bioeng-ml-python

December 13, 2019

1 1.

-

1.0.1

[1]:
```
a = 10
print(a)
type(a)
```

```
10
```

[1]:
```
int
```

-

1.0.2 , True, False, and, or

[2]:
```
print(True and False)
print(True or False)
print(not True and False)
```

```
False
True
False
```

-

1.0.3 , if, elif, else

[3]:
```
a = 10
if a < 0:
    print("Negative")
elif a >= 0 and a < 10:
    print("Less than 10")
else:
```

```
    print("Geater than 10")
    print("Or equal to 10")
```

```
Geater than 10
Or equal to 10
```

-

1.0.4 for, while

[4]:
```
for i in [0, 1, 2, 3]:
        print("for1", i)
        print("for2", i)

a = 4
i = 0
while i < a:
    print("while", i)
    if i == 2:
        print("stop")
        break
    i = i+1
```

```
for1 0
for2 0
for1 1
for2 1
for1 2
for2 2
for1 3
for2 3
while 0
while 1
while 2
stop
```

2 2.

2.1 2-1. List ( )

-

2.1.1

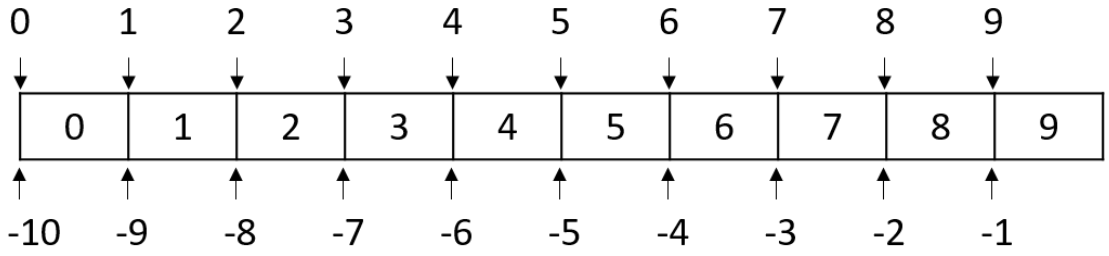
[5]:
```
expression = ["geneA", 1]
expression = [1, 2, 3]
expression = []
```

-

2.1.2 (1)

-

2.1.3 (1)



[6]:
```
geneids = [x for x in range(10)] #
print(geneids[0])
print(geneids[-1])
print(geneids[0:2])
print(geneids[:])
print(geneids[:-1])
print(geneids[1:])
print(geneids[:-10])
```

```
0
9
[0, 1]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[]
```

-

2.1.4

[12]:
```
geneids = [1, 2, 3]
print(geneids)
geneids.append(4)
print(geneids)
print("length: %d" % len(geneids))
geneids[len(geneids):] = [5]
print(geneids)
print(geneids.pop())
print(geneids)
```

```
[1, 2, 3]
[1, 2, 3, 4]
length: 4
[1, 2, 3, 4, 5]
5
[1, 2, 3, 4]
```

2.2 2-2. Tuple ( )

-

2.2.1 ‘(’, ’)’

-

2.2.2 ,

[45]:
```
geneids = (1, 2, 3)
print(geneids[0:2])
#geneids[0] = 4 ## error
```

```
(1, 2)
```

-

2.2.3

[1]:
```
geneids = ['123', '456', '789']
for geneid in geneids:
    print("geneid: %s" %geneid)
```

```
geneid: 123
geneid: 456
geneid: 789
```

2.3 2-3. Dictionary ( )

-

2.3.1 (key) (value) , '{' '}'

[16]:
```
gene_expr = {}
gene_expr['A'] = 0.5
print(gene_expr)
gene_expr['B'] = 1.2
print(gene_expr)
len(gene_expr)
```

```
{'A': 0.5}
{'A': 0.5, 'B': 1.2}
```

[16]:
```
2
```

-

2.3.2 '[', ']' , ,

[17]:
```
print(gene_expr['A'])
## gene_expr[0] # error
```

```
0.5
```

-

2.3.3 key value , del

[18]:
```
gene_expr['C'] = 0.3
print(gene_expr)
del gene_expr['C']
print(gene_expr)
```

```
{'A': 0.5, 'B': 1.2, 'C': 0.3}
{'A': 0.5, 'B': 1.2}
```

-

2.3.4 key value

[20]:
```
gene_expr_keys = list(gene_expr.keys())
print("keys:", gene_expr_keys)
gene_expr_values = list(gene_expr.values())
print("values:", gene_expr_values)
```

```
keys: ['A', 'B']
values: [0.5, 1.2]
```

-

2.3.5 in

[26]:
```
print('D' in gene_expr_keys)
print('D' in gene_expr)
print('A' in gene_expr)
```

```
False
False
True
```

-

2.3.6 items()

[2]:
```
gene_expr = {'A':0.5, 'B':1.2, 'C':0.3, 'D':3.2}
for geneid, expval in gene_expr.items():
    print("%s expression value is %s" %(geneid, expval))
```

```
A expression value is 0.5
B expression value is 1.2
C expression value is 0.3
D expression value is 3.2
```

3 3. , ,

3.1 3-1.

-

3.1.1

[27]:
```
def average(input):
    if len(input) == 0:
        return None
    return sum(input) / len(input)

x = [1,2,3,4,5,6,7,8,9,10]
average(x)
```

[27]:
```
5.5
```

3.2 3-2.

-

3.2.1 average mystat.py ,

[28]:
```
import mystat
x = list(range(10))
print(mystat.average(x))
```

```
4.5
```

-

3.2.2 test (name == main, True)

[29]:
```
%run mystat
```

```
average function is working well
```

-

3.2.3

[30]:
```
import os
os.getcwd()
```

[30]:
```
'/home/bioengml'
```

[32]:
```
from os import getcwd
getcwd()
```

[32]:
```
'/home/bioengml'
```

3.3 3-3.

