261499: Selected Topic in CI **Semester 1, 2018**

Assignment 2: Feedforward Neural Network

Lecturer: Kasemsit Teeyapan *Date: 21 September 2018*

Instruction

Due date: 9 October 2018

List of files

1. `problem2/feedforward_nn.py` – L-layer feedforward model**
2. `problem2/testcases.py` – Test case generator
3. `problem2/testcases.txt` – Correct test cases
4. `problem2/classify_cat.py` – Cat/Non-cat classification**
5. `problem2/datasets/test_catvnoncat.h5` – Cat/Non-cat training dataset
6. `problem2/datasets/train_catvnoncat.h5` – Cat/Non-cat testing dataset
7. `problem2/classify_2d.py` – Artifical 2D data classification**

(** = The files you have to modify.)

How to submit this assignment?

1. Submit the answers to this worksheet in the class (*in separate paper*)
2. Submit all the completed assignment files to `kasemsit@gmail.com` with the email's title "261499 Assignment 2 Submission". Please also state your name and student ID number.

Problem 2: Feedforward neural network

Question 2.1: Activation function $\tanh(z)$

1. [2 pts] Show that $\tanh(z)$ can be written in term of the sigmoid function $\sigma(z)$ as $\tanh(z) = 2\sigma(2z) - 1$.

2. [2 pts] Show that the derivative of the activation function $a = \tanh(z)$ is $\frac{da}{dz} = 1 - a^2$.

Question 2.2: Implementation of L-layer feedforward neural network

3. [35 pts] Design an L-layer feedforward neural network by completing "`feedforward_nn.py`". The network will have the sigmoid function for the output layer. All nodes in the hidden layer (the first $L-1$ layer) will use

   - *only* ReLU activation function or
   - *only* tanh activation function.

   Hint 1: The main section in "`feedforward_nn.py`" will print test cases for some functions. The correct test cases are provided in "`testcase.txt`" to help verify your codes.

   Hint 2: The sigmoid function $\sigma(z)$ has the derivative $\sigma'(z) = \sigma(z)(1-\sigma(z))$.

   Hint 3: The ReLU function $a = \max\{0, z\}$ has the derivative

$$\frac{da}{dz} = \begin{cases} 0, & z \leq 0 \\ 1, & \text{otherwise.} \end{cases} \tag{2.1}$$

   Notice that the derivative is, in fact, undefined at $z = 0$, but we set it to 0.

   Hint 4: In "`feedforward_nn.py`", all cost gradients are denoted using `d` as a prefix. For example, $\boldsymbol{\delta}_{\mathbf{Z}} = \frac{\partial J}{\partial \mathbf{Z}}$ is `dZ`.

   <u>Forward propagation of m examples:</u>

$$\mathbf{Z}^{[\ell]} = \mathbf{W}^{[\ell]}\mathbf{A}^{[\ell-1]} + \mathbf{B}^{[\ell]} \quad \text{with} \quad \mathbf{B}^{[\ell]} = \overbrace{\begin{bmatrix} \mathbf{b}^{[\ell]} & \cdots & \mathbf{b}^{[\ell]} \end{bmatrix}}^{m \text{ columns}} \tag{2.2}$$

$$\mathbf{A}^{[\ell]} = g(\mathbf{Z}^{[\ell]}) \qquad \text{(element-wise)} \tag{2.3}$$

   <u>Backward propagation of m examples:</u>

$$\boldsymbol{\delta}_{\mathbf{Z}^{[\ell]}} = \boldsymbol{\delta}_{\mathbf{A}^{[\ell]}} \odot g'(\mathbf{Z}^{[\ell]}) \quad \in \mathbb{R}^{n^{[\ell]} \times m} \tag{2.4}$$

$$\boldsymbol{\delta}_{\mathbf{W}^{[\ell]}} = \frac{1}{m}\boldsymbol{\delta}_{\mathbf{Z}^{[\ell]}}\mathbf{A}^{[\ell-1]T} \quad \in \mathbb{R}^{n^{[\ell]} \times n^{[\ell-1]}} \tag{2.5}$$

