A Study of Distribution Free Non-parametric Regression

Ayush Chaurasia
2016MT10617

Kumar Prithvi Mishra
2016MT10618

Prof. Sivananthan Sampath
Supervisor

BTP mid-term report

Abstract

The paper first talks about "Why is non-parametric regression important?" and shows the mathematical formulation of parametric methods and then the need of non-parametric estimation. Then we dive into one specific type of non-parametric method specifically deep CNNs, how are they formulated and see how well do they approximate a function and other nuances related to it.

Contents

1 Why is Non-Parametric Regression Important?

1.1 Regression Analysis and L_2 Risk

In regression analysis one considers a random vector (X, Y), where X is $\mathbb{R}^d$-valued and Y is $\mathbb{R}$-valued, and one is interested how the value of variable Y depends on the value of vector X. This means that one wants to find a (measurable) function $f : \mathbb{R}^d \to \mathbb{R}$, such that $f(X)$ should be a close approximation to Y in some sense, i.e. making $|f(X) - Y|$ "small". We can resolve this problem by introducing L_2 *risk or mean squared error of* f,

$$\mathbb{E}|f(X) - Y|^2,$$

and requiring it to be as small as possible.

There are two reasons for considering the L_2 risk. First, this simplifies the mathematical treatment of the whole problem (the function which minimizes the L_2 risk can be derived explicitly). Second, trying to minimize the L_2 risk leads naturally to estimates which can be computed rapidly.

So we are interested in a (measurable) function $m^* : \mathbb{R}^d \to \mathbb{R}$ such that

$$\mathbb{E}|m^*(X) - Y|^2 = \min_{f:\mathbb{R}^d \to \mathbb{R}} \mathbb{E}|f(X) - Y|^2.$$

Such a function can be obtained explicitly as follows. Let

$$m(x) = \mathbb{E}\{Y|X = x\}$$

be the *regression function*. We will show that the regression function minimizes the L_2 risk. Indeed, for an arbitrary $f : \mathbb{R}^d \to \mathbb{R}$, one has

$$\mathbb{E}|f(X) - Y|^2 = \mathbb{E}|f(X) - m(X) + m(X) - Y|^2$$
$$\mathbb{E}|f(X) - Y|^2 = \mathbb{E}|f(X) - m(X)|^2 + \mathbb{E}|m(X) - Y|^2,$$

where we have used

$$\begin{aligned}
&\mathbb{E}\{(f(X) - m(X))(m(X) - Y)\} \\
&= \mathbb{E}\{\mathbb{E}\{(f(X) - m(X))(m(X) - Y)|X\}\} && \because \mathbb{E}\{\mathbb{E}\{\mathbb{Y}|\mathbb{X}\}\} = \mathbb{E}\{\mathbb{Y}\} \\
&\quad = \mathbb{E}\{(f(X)-m(X))\mathbb{E}\{(m(X)-Y)|X\}\} && \because \mathbb{E}\{m(X)|X\} = \mathbb{E}\{m(X)\} \\
&\quad = \mathbb{E}\{(f(X) - m(X))(m(X) - m(X))\} \\
&= 0.
\end{aligned}$$

Hence,

$$\mathbb{E}|f(X) - Y|^2 = \int_{\mathbb{R}^d} |f(x) - m(x)|^2 \mu(dx) + \mathbb{E}|m(X) - Y|^2 \tag{1}$$

The second term is fixed, so it doesn't play any role in optimizing L_2 risk. Hence, we only reduce the first term, which is the L_2 error. From above, we can see that $f(x) = \mathbb{E}\{Y|X = x\}$ is the optimal minimizer for the L_2 risk, but often we don't have the distribution of $f_{y/x}(y/x)$. Hence, we now try to estimate the distribution in the next section.