-

3.3.1 Gene, Strain class

Gene attribute: name, chromosomal location, length
Strain attribute ( ): name, length of chromosome
Strain method ( ): compute average length of the genes

[76]:
```
import statistics
class ORF:
    def __init__(self, location, length, seq):
        self.location = location
        self.length = length
        self.sequence = seq

class Strain:
    def __init__(self, name, chrlength):
        self.name = name
        self.chr_length = chrlength
        self.orfs = []
    def add_orf(self, location, length, seq):
        self.orfs.append(ORF(location, length, seq))
    def gene_average(self):
        return statistics.mean([s.length for s in self.orfs])
```

[81]:
```
ecoli = Strain("ecoli", 5000000)
ecoli.add_orf(1, 1000, "ATG")
ecoli.add_orf(1001, 2000, "CCT")
ecoli.add_orf(3001, 3000, "ATC")
```

[82]:
```
print([g.location for g in ecoli.orfs])
print([g.sequence for g in ecoli.orfs])
```

```
[1, 1001, 3001]
['ATG', 'CCT', 'ATC']
```

-

3.3.2

[83]:
```
class Gene(ORF):
    def add_protein(self, prot_name, prot_seq):
        self.prot_name = prot_name
        self.prot_sequence = prot_seq
```

[86]:
```
gene1 = Gene(1, 1000, "ATG")
print(gene1.location)
gene1.add_protein("myprotein", "M")
print(gene1.prot_name)
```

```
1
myprotein
```

4 4.

[87]:
```
f = open("README.md", 'rt')
lines = f.readlines()
for line in lines:
    nline = line.split('\n')[0]
    print(nline)
```

```
#
12          [     ]          .
```

[88]:
```
f = open("write_test.txt", 'wt')
f.write('gene1;')
f.write('1;')
f.write('1000')
f.close()
```

[89]:
```
f = open("write_test.txt", 'rt')
lines = f.readlines()
for line in lines:
    nline = line.split(';')
    print(nline)
```

```
['gene1', '1', '1000']
```

5 5. Numpy ndarray

-

5.0.1

-

5.0.2

-

5.0.3 20

[90]:
```
import numpy as np
```

[94]:
```
arr = [1, 2, 3]
print(arr)
print(type(arr))

a = np.array([1,2,3])
print(a)
print(a.dtype)
print(a.shape)
print(type(a))
```

```
[1, 2, 3]
<class 'list'>
[1 2 3]
int64
(3,)
<class 'numpy.ndarray'>
```

-

5.0.4 numpy

- int(8, 16, 32, 64)
- uint(8 ,16, 32, 54)
- float(16, 32, 64, 128)
- complex(64, 128, 256)
- bool
- string_
- object
- unicode_
-

5.0.5 np.zeros(), np.ones(), np.arange()

-

5.0.6

[5]:
```
a = np.arange(1, 10).reshape(3,3)
print(a)
a = np.ones((3,4), dtype=np.int16)
b = np.ones((3,4), dtype=np.int16)
print(a)
print(b)
print(a+b)
print(a-b)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[1 1 1 1]
 [1 1 1 1]
 [1 1 1 1]]
[[1 1 1 1]
 [1 1 1 1]
 [1 1 1 1]]
[[2 2 2 2]
 [2 2 2 2]
 [2 2 2 2]]
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]
```

-

5.0.7 numpy

- np.sqrt()
- np.log()
- np.square()
- np.log()
- np.ceil()
- np.floor()
- np.isnan()
- np.sum()

- np.mean()
- np.std()
- np.min()

6 6. Pandas Series, DataFrame

-

6.0.1 Pandas Series 1 , DataFrame 2

-

6.0.2

-

6.0.3 , ,

[95]:
```
from pandas import Series, DataFrame
```

[96]:
```
genes = Series([0.1, 0.2, 1.4, 0.6, 1.1])
print(genes)
```

```
0    0.1
1    0.2
2    1.4
3    0.6
4    1.1
dtype: float64
```

[97]:
```
genes = Series([0.1, 0.2, 1.4, 0.6, 1.1], index=['A', 'B', 'C', 'D', 'E'])
print(genes)
```

```
A    0.1
B    0.2
C    1.4
D    0.6
E    1.1
dtype: float64
```

-

6.0.4 ,

[99]:
```
genes1 = Series([0.1, 0.2, 1.4, 0.6, 1.1], index=['A', 'B', 'C', 'D', 'E'])
genes2 = Series([0.1, 0.2, 1.4, 0.6, 1.1], index=['B', 'C', 'D', 'E', 'A'])
genes1 + genes2
```

[99]:
```
A    1.2
B    0.3
C    1.6
D    2.0
E    1.7
dtype: float64
```

[100]:
```
print(genes2.sort_values())
print(genes2.sort_index())
```

```
B    0.1
C    0.2
E    0.6
A    1.1
D    1.4
dtype: float64
A    1.1
B    0.1
C    0.2
D    1.4
E    0.6
dtype: float64
```

-

6.0.5 DataFrame ‘{’, ‘}’

-

6.0.6 DataFrame Series

[104]:
```
genes = {'A': [0.5, 0.1, 0.3],
         'B': [0.8, 0.9, 0.4]}
print(genes)
genes_df = DataFrame(genes)
print(genes_df)
print(genes_df['A'])
print(type(genes_df['A']))
```

```
{'A': [0.5, 0.1, 0.3], 'B': [0.8, 0.9, 0.4]}
     A    B
```

```
0  0.5  0.8
1  0.1  0.9
2  0.3  0.4
0    0.5
1    0.1
2    0.3
Name: A, dtype: float64
<class 'pandas.core.series.Series'>
```

[105]:
```
genes = {'A': [0.5, 0.1, 0.3],
         'B': [0.8, 0.9, 0.4]}
genes_df = DataFrame(genes, columns=['B', 'A'], index=['day1', 'day2', 'day3'])
print(genes_df)
```

```
        B    A
day1  0.8  0.5
day2  0.9  0.1
day3  0.4  0.3
```

[110]:
```
print(genes_df['A'])
print(genes_df.loc['day1'])
print(genes_df.index)
print(list(genes_df.columns))
```

```
day1    0.5
day2    0.1
day3    0.3
Name: A, dtype: float64
B    0.8
A    0.5
Name: day1, dtype: float64
Index(['day1', 'day2', 'day3'], dtype='object')
['B', 'A']
```

7 7. numpy (Tensor)

-

7.0.1 ( )

-

7.0.2 tensorflow - ,

-

7.0.3 theano - python

-

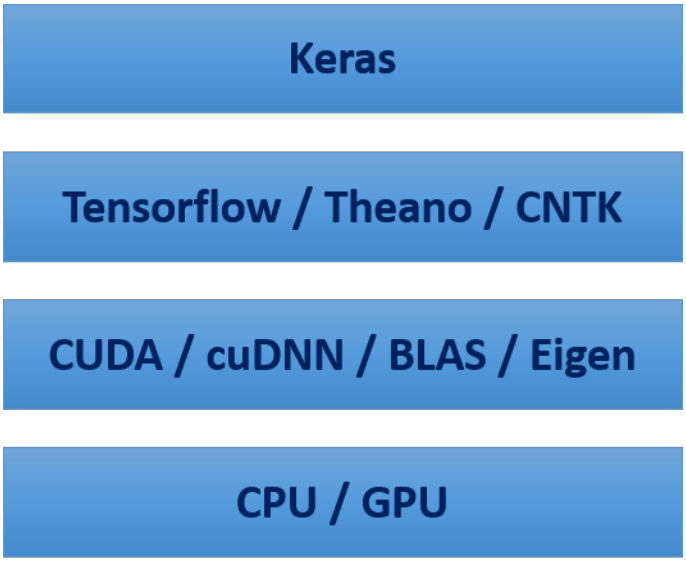
7.0.4 PyTorch - ,

-

7.0.5 CNTK (Cognitive Toolkit) - Microsoft ,

-

7.0.6 Keras tensorflow, theano, CNTK



-

7.0.7 (float32, uint8, float64)

-

7.0.8

-

7.0.9 0D , 1D , 2D , ...

