Foundations of Deep Learning (AMMI)

Assignment 1

Due: Tuesday 13 November 2018 at 08:00

---

1 Nonlinear Activation Functions

Logistic function (sigmoid), hyperbolic tangent, and rectified linear unit (ReLU) are commonly used activation functions in deep learning. They are defined as:

`sigmoid`:

$$y = \sigma(x) = \frac{1}{1 + \exp(-x)}, \tag{1}$$

`tanh`:

$$y = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} = \frac{\exp(2x) - 1}{\exp(2x) + 1}, \tag{2}$$

`ReLU`:

$$y = (x)^+ = \max(0, x), \tag{3}$$

where x is the input scalar and y is the output scalar. Assume the error back-propagated to y is $\frac{\partial E}{\partial y}$. For each activation function, write the expression for $\frac{\partial E}{\partial x}$ in terms of $\frac{\partial E}{\partial y}$.

2 Vanishing and Exploding Gradients

Consider a fully-connected linear network with 51 layers (that considers input layer as one layer), where each layer has the *same* square $n \times n$ weight matrix $\boldsymbol{W}$. Inputs are vectors $\boldsymbol{x} \in \mathbb{R}^n$ and outputs are vectors $\boldsymbol{y} \in \mathbb{R}^n$, and the value at hidden layer k is $\boldsymbol{h}_k = \boldsymbol{W}^k \boldsymbol{x}$. Let $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ be the singular value decomposition of $\boldsymbol{W}$.

Assume that we are given the vector $\frac{\partial E}{\partial \boldsymbol{y}}$, *i.e.* the gradient of the error with respect to the output, and that $\left\| \frac{\partial E}{\partial \boldsymbol{y}} \right\|_2 = 1$.

1. Write down $\frac{\partial E}{\partial \boldsymbol{h}_{k-1}}$ in terms of $\frac{\partial E}{\partial \boldsymbol{h}_k}$ and $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$, for $1 \leq k \leq 50$.

2. Let $\mathbf{\Sigma} = \mathbb{I}$, the identity matrix. What is the ℓ_2 norm of $\frac{\partial E}{\partial \boldsymbol{h}_1}$, *i.e.* the norm of the gradient with respect to the first layer hidden units? Explain why.

3. Let $\mathbf{\Sigma} = \frac{1}{2}\mathbb{I}$. What is the ℓ_2 norm of $\frac{\partial E}{\partial \boldsymbol{h}_1}$? Explain why.

4. Let $\mathbf{\Sigma} = 2\mathbb{I}$. What is the ℓ_2 norm of $\frac{\partial E}{\partial \boldsymbol{h}_1}$? Explain why.

3 Sentence classification

Sentence classification is a common problem in neural language processing (NLP). There are many ways to deal with this problem. As simplest, you could just use machine learning algorithms such as logistic regression, or any you can think of, such as a multi-layer perceptron (MLP). You could even get help from CNNs or RNNs, which can give you good sentence vectors, which you can feed into a linear classification layer. The goal of this question is to make sure that you have a clear picture in your mind about all these possible techniques.

Let's say you have a corpus of 50k words. For the simplicity of this question, assume you have the trained word embedding matrix of size 50k × 300 available to you, which can give you a word vector of size 1×300. Consider one sentence having 10 words for the classification task and describe your approach for the techniques below.

(a) CNN

1. Design a one-layer CNN which first maps the sentence to a vector of length 5 (with the help of convolution and pooling), then feeds this vector to a fully connected layer with soft(arg)max to get the probability values for possible 3 classes.

2. Clearly mention the sizes for your input, kernel, outputs at each step (till you get the final 3×1 output vector from soft(arg)max).

3. Please describe the effect of small filter size *vs.* large filter size during the convolution? What would be your approach to select the filter sizes for classification task?

(b) RNN

1. How can a simple RNN which is trained for language modelling be used to get the sentence vector?

2. Design a simple RNN which first maps the sentence to a vector of length 50, then feeds this vector to fully connected layer with soft(arg)max to get the probability values for possible 4 classes.

3. Clearly mention the sizes of all the RNN components such as your input vector, hidden layer weight matrix, hidden state vectors, cell state vector, output layers (RNN components sizes would be same at each time step).

4 Image classification

You would be using the MNIST dataset and building a CNN to classify the digits 0 to 9 in the dataset. In our practical session, the code we demonstrated uses hard-coded tensor sizes. However, this is not scalable when you expand your model's capacity through a more complex architecture (deeper, wider, etc.). In this programming exercise, we will be creating a class so we can change arguments easily without manually calculating our tensor sizes and hard-coding them into our code.

4.1 New Model Class

You are required to create a model that has the following architecture: (1) input, (2) convolution, (3) pooling, (4) convolution, (5) pooling, (6) linear, and finally (7) soft(arg)max.

Instead of a simple model class we demonstrated in class (please refer to that class to get a sense of what is expected), you are required to add a few more arguments and modify the class accordingly. To keep this exercise simpler, we will be using a fixed number of convolution kernels of 16 and 32 for the first and second convolutional layers respectively in the architecture highlighted above.

`CNN(**kwargs)`

1. `conv_kernel_size (int)`

   This is the size of the convolution kernel. In our class, we have used the size of 3.

2. `pooling_kernel_size (int)`

   This is the size of the pooling kernel.

3. `stride_size (int)`

   This is the stride size of the kernel. We have covered only a stride size of 1 in class.

4. `zero_padding (bool)`

   If True, the model uses zero padding (also called same padding). Else, the model uses valid padding (no padding).

5. `max_pooling (bool)`

   If True, the model uses Max Pooling. Else the model uses Average Pooling.

4.2 Zero Padding Model Tensors

After you have defined your model's class, you are required to instantiate the model's class with the following values for your new arguments and assign it to an object. Print out all of the model's parameters.

For the convolutional kernel size, the size of 5 implies a 5 by 5 kernel size, this logic applies to the pooling kernel size too.

```
model = CNN(**kwargs)
```

1. `conv_kernel_size (int) = 5`
2. `pooling_kernel_size (int) = 2`
3. `stride_size (int) = 1`
4. `zero_padding (bool) = True`
5. `max_pooling (bool) = True`

4.3 Valid Padding Model Tensors

Here we will do the same task as above but with valid padding (no padding). Print out all of the model's parameters.

```
model = CNN(**kwargs)
```

1. `conv_kernel_size (int) = 5`
2. `pooling_kernel_size (int) = 2`
3. `stride_size (int) = 1`
4. `zero_padding (bool) = False`
5. `max_pooling (bool) = True`

Intructions

For Section 1 to 3, you can write down your solutions on paper. However, for Section 4, you are required to send us your Jupyter Notebook for grading.

Submission

Evaluation

Your grade for this assignment will be based on:

- 10% - Section 1.
- 15% - Section 2.
- 25% - Section 3a.
- 25% - Section 3b.
- 25% - Section 4 (programming).

Submission

Send your notebook to ritchieng@u.nus.edu and pass your solutions to Ritchie at the start of Tuesday's class. When you send your notebook, please use the following format as your email header.

```
[Name Surname] Submission HW1
```