Regression

A training set of P input/output observation pairs:

$$\{(\bar{x}_1, y_1), (\bar{x}_2, y_2), \ldots, (\bar{x}_P, y_P)\}$$

where

$$\bar{x}_p = \begin{bmatrix} x_{1,p} \\ x_{2,p} \\ \vdots \\ x_{N,p} \end{bmatrix}$$

Then we have a bias and N associated weights,

$$b + \bar{x}_p \bar{w} \approx y_p \ , \quad p = 1, \ldots, P.$$

The Least Squares cost function for linear regression:

For a given set of parameters $(b, \bar{w})$ this cost function computes the total squared error between the associated hyperplane and the data.

$$g(b, \bar{w}) = \sum_{p=1}^{P} (b + \bar{x}_p^T \bar{w} - y_p)^2$$

$$\therefore \underset{b, \bar{w}}{\text{minimize}} \sum_{p=1}^{P} (b + \bar{x}_p^T \bar{w} - y_p)^2$$

Minimization of the Least Squares cost function.

Assume $\quad \tilde{x}_p = \begin{bmatrix} 1 \\ \bar{x}_p \end{bmatrix} \quad \tilde{w} = \begin{bmatrix} b \\ \bar{w} \end{bmatrix}$

$$\therefore g(\tilde{w}) = \sum_{p=1}^{P} (\tilde{x}_p^T \tilde{w} - y_p)^2$$

$$\nabla g(\tilde{w}) = 2 \sum_{p=1}^{P} \tilde{x}_p (\tilde{x}_p^T \tilde{w} - y_p) = 2 \left(\sum_{p=1}^{P} \tilde{x}_p \tilde{x}_p^T \right) \tilde{w} - 2 \sum_{p=1}^{P} \tilde{x}_p y_p$$

$$\left(\sum_{p=1}^{P} \tilde{x}_p \tilde{x}_p^T\right) \tilde{w} = \sum_{p=1}^{P} \tilde{x}_p y_p$$

$$\Rightarrow \quad \tilde{w}^* = \left(\sum_{p=1}^{P} \tilde{x}_p \tilde{x}_p^T\right)^{-1} \sum_{p=1}^{P} \tilde{x}_p y_p$$

Mean squared error (MSE): $MSE = \frac{1}{P} \sum_{p=1}^{P} (b^* + \bar{x}_p^T \bar{w}^* - y_p)^2 \rightarrow$ efficacy of a learned model.

More generally, $b + f(x_p) w \approx y_p, \quad p = 1, \ldots, P$

$$\underset{b,w}{\text{minimize}} \sum_{p=1}^{P} (b + f_p w - y_p)^2$$

Assume $\tilde{f}_p = \begin{bmatrix} 1 \\ f_p \end{bmatrix}, \quad \tilde{w} = \begin{bmatrix} b \\ w \end{bmatrix}$

$$g(\tilde{w}) = \sum_{p=1}^{P} (\tilde{f}_p^T \tilde{w} - y_p)^2.$$

$$\therefore \quad \tilde{w}^* = \left(\sum_{p=1}^{P} \tilde{f}_p \tilde{f}_p^T\right)^{-1} \sum_{p=1}^{P} \tilde{f}_p y_p$$

"tanh" function (step function)

$$\cancel{\tanh(b + \bar{x}^T \bar{w}) = 2\sigma(b + \bar{x}^T \bar{w}) - 1}$$

$$\tanh(b + \bar{x}^T \bar{w}) \approx \begin{cases} +1 & \text{if } b + \bar{x}^T \bar{w} > 0 \\ -1 & \text{if } b + \bar{x}^T \bar{w} \leq 0 \end{cases}$$

$$\therefore \quad \tanh(y_p (b + \bar{x}^T \bar{w})) \approx 1$$

$$\Rightarrow \quad 1 + e^{-y_p (b + \bar{x}_p^T \bar{w})} \approx 1$$

$$\Rightarrow \quad \log(1 + e^{-y_p (b + \bar{x}_p^T \bar{w})}) \approx 0$$

$$\therefore \quad \underset{b, \bar{w}}{\text{minimize}} \sum_{p=1}^{P} \log(1 + e^{-y_p (b + \bar{x}_p^T \bar{w})})$$

Chapter 4 Automatic feature design for regression.

