Ve572 Lecture 11

Manuel and Jing

UM-SJTU Joint Institute

June 21, 2018

- The term neural network is a two-stage regression or classification.
- The key idea is to extract linear combinations of p inputs/features/predictors

$$x_{i1}, \quad x_{i2}, \quad \dots \quad x_{ij}, \quad \dots \quad x_{ip} \qquad i = 1, 2, \dots, n$$

as m derived features

$$z_{i1}, \quad z_{i2}, \quad \dots \quad z_{ij}, \quad \dots \quad z_{im} \qquad i = 1, 2, \dots, n$$

and them model the responses/target measures

$$y_{i1}, \quad y_{i2}, \quad \dots \quad y_{ij}, \quad \dots \quad y_{ik} \qquad i = 1, 2, \dots, n$$

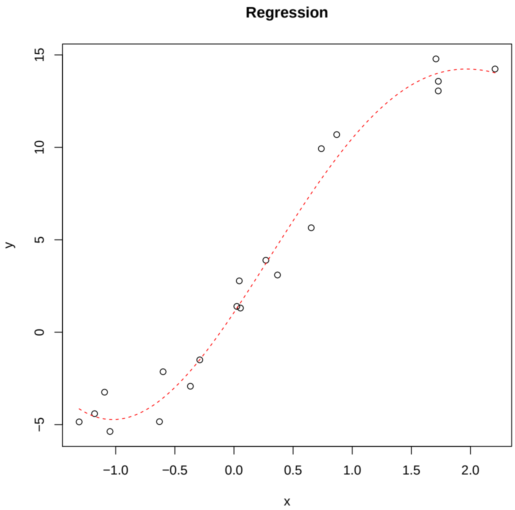
as a nonlinear function of those derived features.

- We will discuss the mostly wildly used neural net,

single hidden layer network

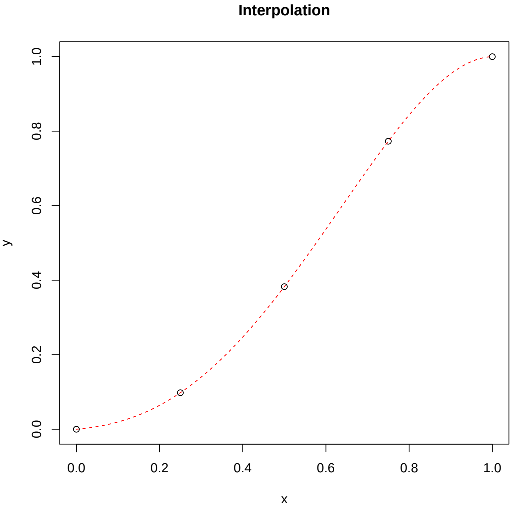
- Linear Regression, e.g.

$$f(y_i) = \hat{\beta}_0 + \hat{\beta}_1 g(x_i) + \hat{e}_i$$

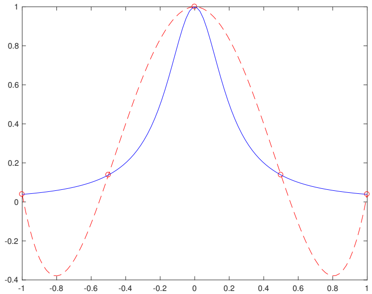


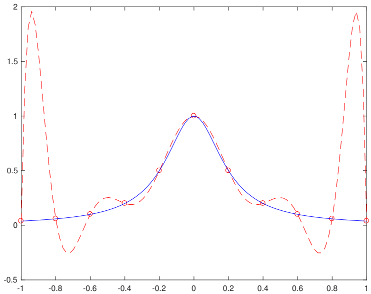
- Polynomial interpolation, e.g.

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta 4 x_i^4$$



- Runge's phenomenon and numerical singularity meant it is very limited.





- On the other hand, Spline, which breaks the region and use only low degree polynomial within each subinterval, is a much better approach in practice.

$$S(x)$$

- Natural Cubic spline

$$S(x) = \begin{cases} S_1(x) = a_1 + b_1 x + c_1 x^2 + d_1 x^3 & x_1 \le x \le x_2 \\ \quad \vdots & \vdots \\ S_i(x) = a_i + b_i x + c_i x^2 + d_i x^3 & x_i \le x \le x_{i+1} \\ \quad \vdots & \vdots \\ S_{n-1}(x) = a_{n-1} + b_{n-1} x + c_{n-1} x^2 + d_{n-1} x^3 & x_{n-1} \le x \le x_n \end{cases}$$

- Continuous

$$1. \qquad S_i(x_i) = y_i \qquad \text{and} \qquad S_i(x_{i+1}) = y_{i+1}$$

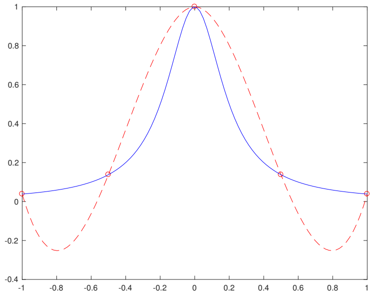
- Continuously differentiable

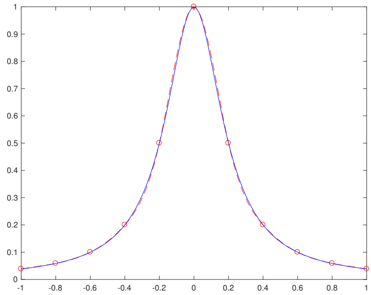
$$2. \qquad S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}), \qquad 3. \qquad S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$$