1.2 Regression Function Estimation and L_2 Error

Let $\mathbb{D}_n$ be the set of data defined by $\mathbb{D}_n = (X_1, Y_1), ..., (X_n, Y_n)$. In the regression function estimation problem one wants to use the data $\mathbb{D}_n$ in order to construct an estimate $m_n : \mathbb{R}^d \to \mathbb{R}$ of the regression function m. In general, the L_p error, $\int_{\mathbb{C}} |m_n(x) - m(x)|^p dx$, where the integration is with respect to the Lebesgue measure, $\mathbb{C}$ is a fixed subset of $\mathbb{R}^d$, and $p \geqslant 1$ is arbitrary (often $p = 2$ is used).
Recall that the main goal was to find a function f such that the L_2 risk $\mathbb{E}|f(X) - Y|^2$ is small. The minimal value of this L_2 risk is $\mathbb{E}|m(X) - Y|^2$, and it is achieved by the regression function m. Similarly to eq. (1), L_2 risk $\mathbb{E}|m_n(X) - Y|^2|D_n$ of an estimate m_n satisfies

$$\mathbb{E}\{|m_n(X) - Y|^2|D_n\} = \int_{\mathbb{R}^d} |m_n(x) - m(x)|^2 \mu(dx) + \mathbb{E}|m(X) - Y|^2. \tag{2}$$

Thus the L_2 risk of an estimate m_n is close to the optimal value if and only if the L_2 error

$$\int_{\mathbb{R}^d} |m_n(x) - m(x)|^2 \mu(dx) \tag{3}$$

is close to zero. Therefore we will use the L_2 error (3) in order to measure the quality of an estimate and we will study estimates for which this L_2 error is small.

1.3 Application to Pattern Recognition

Here we try to model pattern recognition problem also as a regression problem. In pattern recognition, Y takes only finitely many values. For simplicity assume that Y takes two values, say 0 and 1. The goal is to find a function $g^* : \mathbb{R}^d \to \{0, 1\}$ which minimizes the probability of $g^*(X) \neq Y$,i.e., to find a function g^* such that

$$\mathbb{P}\{g^*(X) \neq Y\} = \min_{g:\mathbb{R}^d \to \{0,1\}} \mathbb{P}\{g(x) \neq Y\}, \tag{4}$$

where g^* is called the Bayes decision function, and $\mathbb{P}\{g(X) \neq Y\}$ is the probability of misclassification.

1.3.1 Lemma

$$g^*(x) = \begin{cases} 1 & \text{if } \mathbb{P}\{Y = 1|X = x\} \geq 1/2, \\ 0 & \text{if } \mathbb{P}\{Y = 1|X = x\} < 1/2 \end{cases} \tag{5}$$

is the Bayes decision function, i.e., g^* satisfies (4).
PROOF. Let $g : \mathbb{R}^d \to \{0, 1\}$ be an arbitrary (measurable) function.

$\mathbb{P}\{g(x) \neq Y|X = x\} - \mathbb{P}\{g^*(X) \neq Y|X = x\}$
$= \mathbb{P}\{Y = g^*(x)|X = x\} - \mathbb{P}\{Y = g(X)|X = x\} \geq 0$

because

$$\mathbb{P}\{Y = g^*|X = x\} = \max\{\mathbb{P}\{Y = 0|X = x\}, \mathbb{P}\{Y = 1|X = x\}\}$$

by the definition of g^*. This proves

$$\mathbb{P}\{g^*(X) \neq Y | X = x\} \leq \mathbb{P}\{g(X) \neq Y | X = x\}$$

for all $x \in \mathbb{R}^d$, which implies

$$\begin{aligned}\mathbb{P}\{g^*(X) \neq Y\} &= \int \mathbb{P}\{g^*(X) \neq Y | X = x\}\mu(dx) \\ &\leq \int \mathbb{P}\{g(X) \neq Y | X = x\}\mu(dx) \\ &= \mathbb{P}\{g(X) \neq Y\}\end{aligned}$$

$\mathbb{P}\{Y = 1|X = x\}$ and $\mathbb{P}\{Y = 0|X = x\}$ are the so-called a posterior probabilities. Observe that $\mathbb{P}\{Y = 1|X = x\} = \mathbb{E}\{Y|X = x\} = m(X)$ A natural approach is to estimate the regression function m by an estimate m_n using data D_n and then to use a so-called plug-in estimate

$$g_n^*(x) = \begin{cases} 1 & \text{if } m_n \geq 1/2, \\ 0 & \text{if } m_n < 1/2 \end{cases} \tag{6}$$

The next theorem implies that if m_n is close to the real regression function m, then the error probability of decision g_n is near to the error probability of the optimal decision g^*.