4.1 Automatic feature design for the ideal regression scenario

4.1.1 continuous function approximation

A basis is a set of basic feature transformation $\{f_m(\bar{x})\}_{m=1}^{\infty}$ such that all points $\bar{x}$ in the unit hypercube we can express $y(\bar{x})$ perfectly as

$$\sum_{m=0}^{\infty} f_m(\bar{x})\, w_m = y(\bar{x})$$

for large enough M we can approximate y over its input domain as:

$$\sum_{m=0}^{M} f_m(\bar{x})\, w_m \approx y(\bar{x})$$

This approximation can be made as finely as desired by increasing the number of basis features M, i.e.. by increasing M we can design automatically perfect features to represent $y(\bar{x})$

4.1.2 Common base for continuous function approximation.

Polynomial basis: $(N=1)$

$$\begin{cases} f_0(x) = 1 & m=0 \\ f_m(x) = x^m & \text{for all } m \geq 1. \end{cases}$$

Fourier basis / sinusoidal basis. $(N=1)$

$$\begin{cases} f_0(x) = 1 & m=0 \\ f_{2m-1}(x) = \cos(2\pi m x) & \text{for all } m \geq 1 \\ f_{2m}(x) = \sin(2\pi m x) & \text{for all } m \geq 1 \end{cases}$$

multilayer Neural network (N dimentional) basis

Activation function.

① The hyperbolic tangent function:

$$f_m(\bar{x}) = \tanh(c_m + \bar{x}^T \bar{V}_m) \quad \text{for all } m \geq 1$$

② The max or hinge function

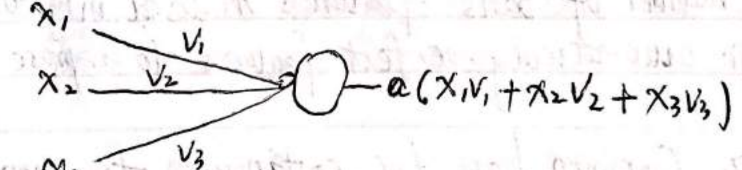
$$f_m(\bar{x}) = \max(0, c_m + \bar{x}^T \bar{V}_m) \quad \text{for all } m \geq 1$$

Graphical representation of a neural network

It is common to represent the weighted sum of M neural network basis features.

$$r = b + \sum_{m=1}^{M} f_m(\bar{x}) w_m$$

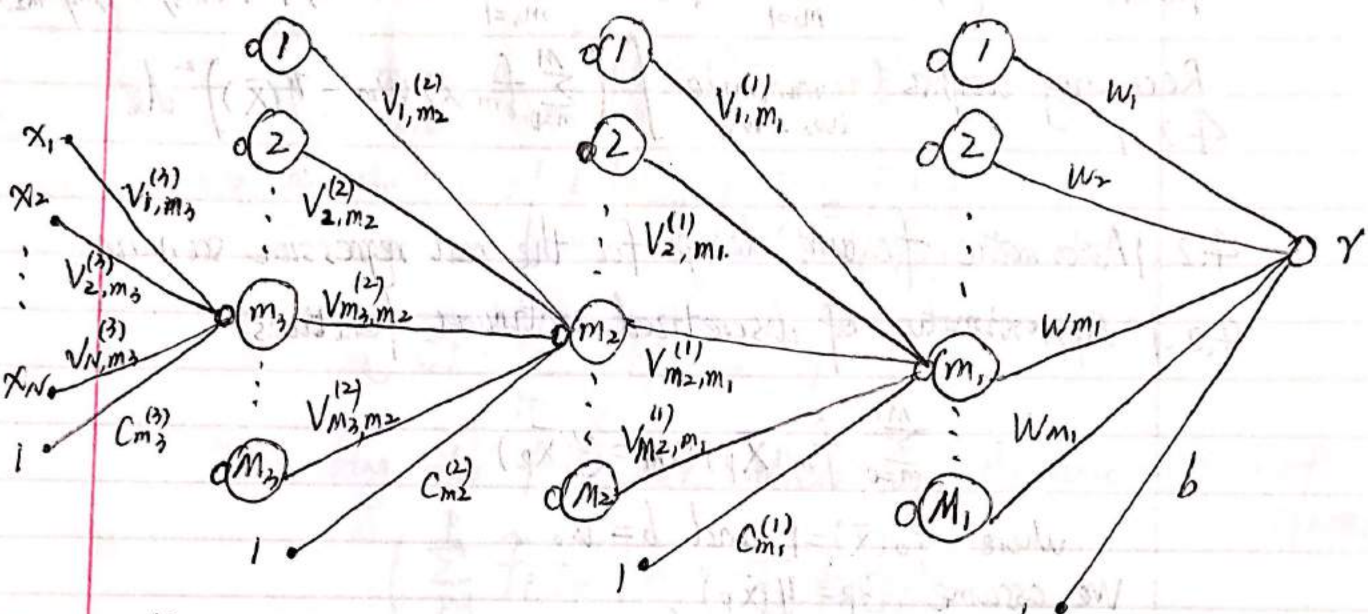
A simple example.