-

7.0.10 (ndim), (shape), (dtype)

[114]:
```
import numpy as np
x = np.array(12)
print(x)
print(x.ndim)
```

```
12
0
```

[116]:
```
x = np.array([1, 2, 3, 4, 5])
print(x.ndim)
```

```
1
```

[119]:
```
x = np.array([[1,2,3,4,5],
              [6,7,8,9,10],
              [11,12,13,14,15]])
print(x.ndim)
```

```
2
```

[146]:
```
x = np.array([[[1,2,3],
               [2,3,4]],
              [[5,6,7],
               [8,9,10]],
              [[11,12,13],
               [14,15,16]]])
```

[145]:
```
x.shape
```

[145]: (3, 2, 3)

[159]:
```
from keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

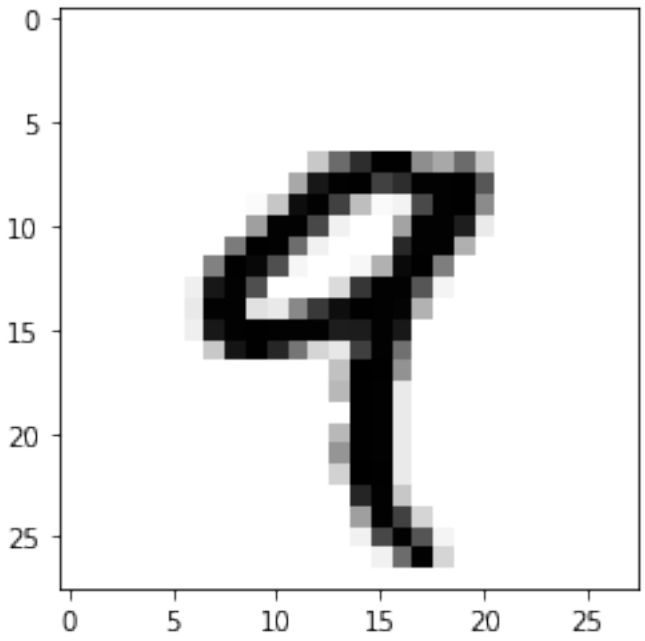
[149]:
```
print(train_images.shape)
print(train_labels.shape)
print(train_images.dtype)
```

```
(60000, 28, 28)
(60000,)
uint8
```

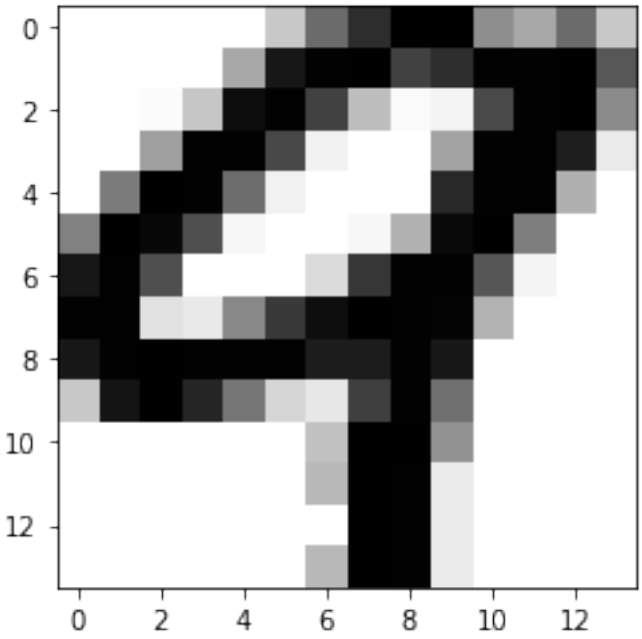
[155]:
```
digit = train_images[4]
print(digit.shape)
```

```
import matplotlib.pyplot as plt
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
```

```
(28, 28)
```



[156]:
```
my_slice = train_images[4,7:-7,7:-7]
plt.imshow(my_slice, cmap=plt.cm.binary)
plt.show()
```



-

7.0.11 (batch) -

-

7.0.12 (epoch) -

[158]:
```
batch1 = train_images[:128]
batch2 = train_images[129:256]
#...
```

8 8. Neural Network

-

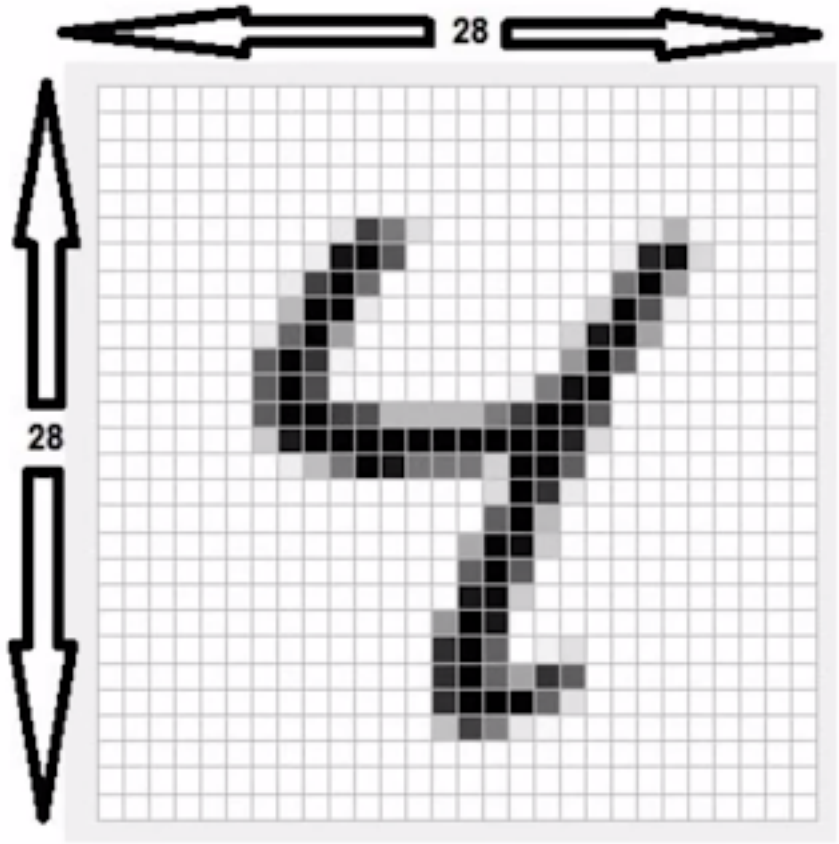
8.0.1 MNIST

-

8.0.2 (28x28) 10 (0 9)

-

8.0.3 (NIST) 60000 1



[80]:
```
from keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

