Sharif University of Technology
Computer Engineering Department
Ali Sharifi-Zarchi

# Homework Assignment 4
Machine Learning for Bioinformatics,
Spring 2023

1402-03-21

Amirmohammad Isazadeh, Ghazal Farahani, Bahar Oveis, Ainaz Rafiei

## 1 Convolutional Neural Networks

### 1.1

output dimensions: $128 \times 128 \times 16$ number of parameters: $(5 \times 5 \times 3 + 1) \times 16 = 76 \times 16 = 1216$

### 1.2

output dimensions: $16 \times 16 \times 16$ number of parameters: $(5 \times 5 \times 3 + 1) \times 16 + (5 \times 5 \times 16 + 1) \times 16 + (5 \times 5 \times 16 + 1) \times 16 = 14048$

### 1.3

$14048 + 16 \times 16 \times 16 \times 10 = 55008$

## 2 Generative Adversarial Network (GAN)

### 2.1

VAE is a type of Autoencoder that can be used to generate new data. In such a way that each point of its latent space that is selected can be decoded and ultimately lead to the production of new data. To ensure that each point selected from the latent space is ultimately decoded into meaningful data, it is necessary to regularize the latent space. This means that during training, this space should be selected in such a way that each point of it is meaningful and also points close to each other in this space will eventually lead to the production of similar data. For this purpose, in VAEs, instead of mapping each sample of training data to a point in the latent space, a distribution over the latent space will be used during training to prevent overfitting on data and obtain a regularized latent space.
The GAN model is also used to generate new data. In such a way that it consists of two components called discriminator and generator and these two components compete with each other so that ultimately the generator learns to produce new data. This competition is such that the discriminator is responsible for distinguishing between generated data and real data, and the generator is responsible for producing synthetic data that are as similar as possible to real data so that it can fool the discriminator.
Given the definitions mentioned above, we realize that although VAEs and GANs both are used for generating new data, they have completely different architectures and structures. VAEs are easier to train than GANs and require less data for training. On the other hand, with VAEs, it is possible to obtain the distribution of data on the latent space. In addition, with their help, representations of data can be obtained on a reduced dimensionality space which helps better understanding of data and helps in saving them in less space as well as being more interpretable. However, the generated data by VAEs are not less clear or lower quality than those generated by GANs. Although GANs require more data for training and their training is harder.

### 2.2

The images produced by GANs have higher quality and clarity than VAEs. In other words, images generated by VAEs are slightly blurred but GANs produce images that have good sharpness.
This is because in VAEs, the objective function is of the type Maximizing likelihood which is determined specifically and is simpler; whereas in GANs, the objective function is learned by the generator in a way that can be much more complex. Also, since in VAEs it is necessary to learn the distribution of data in latent space (a space with fewer dimensions), this causes all details not to be able to be brought into latent space (a bottleneck is created) and the distribution of data in this space is usually simpler and VAEs cannot model complex distributions of data in this space. This is while in GANs there is a more complex objective function and there is no limitation on modeling data in latent space. This makes GANs more free and able to learn more complex data distributions.
Another issue that limits VAEs is the default use of Gaussian distributions and their combination to model data distributions and the use of l2 loss which causes images to become blurred.

### 2.3

We should only use left-right flip augmentation, as the other ones will create artifacts in the generated images like having unreal colors or generating blurred images.

### 2.4

In WGAN, instead of using a discriminator that estimates the probability of real or fake input data, a discriminator is used that gives a score as the degree of realness or fakeness to the input data. The goal is to reduce the distance between the distribution of real data and the distribution of generated data. The loss function in WGAN changes compared to GAN to help solve the Vanishing Gradients problem. By doing so, the discriminator moves towards an optimal state without any problems for updating the generator weights and consequently learning it. By obtaining an optimal discriminator, if the generator goes towards generating repetitive data, the discriminator realizes it and rejects that repetitive data (gives it a bad score); as a result, the generator is forced to generate different data.

## 3 Autoencoders

### 3.1

The main reason is that due to the use of the sigmoid activation function and the fact that the model is relatively deep (7 layers is too much for an autoencoder), we experience gradient vanishing and the parameters of the model cannot be updated properly. The result is that training loss is not reduced. Of course, another reason could be that we did not initialize the model well. For example, if we consider all parameters of the model to be zero, then the model cannot learn and the amount of training loss will not decrease

### 3.2

That the said error value is less than 0.1 is a relatively good number. But we must note that the test set we use may be a special case that even if our model is not suitable enough, it will still give an acceptable answer. In fact, we should also check the training error and compare its value with the test error. If the training error is significantly less than the test error, then we realize that a sampling bias has occurred in our test set. And that in general we can use other metrics like MAE and see if the result is acceptable for these metrics as well or not.

### 3.3

In this case, what most likely happened is that the numerical value of the model parameters is too large. To solve this problem, we can add a suitable regularization sentence to prevent model parameters from becoming too large and to converge to more correct values.

### 3.4

As we can see, despite the fact that the input data has a large number of features, in the end, by using 20 neurons, we can extract a suitable latent representation for them, which means that the features of the input data are not independent of each other, and in fact, there is a good correlation. In another sense, their variance is low and they have stored little information.

