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

Assignment #4 (Neural Networks)

Instructor: Beilun Wang *Name:* Qipeng Zhu, *ID:* 58119304

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

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

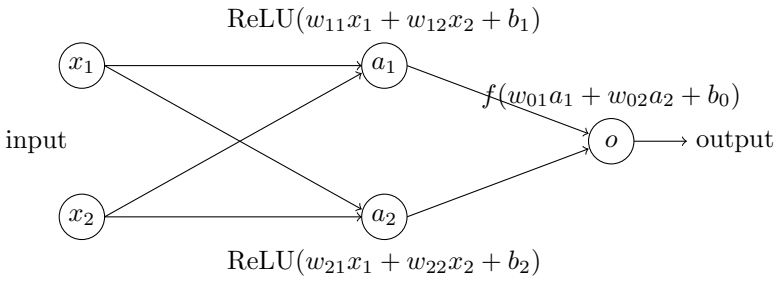
the cross-validated residual can be written as

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

(**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 $\text{ReLU}(u) = u$ 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.



hidden 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)

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

$\because P(\mathbf{X}_i \leqslant t)$ is i.i.d. $\therefore$ according Chebyshev Theorem: we can get:

$$\hat{F}_n(t) \xrightarrow{P} \frac{1}{n}\sum_{i=1}^{n} P(\mathbf{X}_{\leqslant}\mathbf{t}) \xrightarrow{P} \frac{1}{n} \times nP(\mathbf{X_i} \leqslant \mathbf{t}) = F(t)$$

(2) Suppose

$$J(\hat{\boldsymbol{\beta}}) = \|\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}\|^2$$

First we can simplify the optimization:

$$J(\hat{\boldsymbol{\beta}}) = (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})^T(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}) = \hat{\boldsymbol{\beta}}^\top \boldsymbol{X}\hat{\boldsymbol{\beta}} - 2\hat{\boldsymbol{\beta}}^\top \boldsymbol{X}^\top \boldsymbol{y} + \boldsymbol{y}^\top \boldsymbol{y}$$

so its gradient is:

$$\nabla_{\hat{\boldsymbol{\beta}}} J(\hat{\boldsymbol{\beta}}) = 2\boldsymbol{X}^\top \boldsymbol{X}\hat{\boldsymbol{\beta}} - 2\boldsymbol{X}^\top \boldsymbol{y}$$

let

$$\nabla_{\hat{\boldsymbol{\beta}}} J(\hat{\boldsymbol{\beta}}) = 0$$

we can get

$$\hat{\boldsymbol{\beta}}^* = (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{y}$$

If we use Ridge Regression: Suppose

$$J(\hat{\boldsymbol{\beta}}_\alpha) = \|\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}_\alpha\|^2 + \alpha\|\hat{\boldsymbol{\beta}}_\alpha\|^2$$

First we can simplify the optimization:

$$J(\hat{\boldsymbol{\beta}}_\alpha) = (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}_\alpha)^T(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}_\alpha) + \alpha\hat{\beta}_\alpha^T\hat{\beta}_\alpha = \hat{\boldsymbol{\beta}}_\alpha^\top \boldsymbol{X}\hat{\boldsymbol{\beta}}_\alpha - 2\hat{\boldsymbol{\beta}}_\alpha^\top \boldsymbol{X}^\top \boldsymbol{y} + \boldsymbol{y}^\top \boldsymbol{y} + \alpha\hat{\boldsymbol{\beta}}_\alpha^\top \hat{\boldsymbol{\beta}}_\alpha$$

so its gradient is:

$$\nabla_{\hat{\boldsymbol{\beta}}_\alpha} J(\hat{\boldsymbol{\beta}}_\alpha) = 2\boldsymbol{X}^\top \boldsymbol{X}\hat{\boldsymbol{\beta}}_\alpha - 2\boldsymbol{X}^\top \boldsymbol{y} + 2\alpha\hat{\boldsymbol{\beta}}_\alpha$$

let

$$\nabla_{\hat{\boldsymbol{\beta}}_\alpha} J(\hat{\boldsymbol{\beta}}_\alpha) = 0$$

we can get

$$\hat{\boldsymbol{\beta}}_\alpha^* = (\boldsymbol{X}^\top \boldsymbol{X} + \alpha\boldsymbol{I})^{-1}\boldsymbol{X}^\top \boldsymbol{y}$$

(b)

$$Bias^2(\hat{f}(\boldsymbol{x_0})) = \left[f(\boldsymbol{x}_0) - E\left(\hat{f}(\boldsymbol{x}_0)\right]^2$$

$$= (f(\boldsymbol{x}_0) - E(\boldsymbol{X}^\top(\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^T y))^2$$

$$\because Ey = E(f(\boldsymbol{x}_0) + \varepsilon) \quad = f(\boldsymbol{x}_0)$$

$$\therefore Bias^2(\hat{f}(\boldsymbol{x_0})) = \left(1 - \boldsymbol{x}_0^T\left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\right)^2 f^2(\boldsymbol{x}_0)$$

$$Var\left(\hat{f}(\boldsymbol{x}_0)\right) = \hat{f}(\boldsymbol{x}_0) - E\left(\hat{f}(\boldsymbol{x}_0)\right)^2$$

$$= (\boldsymbol{x}_0^T \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top (y - Ey))^2$$

$$= (\boldsymbol{x}_0^T \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \varepsilon)^2$$

$$\therefore EPE\left(x_0\right) = \sigma_\varepsilon^2 + \text{Bias}^2\left(\hat{f}\left(x_0\right)\right) + \text{Var}\left(\hat{f}\left(x_0\right)\right)$$

$$= \sigma_\varepsilon^2 + \left(1 - \boldsymbol{x}_0^T \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top\right)^2 f^2\left(\boldsymbol{x}_0\right) + (\boldsymbol{x}_0^T \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \varepsilon)^2$$