where $a(\cdot)$ is any activation function.

- weighted edges
- summation unit
- activation unit

Three hidden layer neural network.



① $M = M_1$

② input $\bar{x}$ is shown entry-wise on the left and output r on the right.

③ In between the three hidden layers, from right to left each consist of M_1, M_2 and M_3 hidden units respectively.

④ A basis with $L>2$ or 3 hidden-layers is usually called a deep network

A single basis element of a feed forward neural network with two hidden layers. Doing this with the tanh basis:

$$f_m(\bar{x}) = \tanh\left(C_m^{(1)} + \sum_{m_2=1}^{M_2} \tanh\left(C_{m_2}^{(2)} + \bar{x}^T \bar{v}_{m_2}^{(2)}\right) v_{m_2,m}^{(1)}\right)$$

or hinge function:

$$f_m(\bar{x}) = \max\left(0,\ C_m^{(1)} + \sum_{m_2=1}^{M_2} \max\left(0, C_{m_2}^{(2)} + \bar{x}^T \bar{v}_{m_2}^{(2)}\right) v_{m_2,m}^{(1)}\right)$$

A neural network with three hidden layers

hinge basis function:

$$f_m(\bar{x}) = \max\left(0, c_m^{(1)} + \sum_{m_2=1}^{M_2} \max\left(0, c_{m_2}^{(2)} + \sum_{m_3=1}^{M_3} \max\left(0, c_{m_3}^{(3)} + \bar{x}^T \bar{v}_{m_3}^{(3)}\right) v_{m_3,m_2}^{(2)}\right) v_{m_2,m}^{(1)}\right)$$

Recovering weights: $\underset{w_0,\dots,w_M}{\text{minimize}} \int \left(\sum_{m=0}^{M} f_m(\bar{x}) w_m - y(\bar{x})\right)^2 d\bar{x}$

4.2. Automatic feature design for the real regression scenario.

4.2.1 Approximation of discretized continuous functions

$\because \sum_{m=0}^{M} f_m(\bar{x}_p) w_m = y(\bar{x}_p)$

where $f_0(\bar{x}) = 1$ and $b = w_0$.

We assume $y_p = y(\bar{x}_p)$,

feature vector $\bar{f}_p = [f_1(\bar{x}_p) \ f_2(\bar{x}_p) \dots f_M(\bar{x}_p)]^T$

weight vector $\bar{w} = [w_1 \ w_2 \ \dots \ w_M]^T$

$\therefore b + \bar{f}_p^T \bar{w} \approx y_p$

$\therefore \underset{b, \bar{w}}{\text{minimize}} \sum_{p=1}^{P} (b + \bar{f}_p^T \bar{w} - y_p)^2$

4.2.2. The real regression scenario

The general instance of regression is a function approximation problem based on noisy samples of the underlying function.

Exp. Regression with fixed basis of features.

Employing a degree D polynomial or Fourier basis for a $\boxed{\text{scalar input}}$.

