Machine Learning	**(Due: 20th May)**

Assignment #4 (Neural Networks)

Instructor: Beilun Wang — *Name:* Haorui li, *ID:* 61518407

Problem Description:

Problem 1: Model selection and learning theory

(1) Let $X_i,\ i = 1, 2, \cdots, n$ be n i.i.d observations from CDF $F(t) = P(X \leqslant t)$. If we estimate the true CDF by empirical CDF which is

$$\hat{F}_n(t) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{I}(X_i \leqslant t)$$

where $\mathcal{I}(p) = 1$ if statement p is true, otherwise 0.

Write down the expectation and variance of $\hat{F}_n(t)$. Then use Khinchin's law(i.e., the weak law of large numbers) to show that $\hat{F}_n(t) \xrightarrow{P} F(t)$.

(2) Supoose we have a target variable y and a vector of inputs $\boldsymbol{x}$, and the true model is $f(\boldsymbol{x})$. If we assume that $y = f(\boldsymbol{x}) + \varepsilon$ where ε is a Gaussian noise with $E(\varepsilon) = 0$ and $Var(\varepsilon) = \sigma_\varepsilon^2$, we can derive an expression for the expected prediction error of a regression fit $\hat{f}(\boldsymbol{x})$ at an point $\boldsymbol{x} = \boldsymbol{x}_0$, using squared-error loss:

$$EPE(\boldsymbol{x}_0) = \sigma_\varepsilon^2 + Bias^2(\hat{f}(\boldsymbol{x}_0)) + Var(\hat{f}(\boldsymbol{x}_0)).$$

Now give training set $\boldsymbol{X}$ and the corresponding labels $\boldsymbol{y}$.

(a) For a linear model fit $\hat{f}_p(\boldsymbol{x}) = \boldsymbol{x}^\top\hat{\boldsymbol{\beta}}$, where the parameter vector $\hat{\boldsymbol{\beta}}$ with p components is fit by least squares, write down the closed form solution of $\hat{\boldsymbol{\beta}}$. (*Assume that* $\boldsymbol{X}^\top\boldsymbol{X}$ *is invertible.*)

If $\hat{\boldsymbol{\beta}}$ is fit by least squares using ridge regression with regularization parameter α, we can get a model $\hat{f}_\alpha(\boldsymbol{x})$. Also write down the closed form solution of $\hat{\boldsymbol{\beta}}_\alpha$.

(b) For the linear model $\hat{f}_p(\boldsymbol{x}) = \boldsymbol{x}^\top\hat{\boldsymbol{\beta}}$, write down the expression of EPE at $\boldsymbol{x} = \boldsymbol{x}_0$ using the solution $\hat{\boldsymbol{\beta}}$ in (a). (*Assume only* y *is random variable*).

(c) For the model $\hat{f}_\alpha(\boldsymbol{x}) = \boldsymbol{x}^\top\hat{\boldsymbol{\beta}}_{\boldsymbol{\alpha}}$, write the down the expression of EPE at $\boldsymbol{x} = \boldsymbol{x}_0$ using the solution $\hat{\boldsymbol{\beta}}_\alpha$ in (a). (*Assume only* y *is random variable*).

(3) Suppose we stack the outcomes $y_1, y_2, ...y_N$ into a vector $\boldsymbol{y}$, and similarly for the predictions $\hat{\boldsymbol{y}}$. Then a linear fitting method is one for which we can write

$$\hat{\boldsymbol{y}} = \boldsymbol{S}\boldsymbol{y},$$

where S is an $N \times N$ matrix depending on the input vectors x_i but not on the y_i.
Let $\hat{\boldsymbol{f}} = \boldsymbol{S}\boldsymbol{y}$ be a linear fitting of $\boldsymbol{y}$, and let $\hat{\boldsymbol{f}}^{-i}$ be the fitted function computed with $(\boldsymbol{x}_i, y_i)$ removed. If S_{ii} is the ith diagonal element of $\boldsymbol{S}$, show that for

$$\boldsymbol{S} = \boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{\Omega})^{-1}\boldsymbol{X}^\top = \boldsymbol{X}\boldsymbol{A}^{-1}\boldsymbol{X}^\top (A\ is\ PSD)$$

