Hw4

2022年4月14日　　下午 2:50

Homework 4

Machine Learning

Due on April 26 at 11:59AM on Canvas

Part I: Theory questions on Neural Networks

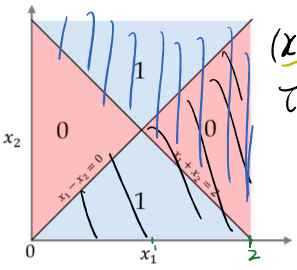
1. Perceptron models (8 pts)

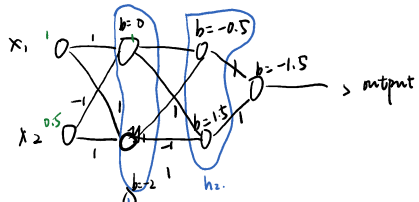
While the perceptron model has limitations in representing boolean functions such as XOR, we show that the XOR function can be represented by the perceptron model with an additional hidden layer using appropriate weights and bias. Suppose the inputs $x_1, x_2 \geq 0$, can you find a multi-layer perceptron that can represent the relationship between the inputs (x_1, x_2) and the label $y \in \{0, 1\}$ described in the following figure? Please describe such perceptron models and explain why it can represent the relationship between (x_1, x_2) and y. (You can ignore the points that sit exactly on the hyperplanes "$x_1 - x_2 = 0$" and "$x_1 + x_2 = 2$")

1. Solve: The result can be represented as:

$(x_1 - x_2 > 0)$ XOR $(x_1 + x_2 - 2 > 0)$

Thus, we can build the model:

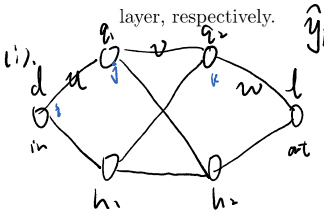




2. Back-Propagation algorithm (12 pts)

In the first hidden layer, y_1 and y_2 denotes the black dash and blue dash area in the figures respectively, the second hidden layer and the output layer is the XOR function in the slides, combined both we can get the result.

Consider a feedforward neural network with two hidden layers. There are d nodes in the input layer, q_i nodes in the hidden layer $i \in \{1, 2\}$, and l nodes in the output layers. Let $\{u_{i,j} : i = 1, \ldots, d, j = 1, \ldots, q_1\}$, $\{v_{ij} : i = 1, \ldots, q_1, j = 1, \ldots, q_2\}$, and $\{w_{ij} : i = 1, \ldots, q_2, j = 1, \ldots, l\}$ denote the weights on the arcs connecting the nodes in the input layer and the first hidden layer, the first hidden layer and the second hidden layers, and the second hidden layer and the output layer, respectively.



$\hat{y}_k$ can be derived from

$\hat{y}_k = f(f(f(x_k \cdot u) \cdot v) w)$ ⊛ where $f()$ is the activation function

$loss = \frac{1}{2} \| \hat{y}_k - y \|^2$.

$$\frac{d\,loss}{d\,u_{ij}} = \frac{d\,loss}{d\,\hat{y}_k} \cdot \frac{d\,\hat{y}_k}{d\,f(f(x_k \cdot u) \cdot v) w} \cdot \frac{d \cdots}{d\,f(f(x_k \cdot u) \cdot v)} \cdot \frac{d\,f(f(x_k \cdot u) \cdot v)}{d\,f(x_k \cdot u) \cdot v} \cdot \frac{d\,f(x_k \cdot u) \cdot v}{d\,f(x_k \cdot u)} \cdot \frac{d\,f(x_k \cdot u)}{d\,x_k \cdot u} \cdot \frac{d\,x_k \cdot u}{d\,u}$$

$= (\hat{y}_k - y) \cdot (1 - \hat{y}_k) \hat{y}_k \cdot w \cdot \cdots$.

$u_{ij}' = u_{ij} - \eta \frac{\partial loss}{\partial u_{ij}}$

$u_{ij} \to \sum_{k=1}^{q_1} v_{jk} \to \sum_{k=1}^{q_2} \sum_{i=1}^{l} w_{ki}$

Output error: $g = (y - \hat{y})(1 - \hat{y}) \cdot \hat{y}$, $(l \times 1)$.

error of node k in h_2: $e_k = \sum_{i=1}^{l} g \times w_{ki} \times out_{2k}$.

error of node j in h_1: $e_j = \sum_{k=1}^{q_2} e_k \times v_{jk} \times out_{ij}$

$u_{ij} \leftarrow u_{ij} + \eta \times e_j \times d_i = u_{ij} + \eta \sum_{k=1}^{q_2} \sum_{i=1}^{l} \left[(y - \hat{y})(1 - \hat{y}) \hat{y} \right] w_{ki} \cdot out_{2k} \cdot v_{jk} \cdot out_{ij} \times d_i$

(i) (6 pts) Assume that the activation function f is the sigmoid function and we use the square error as the loss function. Given a training sample $(\mathbf{x}_k, \mathbf{y}_k)$, how should we update the weight u_{ij} that is between the input layer and the first hidden layer based on the Back-Propagation algorithm?

(i) (6 pts) Assume that the activation function f is the sigmoid function and we use the square error as the loss function. Given a training sample $(\mathbf{x}_k, \mathbf{y}_k)$, how should we update the weight u_{ij} that is between the input layer and the first hidden layer based on the Back-Propagation algorithm?

(Hint: you may apply the chain rule to find the gradient, or make use of the observations that we have for the BP algorithm developed to train the neural networks with single hidden layer.)

(ii) (6 pts) Assume that the activation function f is a linear function, i.e., $f(x) = cx$ where $c > 0$ is a constant. Can you show that this feedforward neural network with two hidden layers is equivalent to a simple neural network with only the input layer and the output layer?

Part II: Hand-written digit recognition with Neural Networks

The MNIST (Modified National Institute of Standards and Technology) is a database of hand-written digits. The database contains more than 55000 handwritten digits for training and 10000 handwritten digits for testing.

One handwritten digit is represented as a 28×28 pixels image in gray scale. Thus mathematically it can be seen as an element in $R^{28\times 28}$. The handwritten digits come with labels as well. The labels tell which number the handwritten digit is supposed to represent. In this homework, you will implement Neural Networks in Python using Tensorflow to classify the handwritten digits.

Programming Assignments (35 pts)

In this programming assignment, you will first implement a simple neural network based on the demo code with TensorFlow. In the following questions, you will incrementally modify/improve the network architecture and observe the effects of those modifications on the accuracy. Finally, you can come up with your own neural network with the objective of further improving the accuracy.

For each step, you need to present the prediction accuracy on the training set and the test set and briefly discuss your observations in the report.

(i) (5 pts) Install the TensorFlow library, run the demo code, "NN_demo.py".