## 4 Beam Search Encoding

### 4.1

Encoding is the process of transforming a sequence of words or symbols into a vector representation that can be used by a neural network. Encoding is important for the Beam Search algorithm because it allows the algorithm to compare different possible translations or outputs based on their probabilities and similarities.
The Beam Search algorithm is a heuristic search algorithm that explores a graph by expanding the most promising node in a limited set. It uses a beam width parameter to control how many nodes are kept at each level of the search tree. The beam width bounds the memory required to perform the search but also affects the completeness and optimality of the algorithm.
A common application of Beam Search is in encoder-decoder models such as machine translation. In this case, the encoder transforms the source language into an intermediate representation, and the decoder generates the target language using Beam Search. The decoder starts with a token that signals the beginning of a

sequence and then expands it with all possible words from the vocabulary. It then selects the best words based on their probabilities and continues to expand them until it reaches the end of the sequence or a predefined length limit. The final output is the most probable sequence among all the candidates.
Encoding and Beam Search work together to produce high-quality translations or outputs that are close to human performance. Encoding captures the meaning and structure of the source language, and Beam Search explores different ways of expressing it in the target language. By using encoding and Beam Search, encoder-decoder models can handle complex and diverse natural language tasks.

### 4.2

The encoding step is incorporated into the overall process of Beam Search by transforming the source sequence into an intermediate representation that can be used by the decoder. The encoder can be a recurrent neural network (RNN) or a transformer that maps the source words or symbols into a vector of numbers. The encoder output is then passed to the decoder as an input along with a start-of-sequence token.
The encoding step impacts the generated sequences by capturing the meaning and structure of the source sequence and providing context for the decoder. The quality of the encoding affects the probability of the words and phrases that are possible in translation or output. A good encoding can help the Beam Search algorithm to explore different ways of expressing the source sequence in the target language and select the most probable ones based on the beam width parameter.

### 4.3

The encoding step in sequence generation is the process of transforming an input sequence into a fixed-length vector representation that captures its semantic information. The quality, fluency, and relevance of the generated sequences depend on how well the encoder can encode the input sequence. A good encoder should be able to preserve the important features of the input and avoid information loss or distortion.
The performance of the beam search algorithm, which is a technique for finding the most likely output sequence given an encoded input, depends on the encoding complexity and the beam size. The encoding complexity refers to the number of parameters and operations involved in the encoder network. A higher encoding complexity may lead to a more expressive and informative representation, but also a higher computational cost and a higher risk of overfitting. The beam size refers to the number of candidate sequences that are kept at each step of the decoding process. A larger beam size may increase the chance of finding the optimal sequence, but also increase the search space and the memory requirement.
One possible trade-off between encoding complexity and beam search performance is to use attention mechanisms, which are techniques that allow the decoder to dynamically focus on different parts of the encoded input based on the decoding context. Attention mechanisms can help reduce the encoding complexity by allowing the encoder to use a lower-dimensional representation, while also improving the beam search performance by providing more relevant information to the decoder at each step. Attention mechanisms can also help overcome the limitations of fixed-length representations, which may not be able to capture long or complex input sequences adequately.

## 5 Vanishing Gradient in RNN

### 5.1

RNN predict an output by using a sigmoid activation function. The issue occurs when we are taking the derivative and derivative of the sigmoid is always below 0.25 and hence when we multiply a lot of derivatives together according to the chain rule in backpropagation, we end up with a vanishing gradient such that we cant use them to effectively update the weights of the network.
The simplest solution is to replace the activation function of the network. Instead of sigmoid, use an activation function such as ReLU or its variants like LeakyReLU.
Initialize the weight matrix properly : Using initialization methods that produce larger weight values, such as Glorot/Xavier initialization, which helps to balance the scale of the activations and gradients throughout the network.
Using Skip Connections : where the output of an earlier layer is directly fed to a later layer, bypassing the intermediate layers. This helps to propagate gradients more easily across the network and can improve the flow of information through the model.

### 5.2

LSTM and GRU are two types of RNNs that have been designed to overcome some of the limitations of the basic RNNs, such as the vanishing or exploding gradient problem. LSTM and GRU solve this problem by

introducing gates that control the flow of information in and out of the hidden state, and allow the network to learn what to remember and what to forget. (memory cells: which are capable of retaining information for long periods of time without suffering from the vanishing gradient problem.) LSTM has three gates: input, output, and forget; while GRU has two gates: reset and update.
In the recurrency of the LSTM the activation function is the identity function with a derivative of 1.0. So, the backpropagated gradient neither vanishes or explodes when passing through, but remains constant. The effective weight of the recurrency is equal to the forget gate activation. So, if the forget gate is on (activation close to 1.0), then the gradient does not vanish. Since the forget gate activation is never upper than 1.0 , the gradient can't explode either.

## 5.3

we have:

$$h_m \ = \ \sigma(\theta h_{m-1} \ + \ x_m)$$

$$\frac{\partial \ h_m}{\partial \ h_{m-1}} \ = \ \theta\sigma^{'} \ (\theta h_{m-1} \ + \ x_m)$$

It can easily be shown that the derivative of the sigmoid function is:

$$\sigma^{'}(x) \ = \ \sigma \ (x) \ (1 \ - \ \sigma \ (x))$$

Since the sigmoid function is bounded and has outputs in (0, 1), one can conclude that maximum value for $\sigma^{'}(x)$ is 0.25. This can be seen from the graph or by taking the second derivative of this expression. Now let's see what happens when $k \to \infty$.

$$\frac{\partial \ h_{m+k}}{\partial \ h_m} \ = \ \prod_{i=m+k}^{m+1} \frac{\partial \ h_i}{\partial \ h_{i-1}}$$

$$= \ \prod_{i=1}^{k-1} \theta\sigma^{'} \ (\theta h_{i-1} \ + \ x_i)$$

$$\leqslant \ \prod_{i=1}^{k-1} \ 0.25\theta$$

$$= \ 0.25^{k-1} \ \theta^{k-1}$$

$$\to \ 0 \quad as \ \ k \to \infty \ (since \ |\theta| \ < 1)$$