1.3.2 Theorem

Let $\hat{m} : \mathbb{R}^d \to \mathbb{R}$ be a fixed function and define the plug-in decision $\hat{g}$ by

$$\hat{g}^*(x) = \begin{cases} 1 & \text{if } \hat{m} \geq 1/2, \\ 0 & \text{if } \hat{m} < 1/2 \end{cases}$$

Then

$$\begin{aligned}0 &\leq \mathbb{P}\{\hat{g}(X) \neq Y\} - \mathbb{P}\{g^*(X) \neq Y\} \\ &\leq 2\int_{\mathbb{R}^d} |\hat{m} - m(x)|\mu(dx) \\ &\leq 2\left(\int_{\mathbb{R}^d} |\hat{m} - m(x)|^2\mu(dx)\right)^{1/2}\end{aligned}$$

Note that the second inequality in particular is not tight. Therefore pattern recognition is easier than regression estimation. It follows from Theorem, that the error probability of the plug-in decision g_n defined above satisfies

$$\begin{aligned}0 &\leq \mathbb{P}\{g_n(X) \neq Y|D_n\} - \mathbb{P}\{g^*(X) \neq Y\} \\ &\leq 2\int_{\mathbb{R}^d} |m_n(x) - m(x)|\mu(dx) \\ &\leq 2\left(\int_{\mathbb{R}^d} |m_n(x) - m(x)|^2\mu(dx)\right)^{1/2}.\end{aligned}$$

Thus estimates m_n with small L_2 error automatically lead to estimates g_n with small misclassification probability. Observe however, that for (6) to be a good approximation of (5) it is only important that $m_n(x)$ should be on the same side of the decision boundary (1/2 in this case) as $m(x)$. Nevertheless, one often constructs estimates by minimizing the L_2 risk $\mathbb{E}\{|m_n(X) - Y|^2|D_n\}$ and using the plug-in rule (6), because trying to minimize the L_2 risk leads to estimates which can be computed efficiently. This can be generalized to the case where Y takes $M \geq 2$ distinct values, without loss of generality $1, .., M$. The goal is to find a function $g^* : \mathbb{R}^d \to \{1, ..., M\}$ such that

$$\mathbb{P}\{g^*(X) \neq Y\} = \min_{g:\mathbb{R}^d \to \{1,...,M\}} \mathbb{P}\{g(X) \neq Y\}, \tag{7}$$

where $g^*(x)$ is called the Bayes decision function. It can be computed using the posterior probabilities $\mathbb{P}\{Y = k|X = x\}(k \in \{1, ..., M\})$:

$$g^*(x) = arg \max_{1 \leq k \leq M} \mathbb{P}\{Y = k|X = x\} \tag{8}$$

The a posteriori probabilities are the regression functions

$$\mathbb{P}\{Y = k|X = x\} = \mathbb{E}\{I_{\{Y=k\}}|X = x\} = m^{(k)}(x).$$

Given data D_n, estimates $m_n^{(k)}$ of $m^{(k)}$ can be constructed from the data set

$$D_n^{(k)} = \{(X_1, I_{Y_1=k}), ..., (X_n, I_{Y_n=k})\},$$

and one can use a plug-in estimate

$$g_n(x) = arg \max_{1 \geq k \geq M} m_n^{(k)}(x) \tag{9}$$

to estimate g^*. If the estimates $m_n^{(k)}$ are close to the posterior probabilities, then again the error of the plug-in estimate (9) is close to the optimal error.

2 Universality of Deep Convolutional Neural Networks

In particular, for deep CNNs having convolutional structures without fully connected layers, it is unknown which kinds of functions can be approximated. The paper clarifies the difference between deep CNNs and neural network in general, defines the CNN and provides a rigorous mathematical theory to see which kind of functions can be approximated and with what error rate.