[81]:
```
print(train_images.shape)
print(train_labels.shape)
print(test_images.shape)
print(test_labels.shape)
```

```
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
```

-

8.0.4

[82]:
```
train_images = train_images.reshape((60000, 28*28))
train_images = train_images.astype('float32')/255
test_images = test_images.reshape((10000, 28*28))
test_images = test_images.astype('float32')/255
```

[83]:
```
print(train_images.shape)
print(train_labels.shape)
print(test_images.shape)
print(test_labels.shape)
```

```
(60000, 784)
(60000,)
(10000, 784)
(10000,)
```

[84]:
```
print(test_labels[:10])
```

```
[7 2 1 0 4 1 4 9 5 9]
```

-

8.0.5

[85]:
```
from keras import models
from keras import layers
from keras.utils import to_categorical

network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28*28,)))
network.add(layers.Dense(10, activation='softmax'))

network.compile(optimizer='rmsprop', loss='categorical_crossentropy',␣
↪metrics=['accuracy'])
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
print(test_labels[:10])
```

```
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```

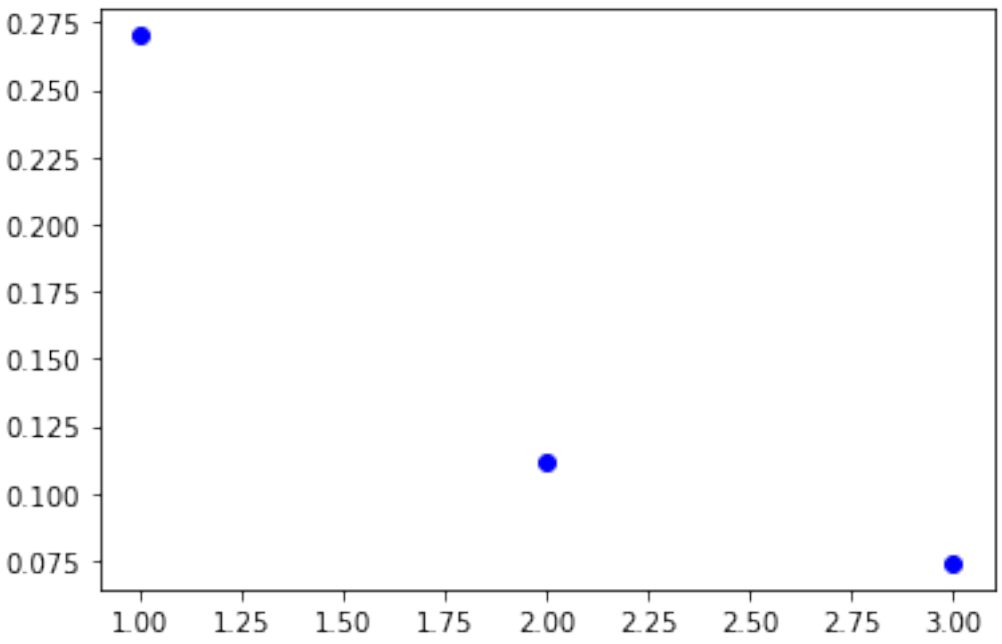
```
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

[86]:
```
history = network.fit(train_images, train_labels, epochs=3, batch_size=128)
```

```
Epoch 1/3
60000/60000 [==============================] - 6s 101us/step - loss: 0.2701 -
acc: 0.9210
Epoch 2/3
60000/60000 [==============================] - 5s 88us/step - loss: 0.1121 -
acc: 0.9664
Epoch 3/3
60000/60000 [==============================] - 5s 90us/step - loss: 0.0742 -
acc: 0.9781
```

[87]:
```
import matplotlib.pyplot as plt

history_dict = history.history
print(history_dict.keys())
loss =history_dict['loss']
acc = history_dict['acc']
epochs = range(1, len(loss)+1)
plt.plot(epochs, loss, 'bo')
plt.show()
```

```
dict_keys(['loss', 'acc'])
```