- Extra parameters

$$S_1''(x_1) = S_{n-1}''(x_n) = 0$$

- No Runge's phenomenon and numerically stable

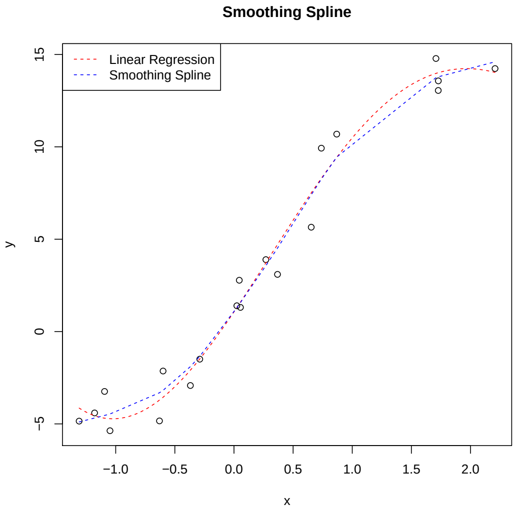




- However, if there are error terms, then it is clearly overfitting the data.
- For those cases, we consider the option of combining spline with regression.

- Smoothing spline

$$y_i = f(x_i) + e_i; \qquad \sum_{i=1}^{n} \left(y_i - \hat{f}(x_i) \right)^2 + \lambda \int \left(\hat{f}''(x) \right)^2 dx$$



- Projection Pursuit Regression

$$y_i = f(\mathbf{x}_i) + e_i; \qquad f(\mathbf{x}_i) = \sum_{j=1}^{m} g_j\left(\mathbf{u}_j^{\mathrm{T}}\mathbf{x}_i\right)$$

where $\mathbf{x}_i \in \mathbb{R}^p$ denotes the feature vector for the ith observation, and

$$\mathbf{u}_j \in \mathbb{R}^p$$

is a unit vector of unknown parameters, which gives the direction.

- Notice the projection is essentially the derived feature

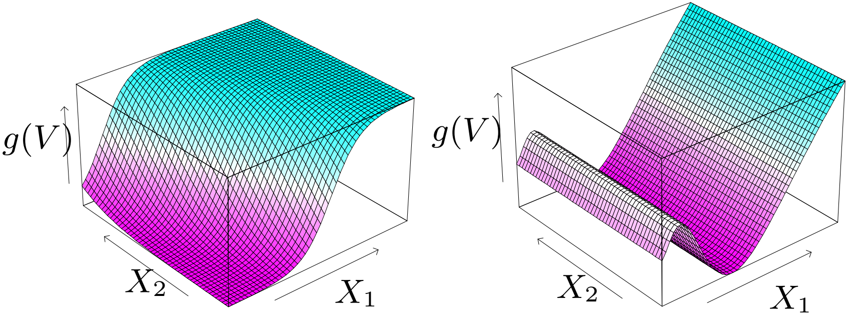
$$z_{ij} = \mathbf{u}_j^{\mathrm{T}}\mathbf{x}_i$$

- The functions g_j are often unspecified and estimated along the with $\mathbf{u}$, they are scalar-valued functions of just one variable, but can be understood as

ridge functions

which changes only if the coordinate of a certain direction changes, i.e. $\mathbf{u}$.

- Two ridge functions:
- Left: $g(V) = \dfrac{1}{1+e^{-5(V-0.5)}}$, where $V = \dfrac{X_1 + X_2}{\sqrt{2}}$.



- Right: $g(V) = (V+0.1)\sin\left(\dfrac{1}{V/3+0.1}\right)$, where $V = X_1$.

- Given a training dataset $(\mathbf{x}_i, y_i)$, for $i = 1, 2, \ldots, n$, we seek the minimum

$$\sum_{i=1}^{n} \left[y_i - \sum_{j=1}^{m} g_j \left(\mathbf{u}_j^{\mathrm{T}} \mathbf{x}_i \right) \right]^2$$

over function g_j and direction vectors $\mathbf{u}_j$.

