

**本科生实验报告**

**实验课程 机器学习**

**学院名称 核技术与自动化工程学院**

**专业名称 测控技术与仪器**

**学生姓名 追梦少年南南**

**学生学号 202006010336**

**指导教师 吴旖旎**

**实验地点 6C702**

**实验成绩**

**二〇二〇年 月 二〇二〇年 月**

**填写说明**

1. 适用于本科生所有的实验报告（印制实验报告册除外）；
2. 专业填写为专业全称，有专业方向的用小括号标明；
3. 格式要求：
4. 用A4纸双面打印（封面双面打印）或在A4大小纸上用蓝黑色水笔书写。
5. 打印排版：正文用宋体小四号，1.5倍行距，页边距采取默认形式（上下2.54cm，左右2.54cm，页眉1.5cm，页脚1.75cm）。字符间距为默认值（缩放100%，间距：标准）；页码用小五号字底端居中。
6. 具体要求：

**题目**（二号黑体居中）；

**摘要**（“摘要”二字用小二号黑体居中，隔行书写摘要的文字部分，小4号宋体）；

**关键词**（隔行顶格书写“关键词”三字，提炼3-5个关键词，用分号隔开，小4号黑体)；

正文部分采用三级标题；

**第1章** ××(小二号黑体居中，段前0.5行)

**1.1** ×××××小三号黑体×××××（段前、段后0.5行）

**1.1.1**小四号黑体（段前、段后0.5行）

**参考文献**（黑体小二号居中，段前0.5行），参考文献用五号宋体，参照《参考文献著录规则（GB/T 7714－2005）》。

**BP神经网络学习**

**一、实验目的**

学习训练神经网络的过程，研究模型参数设置。

**二、内容**

1、完成三个函数sigmoidGradient.m，randInitializeWeights.m, nnCostFunction.com的填写，并调试成功。

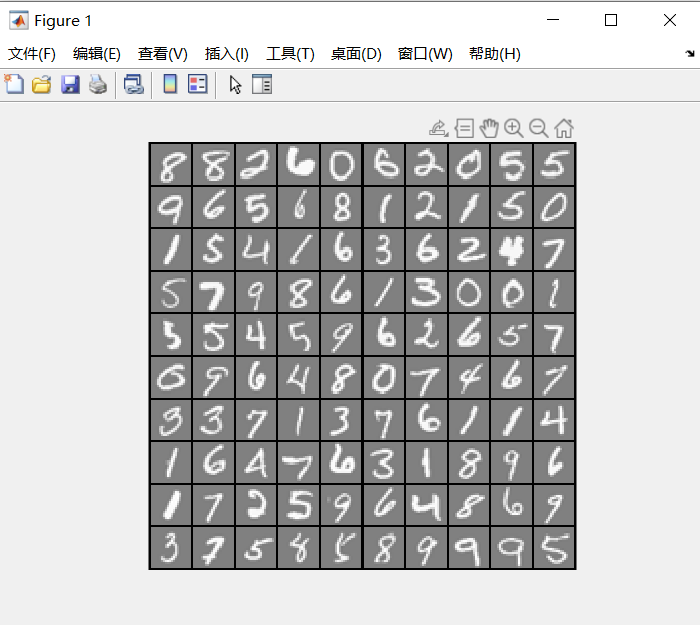
3、结合代码理解训练神经网络的步骤。

4、绘制该神经网络结构图。

**三、实验过程：**

1、理解数据，练习向量与矩阵的转换。

2、填写三个文件，并调试出结果。



文本

描述已自动生成

文本, 信件

描述已自动生成

文本

描述已自动生成

表格

中度可信度描述已自动生成

文本

描述已自动生成

文本

中度可信度描述已自动生成

手机屏幕截图

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

3、熟悉训练神经网络的步骤：

1）参数的随机初始化；

1. epsilon\_init = 0.12;

2. W = rand(L\_out, 1 + L\_in) \* (2 \* epsilon\_init) - epsilon\_init;%W里元素的范围为（-epsilon\_init,epsilon\_init)

2）利用正向传播方法计算所有的h\_θ (x);

1. X = [ones(m, 1) X];

2. temp = sigmoid(X \* Theta1');

3. temp = [ones(m, 1) temp];

4. h = sigmoid(temp \* Theta2');

5. temp\_y = zeros(size(h));

6. for i = 1:m

7. temp\_y(i, y(i)) = 1;

8. end

3）编写计算代价函数J的代码；

1. J = (1 / m) \* sum(sum(-temp\_y .\* log(h) - (1 - temp\_y) .\* log(1 - h)));

2. Theta1\_temp = Theta1(:, 2:end);

3. Theta2\_temp = Theta2(:, 2:end);

4. J = J + (lambda / 2 / m) \* (sum(sum(Theta1\_temp .\* Theta1\_temp)) + sum(sum(Theta2\_temp .\* Theta2\_temp)));

4）利用反向传播方法计算所有偏导数；

1. delta\_3 = h - temp\_y;

