Multimedia II: Machine Learning & Computer Vision SS18

Assignment 01

Multimedia Computing Lab
Prof. Dr. Rainer Lienhart
M. Sc. Philipp Harzig
M. Sc. Dan Zecha

Submitting your solution is not mandatory. Your solutions will be discussed in the exercise on April 27th, 2018.

Exercise 1 (Calculating Derivatives)

Calculate the derivatives of the following functions for all variables:

1. $f(x) = \ln(4 - 2x) + \sqrt{3x + 9}$
2. $g(x) = e^{\frac{x}{x+1}}$
3. $h(x, y) = \sin(x)\,(y^2 + 1)\,\sqrt{x + 1}$

Exercise 2 (Least Mean Squares)

Show that the LMS weight update rule described in the lecture (2–01, Slide 23) performs a gradient descent to minimize the squared error.

In particular, define the squared error $E(b, V_{\text{train}}(b))$ for a given example b and its training value $V_{\text{train}}(b)$. Assume that $\hat{V}(b)$ is a linear combination of features as defined in the lecture. Now calculate the derivative of E with respect to the weight w_i. Gradient descent is achieved by updating each weight proportional to $-\frac{\partial E}{\partial w_i}$. Show that the LMS training rule alters weights in such a proportion for the given training example.

Exercise 3 (Hands-on experience: Tic Tac Toe)

In this exercise, you will design and implement a learning algorithm to play the game of Tic Tac Toe. Following the design to play checkers in the book:

1. Choose board features that characterize a certain state of the board.
2. Design a target function $V(b)$.
3. Design a target function approximation $\hat{V}(b)$ that is a linear combination of the board features.

For your convenience we have prepared a framework for this exercise in C++ which you can download on Digicampus. The code implements a game of TicTacToe between two players who are trained using your procedure. You should set the number of features you have specified in exercise 4.1 in the file TicTacToePlayer.h (search for the `//TODO` comment). After that, implement the following three functions in TicTacToePlayer.cpp:

1. `F = getBoardFeatures(b)`

   This function takes the current board state b and returns the values of your chosen board features as the vector F.

2. `b = makeMove(b)`

   Using the current function approximation represented by the vector of weights W, this function performs the best move on board b. The best move is the move that maximizes the score over all possible moves.

3. `W = learn(n)`

   This function learns the weight vector W representing the target function $\hat{V}$ by playing n games of Tic Tac Toe against itself. As explained in the book, update the weights according to the LMS update rule. The train value of intermediate board states is calculated as follows:

   $$V_{\text{train}}(b) = \hat{V}(\mathit{successor}(b)),$$

   where $\mathit{successor}(b)$ is the board state following b for which it is again the program's turn to move.

If you need to implement more helper-functions, feel free to do so. It is not necessary to alter the existing code for your program to work correctly.