- A typical algorithm will start with a random initialisation of $\mathbf{u}_1$

$$\sum_{i=1}^{n} \left[y_i - g_1 \left(\mathbf{u}_1^{\mathrm{T}} \mathbf{x}_i \right) \right]^2$$

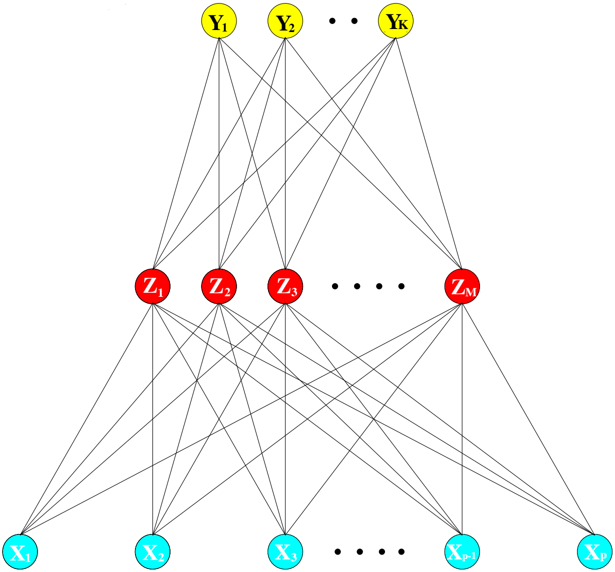
and one ridge function, which is solve by smoothing spline.

- Given g_1, then $\mathbf{u}_1^{\mathrm{T}}$ is updated by minimising

$$\sum_{i=1}^{n} \left[y_i - g_1 \left(\mathbf{u}_1^{\mathrm{T}} \mathbf{x}_i \right) \right]^2$$

with respect to $\mathbf{u}$ using numerical optimisation before $g_2 \left(\mathbf{u}_2^{\mathrm{T}} \mathbf{x}_i \right)$ is added.

- Neural Network



- Notice again neural network can handle more than one responses, that is,

$$y_{i1}, \quad y_{i2}, \qquad \dots \qquad y_{ij}, \qquad \dots \qquad y_{ik} \qquad i = 1, 2, \dots, n$$

- For each response, we have

$$y_{ij} = f_j\left(\mathbf{x}_i\right) + e_i$$

and we minimise the following

$$\sum_{i=1}^{n} \sum_{j=1}^{k} \left(y_{ij} - f_j\left(\mathbf{x}_i\right)\right)^2$$

- Thus the general idea should be really familiar to you.
- Neutral network extends linear models by having the extra layer and a non-linear transformation instead of a linear one in terms of parameters

$$y_i = f(x_i) + e_i = \hat{\beta}_0 + \hat{\beta}_1 g(x_i) + \hat{e}_i$$

- For a typical one hidden layer neural network, we have

$$
\begin{aligned}
f_j(\mathbf{x}) &= g_j(\mathbf{T}) && \text{where } \mathbf{T} = T_1\mathbf{e}_1 + T_2\mathbf{e}_2 + \cdots + T_k\mathbf{e}_k \\
T_j &= \beta_{0j} + \boldsymbol{\beta}_j^{\mathrm{T}}\mathbf{Z} && \text{where } \mathbf{Z} = Z_1\mathbf{e}_1 + Z_2\mathbf{e}_2 + \cdots + Z_m\mathbf{e}_m \\
Z_\ell &= \sigma\left(\alpha_{0j} + \boldsymbol{\alpha}_\ell^{\mathrm{T}}\mathbf{x}\right)
\end{aligned}
$$

- The transformation applies to $V = \alpha_{0j} + \boldsymbol{\alpha}_\ell^{\mathrm{T}}\mathbf{x}$ is usually chosen to be

$$\sigma(V) = \frac{1}{1 + e^{-V}}$$

- The transformation applies to $\mathbf{T} = \beta_{0j} + \boldsymbol{\beta}_j^{\mathrm{T}}\mathbf{Z}$ is usually chosen to be

$$
\begin{aligned}
g_j(\mathbf{T}) &= T_j && \text{for regression} \\
g_j(\mathbf{T}) &= \frac{e^{T_j}}{\mathbf{T}^{\mathrm{T}}\mathbf{1}} && \text{for classification}
\end{aligned}
$$

- Notice the neural network model with one hidden layer has the same form

$$\sum_{i=1}^{n}\sum_{j=1}^{k}\left(y_{ij}-f_j\left(\mathbf{x}_i\right)\right)^2=\sum_{i=1}^{n}\left(y_i-f\left(\mathbf{x}_i\right)\right)^2$$
$$=\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{\ell=1}^{m}\beta_\ell\sigma\left(\alpha_0+\alpha_\ell^{\mathrm{T}}\mathbf{x}_i\right)\right)^2$$

  as the projection pursuit regression when there is only one response

$$\sum_{i=1}^{n}\left[y_i-\sum_{j=1}^{m}g_j\left(\mathbf{u}_j^{\mathrm{T}}\mathbf{x}_i\right)\right]^2$$

- Therefore, NN can be thought as a generalisation of PPR in this case

$$g_\ell\left(\mathbf{u}_\ell^{\mathrm{T}}\mathbf{x}_i\right)=\frac{\beta_0}{m}+\beta_\ell\sigma\left(\alpha_0+\boldsymbol{\alpha}_\ell^{\mathrm{T}}\mathbf{x}_i\right)=\frac{\beta_0}{m}+\beta_\ell\sigma\left(\alpha_0+\|\boldsymbol{\alpha}_\ell\|\mathbf{u}^{\mathrm{T}}\mathbf{x}_i\right)$$

  where NN uses simple $\sigma(V)$ instead of nonparametric functions $g_\ell(V)$.