2. delta\_2 = delta\_3 \* Theta2 .\* sigmoidGradient([ones(m, 1) X \* Theta1']);

3. delta\_2 = delta\_2(:, 2:end);

4. D2 = delta\_3' \* temp;

5. D1 = delta\_2' \* X;

6. Theta2\_grad = 1/m \* D2;

7. Theta1\_grad = 1/m \* D1;

5）利用数值检验方法检验这些偏导数；

1. Theta1\_temp = Theta1;

2. Theta1\_temp(:, 1) = 0;

3. Theta2\_temp = Theta2;

4. Theta2\_temp(:, 1) = 0;

5. Theta1\_grad = Theta1\_grad + lambda / m \* Theta1\_temp;

6. Theta2\_grad = Theta2\_grad + lambda / m \* Theta2\_temp;

6) 使用优化算法来最小化代价函数。

1. if exist('options', 'var') && ~isempty(options) && isfield(options, 'MaxIter')

2. length = options.MaxIter;

3. else

4. length = 100;

5. end

6.

7.

8. RHO = 0.01; % a bunch of constants for line searches

9. SIG = 0.5; % RHO and SIG are the constants in the Wolfe-Powell conditions

10. INT = 0.1; % don't reevaluate within 0.1 of the limit of the current bracket

11. EXT = 3.0; % extrapolate maximum 3 times the current bracket

12. MAX = 20; % max 20 function evaluations per line search

13. RATIO = 100; % maximum allowed slope ratio

14.

15. argstr = ['feval(f, X']; % compose string used to call function

16. for i = 1:(nargin - 3)

17. argstr = [argstr, ',P', int2str(i)];

18. end

19. argstr = [argstr, ')'];

20.

21. if max(size(length)) == 2, red=length(2); length=length(1); else red=1; end

22. S=['Iteration '];

23.

24. i = 0; % zero the run length counter

25. ls\_failed = 0; % no previous line search has failed

26. fX = [];

27. [f1 df1] = eval(argstr); % get function value and gradient

28. i = i + (length<0); % count epochs?!

29. s = -df1; % search direction is steepest

30. d1 = -s'\*s; % this is the slope

31. z1 = red/(1-d1); % initial step is red/(|s|+1)

32.

33. while i < abs(length) % while not finished

34. i = i + (length>0); % count iterations?!

35.

36. X0 = X; f0 = f1; df0 = df1; % make a copy of current values

37. X = X + z1\*s; % begin line search

38. [f2 df2] = eval(argstr);

39. i = i + (length<0); % count epochs?!

40. d2 = df2'\*s;

41. f3 = f1; d3 = d1; z3 = -z1; % initialize point 3 equal to point 1

42. if length>0, M = MAX; else M = min(MAX, -length-i); end

43. success = 0; limit = -1; % initialize quanteties

44. while 1

45. while ((f2 > f1+z1\*RHO\*d1) || (d2 > -SIG\*d1)) && (M > 0)

46. limit = z1; % tighten the bracket

47. if f2 > f1

48. z2 = z3 - (0.5\*d3\*z3\*z3)/(d3\*z3+f2-f3); % quadratic fit

49. else

50. A = 6\*(f2-f3)/z3+3\*(d2+d3); % cubic fit

51. B = 3\*(f3-f2)-z3\*(d3+2\*d2);

52. z2 = (sqrt(B\*B-A\*d2\*z3\*z3)-B)/A; % numerical error possible - ok!

53. end

54. if isnan(z2) || isinf(z2)

55. z2 = z3/2; % if we had a numerical problem then bisect

56. end

57. z2 = max(min(z2, INT\*z3),(1-INT)\*z3); % don't accept too close to limits

58. z1 = z1 + z2; % update the step

59. X = X + z2\*s;

60. [f2 df2] = eval(argstr);

61. M = M - 1; i = i + (length<0); % count epochs?!

62. d2 = df2'\*s;

63. z3 = z3-z2; % z3 is now relative to the location of z2

64. end

65. if f2 > f1+z1\*RHO\*d1 || d2 > -SIG\*d1

66. break; % this is a failure

67. elseif d2 > SIG\*d1

68. success = 1; break; % success

69. elseif M == 0

70. break; % failure

71. end

72. A = 6\*(f2-f3)/z3+3\*(d2+d3); % make cubic extrapolation

73. B = 3\*(f3-f2)-z3\*(d3+2\*d2);

74. z2 = -d2\*z3\*z3/(B+sqrt(B\*B-A\*d2\*z3\*z3)); % num. error possible - ok!