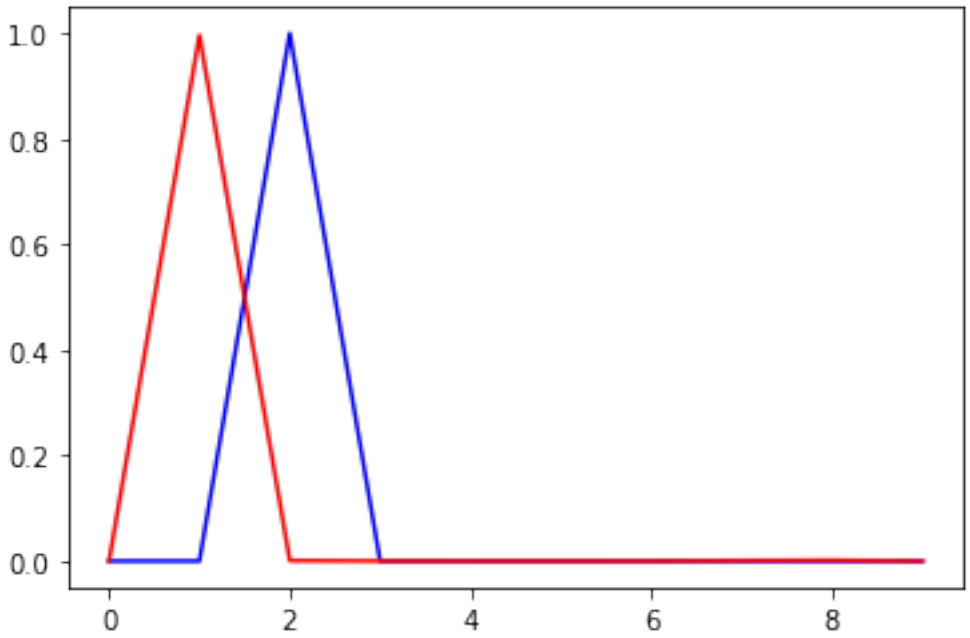
-

8.0.6

[88]:
```
results = network.predict(test_images)
print(results.shape)
```

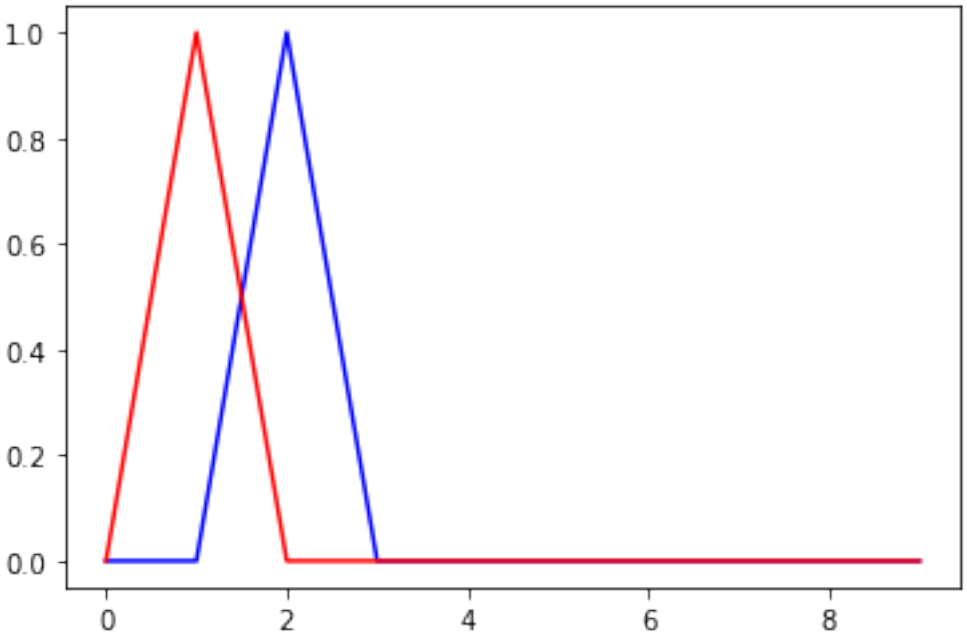
```
(10000, 10)
```

[117]:
```
import matplotlib.pyplot as plt
plt.plot(results[1,], "b")
plt.plot(results[2,], "r")
plt.show()
[round(x) for x in list(results[1,])]
```



[117]:
```
[0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

[119]:
```
plt.plot(test_labels[1,], "b")
plt.plot(test_labels[2,], "r")
plt.show()
print(test_labels[1,])
```




```
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
```

[22]:
```
results = network.evaluate(test_images, test_labels)
print(results)
```

```
10000/10000 [==============================] - 1s 64us/step
[0.07228709341179411, 0.9839]
```

9 9. Biopython - Sequence objects

[3]:
```
from Bio.Seq import Seq
from Bio.Alphabet import IUPAC
my_seq = Seq("AGTACACTGGT", IUPAC.unambiguous_dna)
my_seq
```

[3]:
```
Seq('AGTACACTGGT', IUPACUnambiguousDNA())
```

[4]:
```
for index, letter in enumerate(my_seq):
    print("%i %s " % (index, letter))
```

```
0 A
1 G
2 T
3 A
4 C
```

```
5 A
6 C
7 T
8 G
9 G
10 T
```

[5]:
```
x = [1, 4, 5, 7, 8]
for i in enumerate(x):
    print(i)
```

```
(0, 1)
(1, 4)
(2, 5)
(3, 7)
(4, 8)
```

[9]:
```
print(my_seq)
print(my_seq[0:3])
print(my_seq[0::2])
print(str(my_seq))
print(my_seq + "ATG")
print(my_seq=="ATG")
print("AGT" in my_seq)
my_seq_low = my_seq.lower()
print(my_seq_low)
print(my_seq_low.upper())
print(my_seq.complement())
print(my_seq.reverse_complement())
```

```
AGTACACTGGT
AGT
ATCCGT
AGTACACTGGT
AGTACACTGGTATG
False
True
agtacactggt
AGTACACTGGT
TCATGTGACCA
ACCAGTGTACT
```

[10]:
```
mrna = my_seq.transcribe()
print(mrna)
prot = mrna.translate() ## truncated
print(prot)
print(my_seq.translate())
```

```
AGUACACUGGU
STL
STL
```

[11]:
```
from Bio.Data import CodonTable
standard_table = CodonTable.unambiguous_dna_by_id[1]
print(standard_table)
print(standard_table.start_codons)
print(standard_table.stop_codons)
```

```
Table 1 Standard, SGC0

  |  T      |  C      |  A      |  G      |
--+---------+---------+---------+---------+--
T | TTT F   | TCT S   | TAT Y   | TGT C   | T
T | TTC F   | TCC S   | TAC Y   | TGC C   | C
T | TTA L   | TCA S   | TAA Stop| TGA Stop| A
T | TTG L(s)| TCG S   | TAG Stop| TGG W   | G
--+---------+---------+---------+---------+--
C | CTT L   | CCT P   | CAT H   | CGT R   | T
C | CTC L   | CCC P   | CAC H   | CGC R   | C
C | CTA L   | CCA P   | CAA Q   | CGA R   | A
C | CTG L(s)| CCG P   | CAG Q   | CGG R   | G
--+---------+---------+---------+---------+--
A | ATT I   | ACT T   | AAT N   | AGT S   | T
A | ATC I   | ACC T   | AAC N   | AGC S   | C
A | ATA I   | ACA T   | AAA K   | AGA R   | A
A | ATG M(s)| ACG T   | AAG K   | AGG R   | G
--+---------+---------+---------+---------+--
G | GTT V   | GCT A   | GAT D   | GGT G   | T
G | GTC V   | GCC A   | GAC D   | GGC G   | C
G | GTA V   | GCA A   | GAA E   | GGA G   | A
G | GTG V   | GCG A   | GAG E   | GGG G   | G
--+---------+---------+---------+---------+--
['TTG', 'CTG', 'ATG']
['TAA', 'TAG', 'TGA']
```

10 10. biopython - SeqRecord

-

10.0.1 Sequence annotation objects

-

10.0.2 identifier feature

[12]:
```
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord

simple_seq = Seq("GATC")
simple_seq_r = SeqRecord(simple_seq)
```

[13]:
```
simple_seq_r.id = "AC12345"
simple_seq_r.description = "Made up sequence I wish I could write a paper about"
print(simple_seq_r.description)
print(simple_seq_r.seq)
```