$\bar{f}_p = [x_p \ x_p^2 \dots x_p^D]^T$ or $\bar{f}_p = [\cos(2\pi x_p) \quad \sin(2\pi x_p)$

$\cdots \cos(2\pi_{L}DX_p) \quad \sin(2\pi DX_p)]]^T$

$$\therefore \underset{b, \bar{w}}{\text{minimize}} \sum_{p=1}^{P} (b + \bar{f}_p^T \bar{w} - y_p)^2$$

We assume $\tilde{w} = \begin{bmatrix} b \\ \bar{w} \end{bmatrix}$ and $\tilde{f}_p = \begin{bmatrix} 1 \\ \bar{f}_p \end{bmatrix}$

$$\therefore g(\tilde{w}) = \sum_{p=1}^{P} (\tilde{f}_p^T \tilde{w} - y_p)^2$$

checking the first order condition then gives the linear system of equations

$$\left(\sum_{p=1}^{P} \tilde{f}_p \tilde{f}_p^T \right) \tilde{w} = \sum_{p=1}^{P} \tilde{f}_p y_p$$

that when solved recovers an optimal set of parameters $\tilde{w}$.

Exp: Regression with a basis of single hidden layer neural network features.

feature vector: $\bar{f}_p = [a(c_1 + \bar{x}_p^T \bar{v}_1) \quad a(c_2 + \bar{x}_p^T \bar{v}_2) \ldots a(c_M + \bar{x}_p^T \bar{v}_M)]^T$

$$\underset{b, \bar{w}, \Theta}{\text{minimize}} \sum_{p=1}^{P} (b + \bar{f}_p^T \bar{w} - y_p)^2$$

$$\nabla g = \left[\frac{\partial}{\partial b} g \ \ \frac{\partial}{\partial w_1} g \cdots \frac{\partial}{\partial w_M} g \ \ \frac{\partial}{\partial c_1} g \cdots \frac{\partial}{\partial c_M} g \ \ \nabla_{\bar{v}_1}^T g \cdots \nabla_{\bar{v}_M}^T g \right]^T$$

We have a vector of length $Q = M \cdot (N+1) + M + 1$
$= M \cdot (N+2) + 1$

4.3 Cross-validation for regression

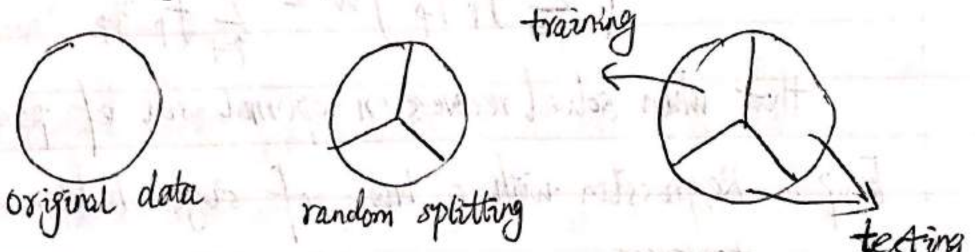
An effective framework for choosing the proper value for M automatically and intelligently so as to prevent the problem of underfitting / overfitting

4.3.1 Diagnosing the problem of overfitting/underfitting.

Criterion: the number M of basis features used should be such that the corresponding model fits well to both the current dataset as well as to new data we will receive in the future.

4.3.2 Hold out cross-validation.

In general the larger/smaller the original dataset, the larger/smaller the portion of original data that should be assigned to the testing set (between 1/10 to 1/3 typically)



k non-overlapping sets (here $k=3$)

4.3.3 Hold out calculation.

$\Omega_{train} = \{p \mid (\bar{x}_p, y_p)$ belongs to the training set$\}$

$\Omega_{test} = \{p \mid (\bar{x}_p, y_p)$ belongs to the testing set$\}$

We then choose a basis type (e.g., polynomial, Fourier, neural network) and choose a range for the number of basis features over which we search for an ideal value for M.