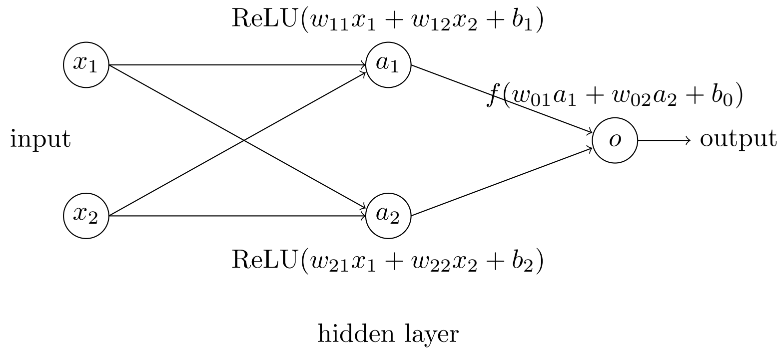
the cross-validated residual can be written as

$$y_i - \hat{f}^{-i}(\boldsymbol{x}_i) = \frac{y_i - \hat{f}(\boldsymbol{x}_i)}{1 - S_{ii}}$$

(**Hint**: use a lemma $(\boldsymbol{A} - \boldsymbol{x}\boldsymbol{x}^\top)^{-1} = \boldsymbol{A}^{-1} + \frac{\boldsymbol{A}^{-1}\boldsymbol{x}\boldsymbol{x}^\top\boldsymbol{A}^{-1}}{1-\boldsymbol{x}^\top\boldsymbol{A}^{-1}\boldsymbol{x}}$)

Problem 2: Neural Networks

Suppose that we apply neural networks on a problem which has boolean inputs $\boldsymbol{x} \in \{0,1\}^p$ and boolean output $y \in \{0,1\}$. The network structure example is showed as below. In this example we set $p = 2$, single hidden layer with 2 neurons, activation function $\mathrm{ReLU}(u) = u$ if $u > 0$ otherwise 0, and an additional threshold function(e.g., $f(v) = 1$ if $v > 0$, otherwise $f(v) = 0$) for output layer.



(1) Using the structure and settings of neural network above, show that such a simple neural network could output the function x_1 XOR x_2(equals to 0 if $x_1 = x_2$ and otherwise 1), which is impossible for linear models. State the values of parameters(i.e., w_{ij} and b_i) you found.

(2) Now we allow the number of neurons in the hidden layer to be more than 2 but finite. Retain the structure and other settings. Show that such a neural network with single hidden layer could output an arbitrary binary function $h: \{0,1\}^p \mapsto \{0,1\}$. You can apply threshold function after each neuron in the hidden layer.

Answer:

Problem 1: Model selection and learning theory

(1)

$$E(\mathcal{I}\{x_i \leq t\}) = P(x_i \leq t)$$

$$E(\hat{F}_n(t)) = \frac{1}{n}\sum_{i=1}^{n} E(\mathcal{I}\{x_i \leq t\}) = \frac{1}{n}\sum_{i=1}^{n} P(x_i \leq x) = \frac{1}{n}\sum_{i=1}^{n} F_i(t)$$

$$E(F_n^2) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{i=1}^{n} E(\mathcal{I}\{x_i \leq t\}\mathcal{I}\{x_i \leq t\}) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{i=1}^{n} P(x_i \leq x, x_j \leq x)$$

$$Var(F_n(x)) = E(F_n)^2 - E(F_n^2) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{i=1}^{n} P(x_i \leq x, x_j \leq x) - (\frac{1}{n}\sum_{i=1}^{n} F_i(t))^2$$

Let$\mu = E(\mathcal{L}(X_i \leq t)) = F(t)$

$$lim_{n\to} P\{|\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(X_i \leq t) - \mu| < \varepsilon\} = 1$$

Therefore:

$$lim_{n\to} P\{|\hat{F}_n - F(t)| < \varepsilon\} = 1$$

(2)

(a)

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T Y$$

$$\hat{\boldsymbol{\beta}_\alpha} = (X^T X + \alpha I)^{-1} X^T Y$$

(b)

$$\begin{aligned}
&EPE\left(\boldsymbol{x}_0\right) = \sigma_\varepsilon^2 + \operatorname{Bias}^2\left(f\left(\boldsymbol{x}_0\right)\right) + \operatorname{Var}\left(f\left(\boldsymbol{x}_0\right)\right) \\
&= \sigma_\varepsilon^2 + \left(f\left(\boldsymbol{x}_0\right) - E\hat{f}\left(\boldsymbol{x}_0\right)\right)^2 + \operatorname{Var}\left(\hat{f}\left(\boldsymbol{x}_0\right)\right) \\
&= \sigma_\varepsilon^2 + \left(f\left(\boldsymbol{x}_0\right) - E\hat{f}\left(\boldsymbol{x}_0\right)\right)^2 + \left(\boldsymbol{x}_0^T\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\right)^2 \operatorname{Var}(\boldsymbol{y}) \\
&= \sigma_\varepsilon^2 + \left(f\left(\boldsymbol{x}_0\right) - E\hat{f}\left(\boldsymbol{x}_0\right)\right)^2 + \left(\boldsymbol{x}_0^T\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\right)^2 \sigma_\varepsilon^2
\end{aligned}$$