```
Made up sequence I wish I could write a paper about
GATC
```

[ ]:
```
help(SeqRecord)
```

-

10.0.3 Read fasta file

-

10.0.4 *Yersinia pestis biovar Microtus* str. 91001 plasmid ( )

[21]:
```
from Bio import SeqIO
record = SeqIO.read("datasets/NC_005816.fna", "fasta")
```

[23]:
```
record
```

[23]:
```
SeqRecord(seq=Seq('TGTAACGAACGGTGCAATAGTGATCCACACCCAACGCCTGAAATCAGATCCAGG…CTG'
, SingleLetterAlphabet()), id='gi|45478711|ref|NC_005816.1|',
name='gi|45478711|ref|NC_005816.1|', description='gi|45478711|ref|NC_005816.1|
Yersinia pestis biovar Microtus str. 91001 plasmid pPCP1, complete sequence',
dbxrefs=[])
```

[24]:
```
print(record.id)
print(record.name)
print(record.description)
```

```
gi|45478711|ref|NC_005816.1|
gi|45478711|ref|NC_005816.1|
gi|45478711|ref|NC_005816.1| Yersinia pestis biovar Microtus str. 91001 plasmid
pPCP1, complete sequence
```

-

10.0.5 Read GenBank file

[25]:
```
from Bio import SeqIO
record = SeqIO.read("datasets/NC_005816.gb", "genbank")
```

[26]:
```
print(record.id)
print(record.name)
print(record.description)
print(len(record.features))
print(record.features[0])
print(record.features[2])
print(record.features[3])
```

```
NC_005816.1
NC_005816
Yersinia pestis biovar Microtus str. 91001 plasmid pPCP1, complete sequence
41
type: source
location: [0:9609](+)
qualifiers:
    Key: biovar, Value: ['Microtus']
    Key: db_xref, Value: ['taxon:229193']
    Key: mol_type, Value: ['genomic DNA']
    Key: organism, Value: ['Yersinia pestis biovar Microtus str. 91001']
    Key: plasmid, Value: ['pPCP1']
    Key: strain, Value: ['91001']

type: gene
location: [86:1109](+)
qualifiers:
    Key: db_xref, Value: ['GeneID:2767718']
    Key: locus_tag, Value: ['YP_pPCP01']

type: CDS
location: [86:1109](+)
qualifiers:
    Key: codon_start, Value: ['1']
    Key: db_xref, Value: ['GI:45478712', 'GeneID:2767718']
    Key: locus_tag, Value: ['YP_pPCP01']
    Key: note, Value: ['similar to corresponding CDS from previously sequenced
pPCP plasmid of Yersinia pestis KIM (AF053945) and CO92 (AL109969), also many
transposase entries for insertion sequence IS100 of Yersinia pestis. Contains
IS21-like element transposase, HTH domain (Interpro|IPR007101)']
    Key: product, Value: ['putative transposase']
    Key: protein_id, Value: ['NP_995567.1']
    Key: transl_table, Value: ['11']
    Key: translation, Value: ['MVTFETVMEIKILHKQGMSSRAIARELGISRNTVKRYLQAKSEPPKYTP
```

```
RPAVASLLDEYRDYIRQRIADAHPYKIPATVIAREIRDQGYRGGMTILRAFIRSLSVPQEQEPAVRFETEPGRQMQVDWG
TMRNGRSPLHVFVAVLGYSRMLYIEFTDNMRYDTLETCHRNAFRFFGGVPREVLYDNMKTVVLQRDAYQTGQHRFHPSLW
QFGKEMGFSPRLCRPFRAQTKGKVERMVQYTRNSFYIPLMTRLRPMGITVDVETANRHGLRWLHDVANQRKHETIQARPC
DRWLEEQQSMLALPPEKKEYDVHLDENLVNFDKHPLHHPLSIYDSFCRGVA']
```

-

10.0.6 in

[117]:
```
for feature in record.features:
    if 4350 in feature:
        print(feature)
        print("%s %s" % (feature.type, feature.qualifiers.get("db_xref")))
```

```
type: source
location: [0:9609](+)
qualifiers:
    Key: biovar, Value: ['Microtus']
    Key: db_xref, Value: ['taxon:229193']
    Key: mol_type, Value: ['genomic DNA']
    Key: organism, Value: ['Yersinia pestis biovar Microtus str. 91001']
    Key: plasmid, Value: ['pPCP1']
    Key: strain, Value: ['91001']

source ['taxon:229193']
type: gene
location: [4342:4780](+)
qualifiers:
    Key: db_xref, Value: ['GeneID:2767712']
    Key: gene, Value: ['pim']
    Key: locus_tag, Value: ['YP_pPCP05']

gene ['GeneID:2767712']
type: CDS
location: [4342:4780](+)
qualifiers:
    Key: codon_start, Value: ['1']
    Key: db_xref, Value: ['GI:45478716', 'GeneID:2767712']
    Key: gene, Value: ['pim']
    Key: locus_tag, Value: ['YP_pPCP05']
    Key: note, Value: ['similar to many previously sequenced pesticin immunity
protein entries of Yersinia pestis plasmid pPCP, e.g. gi|
16082683|,ref|NP_395230.1| (NC_003132) , gi|1200166|emb|CAA90861.1| (Z54145 ) ,
gi|1488655| emb|CAA63439.1| (X92856) , gi|2996219|gb|AAC62543.1| (AF053945) ,
and gi|5763814|emb|CAB531 67.1| (AL109969)']
    Key: product, Value: ['pesticin immunity protein']
```

```
    Key: protein_id, Value: ['NP_995571.1']
    Key: transl_table, Value: ['11']
    Key: translation, Value: ['MGGGMISKLFCLALIFLSSSGLAEKNTYTAKDILQNLELNTFGNSLSHG
IYGKQTTFKQTEFTNIKSNTKKHIALINKDNSWMISLKILGIKRDEYTVCFEDFSLIRPPTYVAIHPLLIKKVKSGNFIV
VKEIKKSIPGCTVYYH']

CDS ['GI:45478716', 'GeneID:2767712']
```

-

10.0.7 format

[ ]:
```
record.format("fasta")
```

-

10.0.8 Slicing

[28]:
```
print(len(record))
print(len(record.features))
```