Procedure: 1) Form the corresponding feature vector $F_p = [f_1(\bar{x}_p)$

$f_2(\bar{x}_p) \dots f_m(\bar{x}_p)]^T$

2) fit a corresponding model to the training set by LS.

$$\underset{b,\, \bar{w},\, \Theta}{\text{minimize}} \sum_{p \in \Omega_{train}} \left(b + \bar{f}_p^T \bar{w} - y_p\right)^2$$

Denoting a solution to the problem above as $(b_m^*, \bar{w}_m^*, \Theta_m^*)$

$$\text{Training error} = \frac{1}{|\Omega_{train}|} \sum_{p \in \Omega_{train}} \left(b_m^* + \bar{f}_p^T \bar{w}_m^* - y_p\right)^2$$

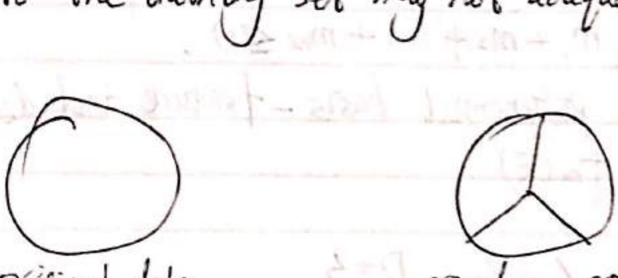
$$\text{Testing error} = \frac{1}{|\Omega_{test}|} \sum_{p \in \Omega_{test}} \left(b_m^* + \bar{f}_p^T \bar{w}_m^* - y_p\right)^2$$

We choose the one that provides the lowest testing error denoted by m^*.

3) Form the feature vector $\bar{f}_p = [f_1(\bar{x}_p)\ f_2(\bar{x}_p) \cdots f_{m^*}(\bar{x}_p)]^T$ for all the points in the entire dataset and solve the LS problem over the entire dataset to form the final model

4.3.4 K-fold cross-validation

Flaw of Hold out method. Having been chosen at random, the points assigned to the training set may not adequately describe the original data.



original data random splitting

1st fold 2nd fold 3rd fold.

By averaging the testing errors and choosing the model with minimum associated average test error, and determine an excellent model for the phenomenon.

Supplement: Comparison:

Scalar x Vector $\bar{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$

Approximation $y(x) \approx \sum_{m=0}^{M} f_m(x) w_m$ $y(\bar{x}) \approx \sum_{m=0}^{M} f_m(\bar{x}) w_m$

Polynomial $f_m(x) = x^m$ $f_m(\bar{x}) = x_1^{m_1} x_2^{m_2} \cdots x_N^{m_N}$

Single hidden-layer NN $f_m(x) = a(c_m + x v_m)$ $f_m(\bar{x}) = a(c_m + \bar{x}^T \bar{v}_m)$

For a general N-dimensional input $\bar{x}$,

1) Polynomial:

$$f_m(\bar{x}) = x_1^{m_1} x_2^{m_2} \cdots x_N^{m_N}$$

where $0 \le m_1 + m_2 + \cdots + m_N \le D$.

That means polynomial basis-feature includes all monomials of form $f_m(\bar{x})$

Exp. $N=2$, $D=3$

$$\begin{array}{ccc} x_1 & x_1x_2 & x_1^2x_2 \\ x_2 & x_1^2 & x_1x_2^2 \\ \hline D_1 & x_2^2 & x_1^3 \\ & \overline{D_2} & x_2^3 \\ & & \overline{D_3} \end{array} \qquad M = 9+1 = 10.$$

$$M = \binom{N+D}{D} = \frac{(N+D)!}{D!\,(N+D-D)!} = \frac{(N+D)!}{D!\,N!}$$

2) Fourier:

$$f_m(\bar{x}) = e^{2\pi i m_1 x_1} e^{2\pi i m_2 x_2} \cdots e^{2\pi i m_N x_N}$$

where $-D \leq m_1, m_2 \cdots m_N \leq D$.

$$M = (2D+1)^N$$

$\because \cos(\alpha) = \frac{1}{2}(e^{j\alpha} + e^{-j\alpha})$ and $\sin(\alpha) = \frac{1}{2i}(e^{j\alpha} - e^{-j\alpha})$

we can write the partial Fourier expansion.

$$w_0 + \sum_{m=1}^{M} (\cos(2\pi m x) w_{2m-1} + \sin(2\pi m x) w_{2m})$$

equivalently as

$$\sum_{m=-M}^{M} e^{2\pi i m x} w_m'$$

$$\text{where } w_m' = \begin{cases} \frac{1}{2}(w_{2m-1} - i w_{2m}) & \text{if } m>0 \\ w_0 & \text{if } m=0 \\ \frac{1}{2}(w_{-2m-1} - i w_{2m}) & \text{if } m<0 \end{cases}$$