(c)

$$\begin{aligned}
&EPE\left(\boldsymbol{x}_0\right) = \sigma_\varepsilon^2 + \operatorname{Bias}^2\left(\hat{f}\left(\boldsymbol{x}_0\right)\right) + \operatorname{Var}\left(\hat{f}\left(\boldsymbol{x}_0\right)\right) \\
&= \sigma_\varepsilon^2 + \left(f\left(\boldsymbol{x}_0\right) - E\hat{f}\left(\boldsymbol{x}_0\right)\right)^2 + \operatorname{Var}\left(\hat{f}\left(\boldsymbol{x}_0\right)\right) \\
&= \sigma_\varepsilon^2 + \left(f\left(\boldsymbol{x}_0\right) - E\hat{f}\left(\boldsymbol{x}_0\right)\right)^2 + \left(\boldsymbol{x}_0^T\left(\boldsymbol{X}^\top\boldsymbol{X} + \alpha\boldsymbol{I}\right)^{-1}\boldsymbol{X}^\top\right)^2 \sigma_\varepsilon^2
\end{aligned}$$

(3)

$$S_{ii} = x_i^T (X^T X + \lambda\Omega)^{-1} x_i$$

$$\hat{\boldsymbol{f}}(x_i) = x_i^T (X^T X + \lambda\Omega)^{-1} X^T y$$

$$\hat{\boldsymbol{f}}^{-i}(x_i) = x_i^T(X^TX - x_ix_i^T + \lambda\Omega)^{-1}(X^Ty - x_iy_i) \tag{1}$$
$$= x_i^T(A - x_ix_i^T)^{-1}(X^Ty - x_iy_i) \tag{2}$$
$$= y_i - \frac{y_i - \hat{f}(\boldsymbol{x}_i)}{1 - S_{ii}} \tag{3}$$

Problem 2: Neural Networks

(1)

$$w_{11} = 1; w_{12} = -1; w_{21} = -1; w_{22} = 1; w_{01} = w_{02} = 1; b_1 = b_2 = b_0 = 0$$
$$f = \begin{cases} 1, x > 0 \\ 0,\ \text{other} \end{cases}$$

(2) Any boolean functions can be represented by AND, NOT and OR.
And these logic can be represented by neural networks as follow:

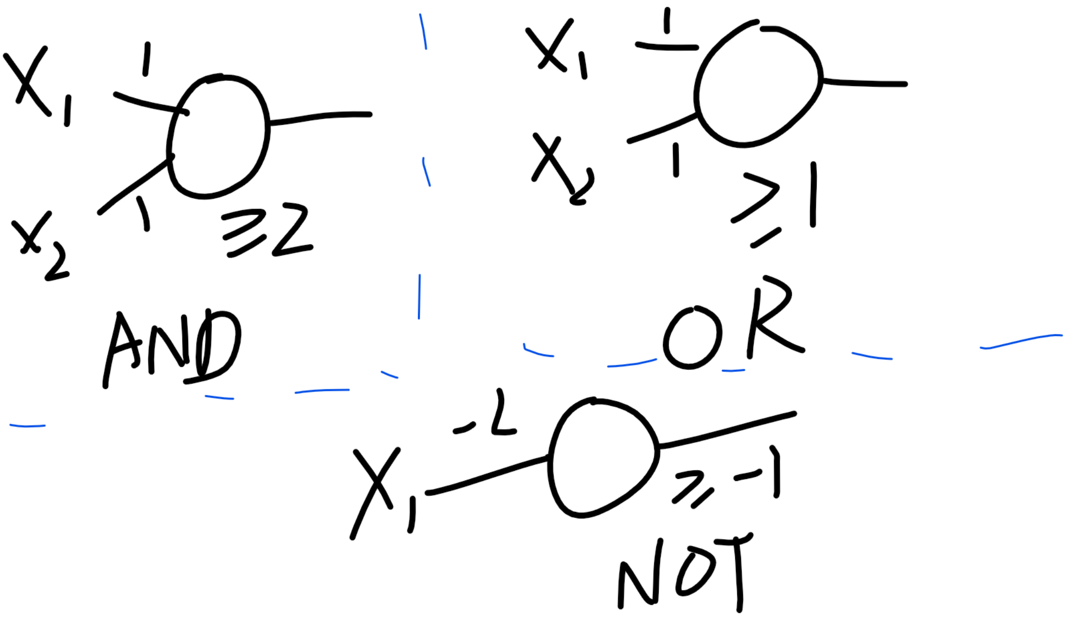


Figure 1: Network logic

So, NN can represent arbitrary binary functions.