barron function f is

$$f: \ B \rightarrow \mathbb{R} \ \ B: \ unit \ ball(for \ this \ case) \ in \ \mathbb{R}^d, \ B = \{x, \ ||x|| \leq 1\}$$

Where f has the Fourier transform $\widehat{f}(w)$ and we have the inverse transform:

$$f(x) = f(0) + \int_{R^d} (e^{iw^T x} - 1)\widehat{f}(w)dw$$

Also, $c_f < +\infty$. Based on the above theory, we will continue by discussion of approximation theory in supervised learning.
Recall the network structure ($x \in R^d, h \in R^m, y \in R^1$). The first output after the hidden layer:

$$h_R(x) = \sigma(\alpha_k^T x + b_k)$$

Where $\alpha_k \in \mathbb{R}^d$, $b_k \in \mathbb{R}$, $k = 1, 2, \cdots m$. We also have the output y:

$$y(h) = \sum_{k=1}^{m} c_k h_k + c_0$$

Then we have (m is the number of neurons):

$$f_m(x) = c_0 + \sum_{k=1}^{m} c_k \sigma(\alpha_k^T x + b_k)$$

Where the σ is the sigmoid function, c_k is the weight on the second layer. We then come up with a theorem:

Theorem 2.1 *Suppose f is Barron function with $c_f < +\infty$, then $\forall \mu$, we have $\forall \sigma$ sigmoid function, $\forall m \geq 1$, $\exists \{\alpha_k, b_k, c_k\}_k$ s.t f can be approximated by f_n of the form that satisfies $\int_B |f(x) - f_m(x)|^2 d\mu(x) \leq \frac{(2c_f)^2}{m}$*

However, the question lies in how big is the coefficient c_k. Particularly, we have [reference from paper without details here]:

$$c_0 = f(0), \; \sum_{k=1}^{m} |c_k| \leq 2c_f$$

2.4 Generalization analysis

We want f to be generalizable given that the model can be applied to other data rather than only the training data. We first introduce the generalization error:

$$L_R(f_\theta) = \mathbb{E}_{x \sim p_x} |f_\theta - f(x)|^2$$

Where L_R is the expected loss or population error. We also have the training loss:

$$\widehat{L}_R[f_\theta] = \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} |f_\theta(x) - f(x)|^2$$

We want to find the gap between expectation loss and training loss.

$$gap = L_R(f_\theta) - \widehat{L}_R[f_\theta]$$

One question for this problem is that can we use simple concentration argument for this problem, as we discussed before, so that we can easily control the gap.

That is to say can we use concentration method to bound or control the loss.

However, they(The two loss) cannot be considered as independent since they are not independent for the training data so that not concentration methods can be used here(they cannot be considered independent), as shown in the following. Suppose $\widehat{\theta}$ is trained from $x_{tr} = \{x_i\}_{i=1}^{n_{tr}}$, and we have:

$$\widehat{R_n}[f(\widehat{\theta})] = \frac{1}{n}\sum_{i=1}^{n} |f_{\widehat{\theta}}(x_i) - f(x_i)|^2$$

The $\widehat{\theta}$ from the $f_{\widehat{\theta}}(x_i)$ is actually depending on x_{tr}. That is the reason we cannot use concentration strategy here.

3 Note

- ReLu and sigmoid functions are not fully discussed for this note.
- Some details or build-in theory are referred from the papers and the details (such as prove or demonstration) are not provided here.