University of Siena

Bioinformatics Project

Splice-junction Gene Sequences

Surya Sai Kadali

077240

DATA SET INFORMATION

Problem Description:
Splice junctions are points on a DNA sequence at which `superfluous' DNA is removed during the process of protein creation in higher organisms. The problem posed in this dataset is to recognize, given a sequence of DNA, the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out). This problem consists of two subtasks: recognizing exon/intron boundaries (referred to as EI sites), and recognizing intron/exon boundaries (IE sites). (In the biological community, IE borders are referred to a ``acceptors" while EI borders are referred to as ``donors".).

PREPROCESSING THE DATA

#importing dataset and libraries

We need to import the dataset with the help of pandas and we need to assign the dependent variables to ‘y’ and independent variables to ‘x’ by using the following code.

- **import numpy as np**
- **import pandas as pd**
- **data = pd.read_csv('E:\\datasets\\splice data.csv',names = ('class','instance name','sequence'))**
- **data = np.random.permutation(data)**
- **data = pd.DataFrame(data)**
- **x = data.iloc[0:,2]**
- **y = data.iloc[0:,0]**
- **x.head()**

```
0                  TTAACTGTGTCTCGTGCACGGAAAATCCAG...
1               AGAACTGGAGGAACCTGATCGCGGTGCTGGGTGGG...
2                  AGGCAACCAGCTAGGAGGAGGAGACTCGGACC...
3                   TGACAATAGCCCTGTGCTGCAGGAATGGGTA...
4                    GATTCTTACAGAAAACAAGTGGTTATAGA...
```

#converting y to int

Now by using the following code we need to convert the character dependent variables to integers which is a result of vector of numbers.

```
def state(u):
    for n,i in enumerate(u):
        if i == 'IE':
            u[n] = 0
        elif i == 'N':
            u[n] = 1
```

elif i == 'EI':

u[n] = 2

- **state(y)**

```
array ([2, 2, 1, ..., 1, 2, 2])
```

#converting y into matrix

As there is no catergorical priority within the outputs, we need to convert vector into a matrix of size (3190, 3) as there are 3 categories in the output.

- **from keras.utils import to_categorical**
- **y = to_categorical(y)**

```
array([[0., 0., 1.],
       [0., 0., 1.],
       [0., 1., 0.],
       ...,
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 0., 1.]],
```

#to remove spaces

We can observe extra spaces before the sequences which we can see above after executing **x.head().** So**,** inorder to remove the extra spaces from the starting and ending of the sequence we use the following code and the result will be displayed as,

- **x=x.str.strip()**

```
0    TTAACTGTGTCTCGTGCACGGAAAATCCAGCGTTTCTTGTCTCAGC...
1    AGAACTGGAGGAACCTGATCGCGGTGCTGGGTGGGTACCACTCTCC...
2    AGGCAACCAGCTAGGAGGAGGAGACTCGGACCCAGCTTGCAGCTGA...
3    TGACAATAGCCCTGTGCTGCAGGAATGGGTAACTGCCACTGACATC...
4    GATTCTTACAGAAAACAAGTGGTTATAGATGGTGAAACCTGTTTGT...
```

#to convert into kmers

We are converting the DNA sequence into overlapping Kmers of size 6 by using the following code.

- **def getKmers(x, size=6):**
  return [x[z:z+size].lower() for z in range(len(x) - size + 1)]
- **x=x.apply(getKmers)**

```
0       [ttaact, taactg, aactgt, actgtg, ctgtgt, tgtgt...
1       [agaact, gaactg, aactgg, actgga, ctggag, tggag...
2       [aggcaa, ggcaac, gcaacc, caacca, aaccag, accag...
3       [tgacaa, gacaat, acaata, caatag, aatagc, atagc...
4       [gattct, attctt, ttctta, tcttac, cttaca, ttaca...
                              ...
3185    [ttcaga, tcagaa, cagaag, agaagc, gaagct, aagct...
3186    [ctctct, tctctc, ctctct, tctctc, ctctct, tctct...
3187    [cggccc, ggccct, gccctg, ccctgg, cctggt, ctggt...
```

```
3188    [ggagtt, gagttg, agttgg, gttggc, ttggct, tggct...
3189    [gaatcg, aatcgt, atcgtc, tcgtca, cgtcat, gtcat...
```

#converting words into sentences

As we have obtained words of Kmers from the above code, we need to join the words into sequence.