2.1 Notations and Concepts

The deep CNNs considered here have a few essential ingredients:

2.1.1 Activation function

A rectified linear unit (ReLU) defined as a uni-variate nonlinear function σ given by

$$\sigma(u) = (u)_+ = max\{u, 0\}, \quad u \in \mathbb{R}$$

2.1.2 Convolutional Filter Masks

A sequence of convolutional filter masks $\mathbf{w} = \{w^{(j)}\}_j$ induce a sparse convolutional structure. A filter mask $w = (w_k)_{k=-\infty}^{\infty}$ means a sequence of filter coefficients. A fixed integer filter length $s \geqslant 2$ is used to decide non-zero weights namely, $w_k^{(j)} \neq 0$ only for $0 \leq k \leq s$ and hence controls the sparsity. The convolution of such a filter mask w with another sequence $v = (v_0, ..., v_D)$ is a sequence $w \star v$ given by $(w \star v)_i = \sigma_{k=0}^{D} w_{i-k} v_k$. This convolution leads to a $(D+s) \times D$ convolutional matrix T which has constant valued diagonals.

$$T = \begin{bmatrix} w_0 & 0 & 0 & 0 & \cdot & 0 \\ w_1 & w_0 & 0 & 0 & \cdot & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ w_s & w_{s-1} & \cdots & w_0 & 0 \cdots & 0 \\ 0 & w_s & \cdots & w_1 & w_0 \cdots & 0 \\ \vdots & \ddots & \ddots \ddots & \ddots & \ddots \ddots & \vdots \\ \cdots & \cdots & 0 & w_s & \cdots & w_0 \\ \cdots & \cdots & \cdots & 0 & w_s \cdots & w_1 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots\cdots & \cdots 0 & w_s & w_{s-1} \\ 0 & \cdots & \cdots & \cdots & \cdots 0 & w_s \end{bmatrix}$$

2.1.3 Convolutional Neural Networks

A deep CNN is a sequence of J vectors $h^{(j)}(x)$ of functions on $\mathbb{R}^d$ with $h^{(0)}(x) = x$ given iteratively by:

$$h^{(j)}(x) = \sigma(T^j h^{j-1}(x) - b^{(j)}), \quad j = 1, 2, ..., J$$

where $T^{(j)} = (w_{i-k}^{(j)})$ is a $d_j \times d_{j-1}$ convolutional matrix, σ acts on vectors component-wise, and $\boldsymbol{b}$ is a sequence of bias vectors $b^{(j)}$ where $b^{(j)}$ for $j = 1, 2, ..., J-1$ is of the form:

$$\begin{bmatrix} \mathrm{b}_1 & ...b_s & b_{s+1} & ...bs+1 & b_{d_{j-s+1}} & ...b_{d_j} \end{bmatrix}$$

i.e. $d_j - 2s$ repeated components in the middle and s components on each side. The last layer bias vector $b^{(J)}$ is of the form:

$$\begin{bmatrix} \mathrm{b}_1 & b_2 & ...b_{d_J-1} & b_{d_J} \end{bmatrix}$$

2.1.4 Number of parameters in CNN

For last layer, d_J parameters for bias, d_J parameters for $c \in \mathbb{R}^{d_J}$ (linear combination of the outputs of last layer) and $(s+1)$ parameters for w^J. For layers apart from last layer, $2s+1$ parameters for $b^{(j)}$ and $(s+1)$ parameters for w^J. So the total number of free parameters in the deep CNN is $(3s+2)J + 2d_J - 2s - 1$, much smaller than that in a classical fully connected multi-layer neural network with full connection matrices $T^{(j)}$ involving $d_j d_{j-1}$ free parameters. It demonstrates the computational efficiency of deep CNNs.

2.2 Hypothesis Space

The hypothesis space of a learning algorithm is the set of all possible functions that can be represented or produced by the algorithm. For the deep CNN of depth J considered here, the hypothesis space is a set of functions defined by

$$\mathcal{H}_J^{w,b} = \left\{ \Sigma_{k=1}^{d_j} c_k h_k^J(x) : c \in \mathbb{R}^{d_J} \right\}$$

The approximation ability of the hypothesis space depend completely on the sequence of convolutional filter masks $\boldsymbol{w} = \{w^{(j)}\}_{j=1}^J$ and the sequence of bias vectors $\boldsymbol{b} = \{b^{(j)}\}_{j=1}^J$ which have been described above.