75. if ~isreal(z2) || isnan(z2) || isinf(z2) || z2 < 0 % num prob or wrong sign?

76. if limit < -0.5 % if we have no upper limit

77. z2 = z1 \* (EXT-1); % the extrapolate the maximum amount

78. else

79. z2 = (limit-z1)/2; % otherwise bisect

80. end

81. elseif (limit > -0.5) && (z2+z1 > limit) % extraplation beyond max?

82. z2 = (limit-z1)/2; % bisect

83. elseif (limit < -0.5) && (z2+z1 > z1\*EXT) % extrapolation beyond limit

84. z2 = z1\*(EXT-1.0); % set to extrapolation limit

85. elseif z2 < -z3\*INT

86. z2 = -z3\*INT;

87. elseif (limit > -0.5) && (z2 < (limit-z1)\*(1.0-INT)) % too close to limit?

88. z2 = (limit-z1)\*(1.0-INT);

89. end

90. f3 = f2; d3 = d2; z3 = -z2; % set point 3 equal to point 2

91. z1 = z1 + z2; X = X + z2\*s; % update current estimates

92. [f2 df2] = eval(argstr);

93. M = M - 1; i = i + (length<0); % count epochs?!

94. d2 = df2'\*s;

95. end % end of line search

96.

97. if success % if line search succeeded

98. f1 = f2; fX = [fX' f1]';

99. fprintf('%s %4i | Cost: %4.6e\r', S, i, f1);

100. s = (df2'\*df2-df1'\*df2)/(df1'\*df1)\*s - df2; % Polack-Ribiere direction

101. tmp = df1; df1 = df2; df2 = tmp; % swap derivatives

102. d2 = df1'\*s;

103. if d2 > 0 % new slope must be negative

104. s = -df1; % otherwise use steepest direction

105. d2 = -s'\*s;

106. end

107. z1 = z1 \* min(RATIO, d1/(d2-realmin)); % slope ratio but max RATIO

108. d1 = d2;

109. ls\_failed = 0; % this line search did not fail

110. else

111. X = X0; f1 = f0; df1 = df0; % restore point from before failed line search

112. if ls\_failed || i > abs(length) % line search failed twice in a row

113. break; % or we ran out of time, so we give up

114. end

115. tmp = df1; df1 = df2; df2 = tmp; % swap derivatives

116. s = -df1; % try steepest

117. d1 = -s'\*s;

118. z1 = 1/(1-d1);

119. ls\_failed = 1; % this line search failed

120. end

121. if exist('OCTAVE\_VERSION')

122. fflush(stdout);

123. end

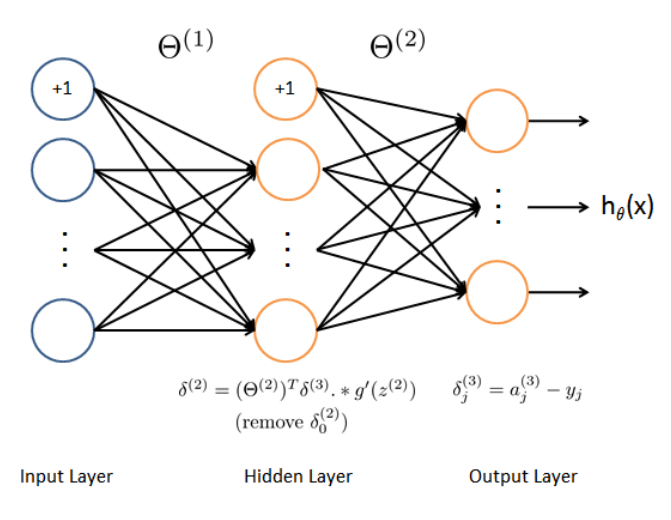
124. end

125. fprintf('\n');

126.

把相应的程序复制到对应的步骤。

4、绘制出网络结构图。



其中：输入层为400个单元；隐层为25个单元，输出层为10个单元。

|  |  |
| --- | --- |
| **学生实验 心得** | 机器学习的实验带给了我一种全新的实验模式，我在感到新奇之余认真的体验了下这种实验模式。本次的实验内容相对较难，我在与同学讨论之余体会了这次实验。通过本次实验，我掌握了神经网络的一些基本知识，例如神经网络模型实现、梯度下降算法等等。今后的学习中，我也会强化关于机器学习部分的相关知识。  学生（签名）：  2022年12月20日 |
| **指导**  **教师**  **评语** | 该同学理论基础 ，实验过程中，主动性  ， 独立完成实验内容， 了相关的原理和方法，实验结果 ，对思考题的理解和回答 ，报告书写 认真。  成绩评定：  指导教师（签名）：  2020 年 月 日 |