- **for i in range(len(x)):**
  x[i] = ' '.join(x[i])
- **x[0]**

```
'ccaaca caacat aacata acatag catagt atagtg tagtga agtgaa gtgaaa tgaaac gaaacc
aaaccc aacccc accccc ccccca ccccaa cccaac ccaacc caacct aacctc acctct cctcta ctctac
tctact ctacta tactaa actaaa ctaaaa taaaaa aaaaat aaaata aaatac aataca atacaa tacaaa
acaaaa caaaaa aaaaat aaaatt aaatta aattag attagc ttagct tagctg agctgg gctggg ctgggg
tggggt ggggtg gggtgt ggtgtg gtgtgg tgtggt gtggtg tggtgg'
```

#splitting the data into training set and testing

Splitting the dataset into training set and test set by using train test split from sklearn library.

- **from sklearn.model_selection import train_test_split**
- **x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)**

#importing keras library from tensorflow and keras classes

- **import tensorflow as tf**
- **from tensorflow import keras**
- **from keras.preprocessing.text import Tokenizer**
- **from keras.models import Sequential**
- **from keras import layers**
- **from keras.layers import Dense,Dropout,Activation, Embedding, Flatten, Conv1D, MaxPooling1D, LSTM**

TOKENIZING AND PADDING THE DATA

 # tokenizing

Tokenizing means converting the words into integers, it is a part of Natural language processing.

- **tokens = Tokenizer()**
- **tokens.fit_on_texts(x_train)**
- **x_train = tokens.texts_to_sequences(x_train)**
- **x_test = tokens.texts_to_sequences(x_test)**
- **vocab_size = len(tokens.word_index) + 1**
- **print("Total words", vocab_size)**

 # padding

Padding converts vector into a matrix.

- **from keras.preprocessing.sequence import pad_sequences**
- **maxlen =55**
- **x_train = pad_sequences(x_train, padding='post', maxlen=maxlen)**
- **x_test = pad_sequences(x_test, padding='post', maxlen=maxlen)**
- **x_train.shape**
- **(2552, 55)**

CREATING THE CNN MODEL

We are creating a embedding layer which will be as an 3D input for the convolutional neural network and the maxpooling which extracts features from the input and then flatten the features into 1D which can fit into the dense layer and we use relu activation function in the input and hidden layers and the softmax in output layer as it consists of 3 outputs.

- **model = Sequential()**
- **model.add(Embedding(vocab_size,30,input_length=maxlen))**
- **model.add(Dropout(0.2))**
- **model.add(Conv1D(10, 5,activation='relu'))**
- **model.add(MaxPooling1D())**
- **model.add(Flatten())**
- **model.add(Dense(20,activation='relu'))**
- **model.add(Dropout(0.5))**
- **model.add(Dense(3,activation='softmax'))**
- **model.summary()**

COMPILING THE MODEL

We compile the model by using adam optimizer and categorical_crossentropy loss function.