(c)

$$Bias^2(\hat{f}(\boldsymbol{x_0})) = \left[f\left(\boldsymbol{x}_0\right) - E\left(\hat{f}\left(\boldsymbol{x}_0\right)\right]^2\right.$$

$$= (f(\boldsymbol{x}_0) - E(\boldsymbol{X}^\top (\boldsymbol{X}^\top \boldsymbol{X} + \alpha \boldsymbol{I})^{-1} \boldsymbol{X}^T y)^2.$$

$$= \left(1 - \boldsymbol{x}_0^T \left(\boldsymbol{X}^\top \boldsymbol{X} + \alpha \boldsymbol{I}\right)^{-1} \boldsymbol{X}^\top\right)^2 f^2\left(\boldsymbol{x}_0\right)$$

$$Var\left(\hat{f}\left(\boldsymbol{x}_0\right)\right) = \hat{f}\left(\boldsymbol{x}_0\right) - E\left(\hat{f}\left(\boldsymbol{x}_0\right)\right)^2$$

$$= (\boldsymbol{x}_0^T \left(\boldsymbol{X}^\top \boldsymbol{X} + \alpha \boldsymbol{I}\right)^{-1} \boldsymbol{X}^\top (y - Ey))^2$$

$$= (\boldsymbol{x}_0^T \left(\boldsymbol{X}^\top \boldsymbol{X} + \alpha \boldsymbol{I}\right)^{-1} \boldsymbol{X}^\top \varepsilon)^2$$

$$\therefore EPE\left(x_0\right) = \sigma_\varepsilon^2 + \text{Bias}^2\left(\hat{f}\left(x_0\right)\right) + \text{Var}\left(\hat{f}\left(x_0\right)\right)$$

$$= \sigma_\varepsilon^2 + \left(1 - \boldsymbol{x}_0^T \left(\boldsymbol{X}^\top \boldsymbol{X} + \alpha \boldsymbol{I}\right)^{-1} \boldsymbol{X}^\top\right)^2 f^2\left(\boldsymbol{x}_0\right) + (\boldsymbol{x}_0^T \left(\boldsymbol{X}^\top \boldsymbol{X} + \alpha \boldsymbol{I}\right)^{-1} \boldsymbol{X}^\top \varepsilon)^2$$

(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

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

$$(\boldsymbol{A} - \boldsymbol{x}\boldsymbol{x}^\top)^{-1} = \boldsymbol{A}^{-1} + \frac{\boldsymbol{A}^{-1}\boldsymbol{x}\boldsymbol{x}^\top \boldsymbol{A}^{-1}}{1 - \boldsymbol{x}^\top \boldsymbol{A}^{-1}\boldsymbol{x}}$$

the cross-validated residual can be written as

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

Problem 2: Neural Networks

(1) It's clear that we can found when $w_{11} = w_{22} = 1, w_{12} = w_{21} = -1, b_1 = b_2 = 0$ and $w_{01} = 2_{02} = -1, b_0 = 1$, the neural network can output the function x_1 XOR x_2. This is because:
When $x_1 = x_2 = 0$,

$$a_1 = \text{ReLU}(x_{11}x_1 + w_{12}x_2 + b_1) = 0,$$

Similarly, $a_2 = a_1 = 0$, so the output is $f(1) = 1$.
When $x_1 = 0, x_2 = 1$,

$$a_1 = \text{ReLU}(x_{11}x_1 + w_{12}x_2 + b_1) = 0,$$

$$a_2 = \text{ReLU}(x_{21}x_1 + w_{22}x_2 + b_2) = 1.$$

The output is $f(w_{01}a_1 + w_{02}a_2 + b_0) = 0$.
Similarly, when $x_1 = 1, x_2 = 0$, the result is 0.
Now when $x_1 = x_2 = 1$,

$$a_1 = a_2 = 0.$$

The output is $f(1) = 1$. So the result is the definition of function XOR.
(2) For 2 input signals x_1, x_2, we have a total of $2^{2^2} = 16$ different binary functions h, say, $h_1, h_2, \cdots, h_{16}$. To implement these functions, we can simply use $16 \times 2 = 32$ hidden neurons, each pair of them can represent a specific function.
For example, if function h_1 is the XOR function, then we can implement it by simply setting the first to hidden neurons and the corresponding weights to the numbers in question (1) while keeping other 30 neurons' weights as 0 (not used). Similarly, other XOR-like functions can be implemented in this way.
This is because we know that each pair of 2-neurons sub-network can implement any binary non-linear functions like XOR or XNOR as well as linear functions like AND or OR.