2.3 Theorem A

Let $2 \leq s \leq d$. For any compact subset Ω of $\mathbb{R}^d$ and any $f \in C(\Omega)$, $\exists$ a sequence $\boldsymbol{w}$ of filter masks, $\boldsymbol{b}$ of bias vectors and $f_J^{w,b} \in \mathcal{H}_J^{w,b}$ such that

$$\lim_{J \to \infty} \|(f - f_J^{w,b})\|_{C(\Omega)} = 0$$

where $C(\Omega)$ is the space of continuous functions on Ω with norm $\|f\|_{C(\Omega)} = sup_{x \in \Omega}|f(x)|$.

2.3.1 Essence of Theorem A

The result in theorem A verifies the result of universality of deep CNNs, asserting that any $f \in C(\Omega)$, can be approximated by $\mathcal{H}_J^{w,b}$ to an arbitrary accuracy when the depth J is large enough.

2.4 Theorem B

Let $2 \leq s \leq d$ and $\Omega \subseteq [-1,1]^d$. If $J \geqslant 2d/(s-1)$ and $f = F|_\Omega$ with $F \in H^r(\mathbb{R}^d)$ and an integer index $r > 2 + d/2$, then $\exists \boldsymbol{w}, \boldsymbol{b}$ and $f_J^{w,b} \in \mathcal{H}_J^{w,b}$ such that

$$\|(f - f_J^{w,b})\|_{C(\Omega)} \leqslant c\|F\|\sqrt{logJ}(1/J)^{1/2+1/d},$$

where c is an absolute constant and $\|F\|$ denotes the Sobolev norm of $F \in \mathcal{H}^r(\mathbb{R}^d)$.

2.4.1 Essence of Theorem B

Theorem B mainly presents rates of approximation by deep CNNs for functions in the Sobolev space $H^r(\Omega)$ with an integer index $r > 2 + d/2$. In this case the set $H^r(\Omega)$ is dense in $C(\Omega)$, so Theorem A follows from Theorem B by scaling.

2.4.2 Approximation ability of deep CNNs numerically

In theorem B if we take $s = \lceil 1 + d^{\tau}/2 \rceil$ and $J = \lceil 4d^{1-\tau} \rceil L$ with $0 \leqslant \tau \leqslant 1$ and $L \in \mathbb{N}$, where $\lceil u \rceil$ denotes the smallest integer not smaller than u, then we have

$$\|(f - f_J^{w,b})\|_{C(\Omega)} \leqslant c\|F\| \sqrt{\frac{(1-\tau)\log d + \log L + \log 5}{4d^(1-\tau)L}},$$

since $\lceil 4 \rceil = 5$, $\frac{1}{\lceil 4d^{1-\tau} \rceil^{1/d}} \leqslant 1$. Also, width of CNN is bounded by $12Ld$ and total number of free parameters $(5s+2)J + 2d - 2s - 1 \leqslant (73L+2)d$. If we take $L = 1$ and $\tau = 1/2$ to get a bound for the relative error

$$\frac{\|(f - f_J^{w,b})\|_{C(\Omega)}}{\|F\|} \leqslant \frac{c}{2} d^{-\frac{1}{4}} \sqrt{\log(5\sqrt{d})}$$

where depth of CNN $L = 1$ ad $\tau = 1/2$ is $J = \lceil 4d^{1-\tau} \rceil L = \left\lceil 4\sqrt{d} \right\rceil$ and at most $75d$ parameters. This explains the strong approximation of CNN and it's error bounded by depth of CNN.

References

[1] László Györfi. *A Distribution-Free Theory of Non-parametric Regression.* Springer, 2003.

[2] Ding-Xuan Zhou (2018) *Universality of Deep Convolutional Neural Networks.* Applied and Computational Harmonic Analysis: 787-794

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[4] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels.* The MIT Press, 2002.

[5] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, 2014.