- **model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy '])**

TRAINING THE MODEL

- **history = model.fit(x_train, y_train,validation_split=0.1,epochs=20,verbose=True, batch_size=10)**

```
2296/2296 [==============================] - 3s 1ms/step - loss: 0.9848 - acc: 0.54
36 - val_loss: 0.7427 - val_acc: 0.7461
Epoch 2/20
2296/2296 [==============================] - 1s 649us/step - loss: 0.4298 - acc: 0.
8454 - val_loss: 0.2320 - val_acc: 0.9180
Epoch 3/20
2296/2296 [==============================] - 1s 601us/step - loss: 0.1571 - acc: 0.
9486 - val_loss: 0.1764 - val_acc: 0.9453
Epoch 4/20
2296/2296 [==============================] - 2s 665us/step - loss: 0.0950 - acc: 0.
9665 - val_loss: 0.1590 - val_acc: 0.9453
Epoch 5/20
2296/2296 [==============================] - 2s 690us/step - loss: 0.0764 - acc: 0.
9695 - val_loss: 0.1911 - val_acc: 0.9414
Epoch 6/20
2296/2296 [==============================] - 2s 688us/step - loss: 0.0609 - acc: 0.
9743 - val_loss: 0.1918 - val_acc: 0.9258
Epoch 7/20
2296/2296 [==============================] - 2s 669us/step - loss: 0.0574 - acc: 0.
9804 - val_loss: 0.1639 - val_acc: 0.9492
Epoch 8/20
2296/2296 [==============================] - 1s 601us/step - loss: 0.0557 - acc: 0.
9774 - val_loss: 0.1807 - val_acc: 0.9414
Epoch 9/20
2296/2296 [==============================] - 1s 598us/step - loss: 0.0422 - acc: 0.
9848 - val_loss: 0.2111 - val_acc: 0.9414
Epoch 10/20
2296/2296 [==============================] - 1s 603us/step - loss: 0.0438 - acc: 0.
9865 - val_loss: 0.2218 - val_acc: 0.9453
Epoch 11/20
2296/2296 [==============================] - 1s 600us/step - loss: 0.0447 - acc: 0.
9821 - val_loss: 0.2485 - val_acc: 0.9297
Epoch 12/20
2296/2296 [==============================] - 1s 640us/step - loss: 0.0500 - acc: 0.
9782 - val_loss: 0.2545 - val_acc: 0.9375
Epoch 13/20
2296/2296 [==============================] - 1s 602us/step - loss: 0.0479 - acc: 0.
9839 - val_loss: 0.2643 - val_acc: 0.9414
Epoch 14/20
2296/2296 [==============================] - 1s 649us/step - loss: 0.0367 - acc: 0.
9882 - val_loss: 0.2849 - val_acc: 0.9297
Epoch 15/20
2296/2296 [==============================] - 1s 613us/step - loss: 0.0414 - acc: 0.
9878 - val_loss: 0.2671 - val_acc: 0.9219
Epoch 16/20
2296/2296 [==============================] - 1s 635us/step - loss: 0.0400 - acc: 0.
9869 - val_loss: 0.3030 - val_acc: 0.9375
Epoch 17/20
2296/2296 [==============================] - 2s 709us/step - loss: 0.0366 - acc: 0.
9861 - val_loss: 0.3059 - val_acc: 0.9258
Epoch 18/20
2296/2296 [==============================] - 2s 774us/step - loss: 0.0371 - acc: 0.
9887 - val_loss: 0.3125 - val_acc: 0.9375
Epoch 19/20
2296/2296 [==============================] - 2s 716us/step - loss: 0.0406 - acc: 0.
9826 - val_loss: 0.3136 - val_acc: 0.9414
Epoch 20/20
2296/2296 [==============================] - 2s 707us/step - loss: 0.0395 - acc: 0.
9848 - val_loss: 0.3278 - val_acc: 0.9336
```

We have trained the model for 20 epochs with a batch size of 10 and we have obtained an accuracy of `0.9848`

PREDICTION OF TEST SET

- **y_pred = model.predict(x_test).round()**
- **loss_and_metrics = model.evaluate(x_test, y_test, verbose=1)**
- **print(loss_and_metrics)**

```
638/638 [==============================] - 0s 124us/step
[0.20979163880667248, 0.942006268471386]
```

- **from sklearn.metrics import confusion_matrix**
- **cm=confusion_matrix(y_test.argmax(axis=1), y_pred.argmax(axis=1))**
- **cm**

```
array([[145,  14,   5],
       [  8, 316,   4],
       [  1,   5, 140]], dtype=int64)
```