```
9609
41
```

[29]:
```
print(record.features[20])
print(record.features[21])
```

```
type: gene
location: [4342:4780](+)
qualifiers:
    Key: db_xref, Value: ['GeneID:2767712']
    Key: gene, Value: ['pim']
    Key: locus_tag, Value: ['YP_pPCP05']

type: CDS
location: [4342:4780](+)
qualifiers:
    Key: codon_start, Value: ['1']
    Key: db_xref, Value: ['GI:45478716', 'GeneID:2767712']
    Key: gene, Value: ['pim']
    Key: locus_tag, Value: ['YP_pPCP05']
    Key: note, Value: ['similar to many previously sequenced pesticin immunity
protein entries of Yersinia pestis plasmid pPCP, e.g. gi|
16082683|,ref|NP_395230.1| (NC_003132) , gi|1200166|emb|CAA90861.1| (Z54145 ) ,
gi|1488655| emb|CAA63439.1| (X92856) , gi|2996219|gb|AAC62543.1| (AF053945) ,
and gi|5763814|emb|CAB531 67.1| (AL109969)']
    Key: product, Value: ['pesticin immunity protein']
```

```
    Key: protein_id, Value: ['NP_995571.1']
    Key: transl_table, Value: ['11']
    Key: translation, Value: ['MGGGMISKLFCLALIFLSSSGLAEKNTYTAKDILQNLELNTFGNSLSHG
IYGKQTTFKQTEFTNIKSNTKKHIALINKDNSWMISLKILGIKRDEYTVCFEDFSLIRPPTYVAIHPLLIKKVKSGNFIV
VKEIKKSIPGCTVYYH']
```

[30]:
```
sub_record = record[4300:4800]
print(sub_record)
print(len(sub_record))
print(len(sub_record.features))
print(sub_record.features[0])
print(sub_record.features[1])
```

```
ID: NC_005816.1
Name: NC_005816
Description: Yersinia pestis biovar Microtus str. 91001 plasmid pPCP1, complete
sequence
Number of features: 2
Seq('ATAAATAGATTATTCCAAATAATTTATTTATGTAAGAACAGGATGGGAGGGGGA…TTA',
IUPACAmbiguousDNA())
500
2
type: gene
location: [42:480](+)
qualifiers:
    Key: db_xref, Value: ['GeneID:2767712']
    Key: gene, Value: ['pim']
    Key: locus_tag, Value: ['YP_pPCP05']

type: CDS
location: [42:480](+)
qualifiers:
    Key: codon_start, Value: ['1']
    Key: db_xref, Value: ['GI:45478716', 'GeneID:2767712']
    Key: gene, Value: ['pim']
    Key: locus_tag, Value: ['YP_pPCP05']
    Key: note, Value: ['similar to many previously sequenced pesticin immunity
protein entries of Yersinia pestis plasmid pPCP, e.g. gi|
16082683|,ref|NP_395230.1| (NC_003132) , gi|1200166|emb|CAA90861.1| (Z54145 ) ,
gi|1488655| emb|CAA63439.1| (X92856) , gi|2996219|gb|AAC62543.1| (AF053945) ,
and gi|5763814|emb|CAB531 67.1| (AL109969)']
    Key: product, Value: ['pesticin immunity protein']
    Key: protein_id, Value: ['NP_995571.1']
    Key: transl_table, Value: ['11']
    Key: translation, Value: ['MGGGMISKLFCLALIFLSSSGLAEKNTYTAKDILQNLELNTFGNSLSHG
IYGKQTTFKQTEFTNIKSNTKKHIALINKDNSWMISLKILGIKRDEYTVCFEDFSLIRPPTYVAIHPLLIKKVKSGNFIV
VKEIKKSIPGCTVYYH']
```

11 11. Biopython - Parsing Genbank records from the NCBI

-

11.0.1 / with

[31]:
```
from Bio import Entrez
from Bio import SeqIO
Entrez.email = "haseong@kribb.re.kr"
with Entrez.efetch(db="nucleotide", rettype="gb", retmode="text", id="6273291")␣
↪as handle:
    seq_record = SeqIO.read(handle, "gb")
print("%s with %i features" % (seq_record.id, len(seq_record.features)))
```

```
AF191665.1 with 3 features
```

-

11.0.2 record parse

[32]:
```
with Entrez.efetch(db="nucleotide", rettype="gb", retmode="text",␣
↪id="6273291,6273290,6273289") as handle:
    for seq_record in SeqIO.parse(handle, "gb"):
        print("%s %s..." % (seq_record.id, seq_record.description[:50]))
        print("Sequence length %i, %i features, from: %s"
              % (len(seq_record), len(seq_record.features), seq_record.
↪annotations["source"]))
```

```
AF191665.1 Opuntia marenae rpl16 gene; chloroplast gene for c…
Sequence length 902, 3 features, from: chloroplast Grusonia marenae
AF191664.1 Opuntia clavata rpl16 gene; chloroplast gene for c…
Sequence length 899, 3 features, from: chloroplast Grusonia clavata
AF191663.1 Opuntia bradtiana rpl16 gene; chloroplast gene for…
Sequence length 899, 3 features, from: chloroplast Grusonia bradtiana
```

-

[34]:
```
from Bio import SeqIO
SeqIO.write(seq_record, "datasets/my_seq_records.fa", "fasta")
```

[34]: 1

12 12. Biopython - multiple sequence alignment objects

- ### 12.0.1 http://biopython.org/DIST/docs/tutorial/Tutorial.html#htoc70

- ### 12.0.2 Bio.AlignIO.read() returns a single MultipleSeqAlignment object

- ### 12.0.3 Bio.AlignIO.parse() returns MultipleSeqAlignment objects

- ### 12.0.4 Alignment tools

[35]:
```
import Bio.Align.Applications as alnapps
dir(alnapps)
```

[35]:
```
['ClustalOmegaCommandline',
 'ClustalwCommandline',
 'DialignCommandline',
 'MSAProbsCommandline',
 'MafftCommandline',
 'MuscleCommandline',
 'PrankCommandline',
 'ProbconsCommandline',
 'TCoffeeCommandline',
 '_ClustalOmega',
 '_Clustalw',
 '_Dialign',
 '_MSAProbs',
 '_Mafft',
 '_Muscle',
 '_Prank',
 '_Probcons',
 '_TCoffee',
 '__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
```