$$\boldsymbol{\delta}_{\mathbf{b}^{[\ell]}} = \frac{1}{m}\boldsymbol{\delta}_{\mathbf{Z}^{[\ell]}}\mathbf{1}_{m\times 1} \quad \in \mathbb{R}^{n^{[\ell]}} \quad (\boldsymbol{\delta}_{\mathbf{Z}^{[\ell]}}\mathbf{1}_{m\times 1} = \text{row sum of } \boldsymbol{\delta}_{\mathbf{Z}^{[\ell]}}) \tag{2.6}$$

$$\boldsymbol{\delta}_{\mathbf{A}^{[\ell-1]}} = \mathbf{W}^T\boldsymbol{\delta}_{\mathbf{Z}^{[\ell]}} \quad \in \mathbb{R}^{n^{[\ell-1]} \times m} \tag{2.7}$$

   where $\mathbf{1}_{m\times 1} \in \mathbb{R}^m$ is the vector of ones with m elements.

   Hint 5: When compute the cost at the output layer (Layer L), use the cost of logistic regression, i.e.

$$J = -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\log\left(a^{[L](i)}\right) + (1-y^{(i)})\log\left(1-a^{[L](i)}\right) \tag{2.8}$$

   This is called a cross-entropy cost.

   The derivative of the cost J (Eq. (2.8)) with respect to the output $a^{[L](i)}$ (Layer L and example i) is

$$\frac{\partial J}{\partial a^{[L](i)}} = -\frac{y^{(i)}}{a^{[L](i)}} + \frac{1-y^{(i)}}{1-a^{[L](i)}} \tag{2.9}$$

Hint 6: Gradient descent's update equation for parameter θ is

$$\theta := \theta - \alpha \frac{\partial J}{\partial \theta} \tag{2.10}$$

where α is the learning rate.

Hint 7: The output $a^{[L]}$ from the output layer (sigmoid activation) is between 0 and 1. To classify it, we simply threshold it, i.e.

$$\hat{y} = \begin{cases} 1, & a^{[L]} > 0.5 \\ 0, & \text{otherwise.} \end{cases} \tag{2.11}$$

Hint 8: Accuracy is simply defined by

$$\frac{\text{Number of correct predictions}}{\text{Total number of predictions}}. \tag{2.12}$$

Question 2.3: Applications

Use the completed L-layer feedforward neural network from 3. to perform binary classifications. ($n =$ the number of features of the input).

4. Cat vs. Non-cat using "`classify_cat.py`":

   (a) [2 pts] Logistic regression with
   `layer_dims=(n, 1), learning_rate=0.005, num_iterations=2400`.

   Classification accuracy on training set = ____________________

   Classification accuracy on test set = ____________________

   (b) [2 pts] Two-layer neural network with ReLU function for Layer 1 and sigmoid function for the output layer.
   `layer_dims=(n, 7, 1), learning_rate=0.0075, num_iterations=2400`

   Classification accuracy on training set = ____________________

   Classification accuracy on test set = ____________________

   (c) [2 pts] 4-layer neural network with ReLU function for the hidden layers and sigmoid function for the output layer.
   `layer_dims=(n, 20, 7, 5, 1), learning_rate=0.0075, num_iterations=2400`

   Classification accuracy on training set = ____________________

   Classification accuracy on test set = ____________________

(d) [0 pts] Observe how the accuracy improve from (a) – (c).

5. [4 pts] Classify 2D data using "`classify_2d.py`" by training the **two-layer** neural network with `layer_dims = (n, 4, 1), learning_rate=1.2, num_iterations=10000`. Use **tanh** activation function for the first $L - 1$ layers (hidden layers) for all nodes. For the output layer, use the same sigmoid activation function (and cost) defined in 3.

   Classification accuracy on training set = ________________________

   Classification accuracy on test set = ________________________