[HW#3]

AutoEncoder to encode and decode protein sequence

ML Bioinformatics, SNU 2018 Fall

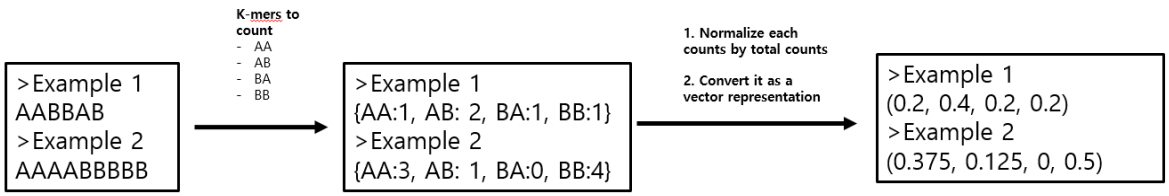
In the assignment3, we will have a chance to implement an autoencoder that encodes and decodes k-mer vector of protein sequences. Objective of this homework is to understand how autoencoder works by implementing it by hands.

1. Input & Pre-processing

As has already been done in assignment2, we will process COG protein sequences: COG1.fasta, COG160.fasta and COG161.fasta. In this assignment, however, we will consider k-mer vectorized version of COG sequences, which considers k-mer counts as vectorized representation of each COG sequences, as an input to an Autoencoder.

So, first tasks to do is input processing, converting each protein sequences in COG data file as k-mer counted vector. For detailed explanation, I will give short example for input-preprocessing. Here, for simplicity, we will consider only 'A' and 'B' for character for protein sequence characterset and we will consider k-mer of size 2.

<Example for Pre-processing>



<TODO : Pre-processing procedures>

1. Count occurrence of all possible k-mers for each sequence
2. Normalize count statistics by dividing each counts by total k-mer counts.
3. Convert normalized count statistics as vector representation.

<TIP>

We recommend you to record pre-processed vectors in a single file, which can be read as a numpy matrix. It will be a lot easier to feed training data to autoencoder. You may refer the tensorflow mnist tutorials to get hints for how to feed in data into neural networks.

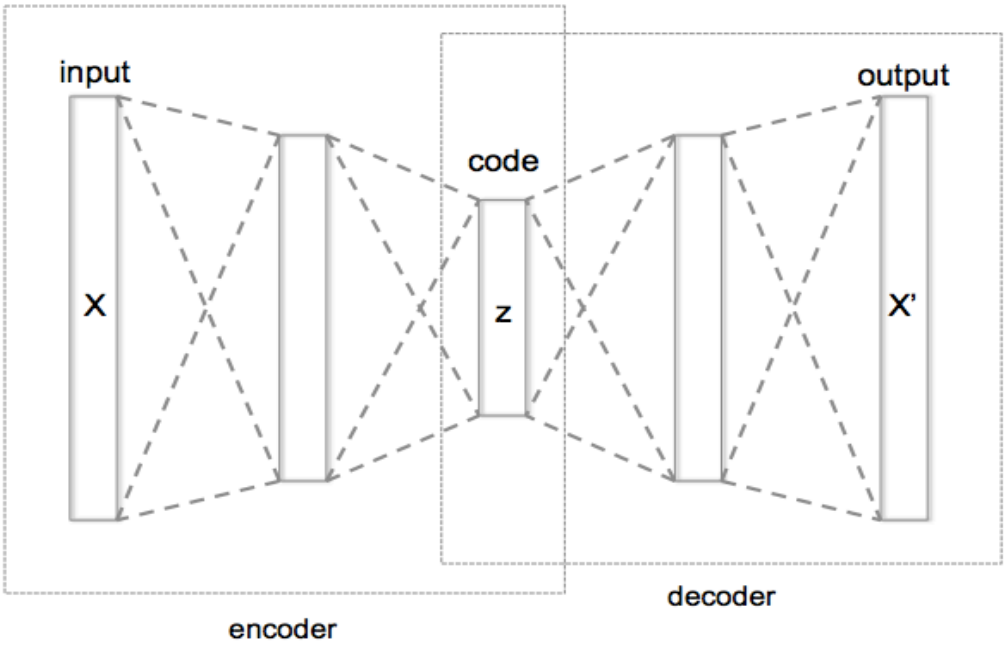
For the Implementation, you can set following properties for protein sequences.

- Charset : 'A','C','D','E','F','G','H','I','K','L','M','N','P','Q','R','S','T','V','W','Y'

- K_mer size : 2

2. Method

You will now build your own autoencoder, which encodes k-mer vectors of each protein sequence into low-dimensional code vector and decodes each code vector into original k-mer vectors. Input data to the network will be resulting pre-processed vectors from **Input&Pre-processing parts**. You may use any machine learning/deep learning libraries including *Tensorflow*, *PyTorch*, … etc.



<TODO : Constructing Networks>

- Construct neural network layers which can work on our pre-processed data.
- Define loss function, which optimizes your networks to do auto-encoding works for our data.
- Make your code to extract encoded vector in autoencoder and record every encoded vector.

3. Submission

- **Codes for pre-processing input data**
- **Codes for train autoencoder and run autoencoder**
  It is a good practice to split tensorflow/pytorch codes into multiple files : train.py, test.py, run.py, util.py, … etc. However, as most of you are new to deep learning or corresponding libraries, regarding the codes for autoencoder, it is perfectly okay to put all the relevant codes into one file.
- **Documentation on how to run the code**
  You should submit documentation on how to run your codes. This documentation should contain commands for the following taks : 1) pre-process COG data
  2) train autoencoder
  3) run autoencoder to extract encoded vector

  e.g)

Codes to run autoencoder and extract encoded vector for sequences in “test.fasta”. $ python run_autoencoder.py test.fasta

If none of above is stated in the documentation, we will not give any points for this assignment. Even if your code is not executable due to some errors, you should write above commands as it gives guides to understanding what your intentions regarding the code. It will help us give you partial points for this assignments.

It is highly recommended to explain how your model works or how your logic works and it will give us an opportunity to give you partial points when your codes doesn't work.

You should put codes and documentation file into directory named 'StudentID_assign3' and compress it will 'tar –c(v)zf *directory_name.tar.gz directory_name*' and then submit it.

Submission to : leetae0130@snu.ac.kr

You can implement this homework with C, C++, python, java, or any language. To run your program, submit your code with document containing how to compile and run in detail, please.

Due

6th December 2018, 11.59pm.
10% penalty per 1 day delay