```
'__loader__',
'__name__',
'__package__',
'__path__',
'__spec__']
```

•

12.0.5 Clustalw

•

12.0.6 Seven prickly-pear DNA sequences (from the cactus family Opuntia( ))

[41]:
```
from Bio.Align.Applications import ClustalwCommandline as clw
#help(clw)
cline = clw("clustalw2", infile="datasets/opuntia.fasta")
stdout, stderr = cline()
print(cline)
print(stdout)
```

```
clustalw2 -infile=datasets/opuntia.fasta



 CLUSTAL 2.1 Multiple Sequence Alignments


Sequence format is Pearson
Sequence 1: gi|6273291|gb|AF191665.1|AF191665   902 bp
Sequence 2: gi|6273290|gb|AF191664.1|AF191664   899 bp
Sequence 3: gi|6273289|gb|AF191663.1|AF191663   899 bp
Sequence 4: gi|6273287|gb|AF191661.1|AF191661   895 bp
Sequence 5: gi|6273286|gb|AF191660.1|AF191660   893 bp
Sequence 6: gi|6273285|gb|AF191659.1|AF191659   894 bp
Sequence 7: gi|6273284|gb|AF191658.1|AF191658   896 bp
Start of Pairwise alignments
Aligning…

Sequences (1:2) Aligned. Score:  99
Sequences (1:3) Aligned. Score:  99
Sequences (1:4) Aligned. Score:  98
Sequences (1:5) Aligned. Score:  98
Sequences (1:6) Aligned. Score:  98
Sequences (1:7) Aligned. Score:  98
Sequences (2:3) Aligned. Score:  99
```

```
Sequences (2:4) Aligned. Score:  98
Sequences (2:5) Aligned. Score:  98
Sequences (2:6) Aligned. Score:  98
Sequences (2:7) Aligned. Score:  98
Sequences (3:4) Aligned. Score:  98
Sequences (3:5) Aligned. Score:  98
Sequences (3:6) Aligned. Score:  98
Sequences (3:7) Aligned. Score:  98
Sequences (4:5) Aligned. Score:  99
Sequences (4:6) Aligned. Score:  99
Sequences (4:7) Aligned. Score:  99
Sequences (5:6) Aligned. Score:  99
Sequences (5:7) Aligned. Score:  99
Sequences (6:7) Aligned. Score:  99
Guide tree file created:   [datasets/opuntia.dnd]

There are 6 groups
Start of Multiple Alignment

Aligning…
Group 1: Sequences:   2      Score:16933
Group 2: Sequences:   2      Score:16703
Group 3: Sequences:   4      Score:16812
Group 4: Sequences:   2      Score:17071
Group 5: Sequences:   3      Score:16845
Group 6: Sequences:   7      Score:16678
Alignment Score 114256

CLUSTAL-Alignment file created  [datasets/opuntia.aln]
```

[43]:
```
#print(stdout)
from Bio import AlignIO
align = AlignIO.read("datasets/opuntia.aln", "clustal")
print(align)
```

```
SingleLetterAlphabet() alignment with 7 rows and 906 columns
TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAG…AGA
gi|6273285|gb|AF191659.1|AF191
TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAG…AGA
gi|6273284|gb|AF191658.1|AF191
TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAG…AGA
gi|6273287|gb|AF191661.1|AF191
TATACATAAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAG…AGA
gi|6273286|gb|AF191660.1|AF191
TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAG…AGA
gi|6273290|gb|AF191664.1|AF191
```

```
TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAG…AGA
gi|6273289|gb|AF191663.1|AF191
TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAG…AGA
gi|6273291|gb|AF191665.1|AF191
```

[44]:
```
from Bio import Phylo
tree = Phylo.read("opuntia.dnd", "newick")
Phylo.draw_ascii(tree)
```

```
                             _____________ gi|6273291|gb|AF191665.1|AF191665
 ____________________________|
|                            |   ______ gi|6273290|gb|AF191664.1|AF191664
|                            |__|
|                               |_____ gi|6273289|gb|AF191663.1|AF191663
|
_|_________________ gi|6273287|gb|AF191661.1|AF191661
 |
 |__________ gi|6273286|gb|AF191660.1|AF191660
 |
 |    __ gi|6273285|gb|AF191659.1|AF191659
 |___|
     | gi|6273284|gb|AF191658.1|AF191658
```

13 13. Biopython - PSSM matrix

[3]:
```
from Bio.Seq import Seq
test_seq=Seq("TAAGCGTGCACGCGCAACACGTGCATTA")
test_seq
print(test_seq)
```

```
TAAGCGTGCACGCGCAACACGTGCATTA
```

[4]:
```
from Bio import AlignIO
from Bio.Align import AlignInfo
```

-

13.0.1 Pfam database, align

-

13.0.2 Family: Sigma54_activ_2 (PF14532) https://pfam.xfam.org/family/PF14532#tabvie

[ ]:
```
align = AlignIO.read("datasets/PF14532_full.txt", "stockholm")
print(align)
len(align)
type(align)
print(align[0])
print(align[0].seq)
print(align[0].format("clustal"))
```

-

13.0.3 slicing alignment

[6]:
```
# print(align[3:8].format("clustal"))
print(align[3:8,:200].format("clustal"))
print(align[3:8,197])
```

```
CLUSTAL X (1.81) multiple sequence alignment


V7EPJ0_9RHOB/141-283
--------------------------------------------------
B1ZTM1_OPITP/145-296
--------------------------------------------------
W3ANH6_9FIRM/219-355
--------------------------------------------------
Q6LNI3_PHOPR/144-289
--------------------------------------------------
A0A1G8U4Y5_9RHOB/145-284
--------------------------------------------------

V7EPJ0_9RHOB/141-283              ----------------------------VGRTPA-M-Q-A-L-
Y-R-L-
B1ZTM1_OPITP/145-296              ----------------------------IGQSAS-M-R-K-L-
V-Q-Q-
W3ANH6_9FIRM/219-355              ----------------------------y--KSRK-M-Q-K-T-
V-D-L-
Q6LNI3_PHOPR/144-289              ----------------------------IGDSPL-S-V-K-L-
R-E-Q-
A0A1G8U4Y5_9RHOB/145-284          ----------------------------r-GTSPQ-S-E-E-L-
R-A-R-

V7EPJ0_9RHOB/141-283              --V---A---R---V---M--------N---T----D-------
L-----
B1ZTM1_OPITP/145-296              --V---K---K---L---A--------A---V----R-------
```

```
T-----
W3ANH6_9FIRM/219-355                --A---E---K---L---S--------R---T----D-------
C-----
Q6LNI3_PHOPR/144-289                --I---A---N---I---A--------L---T----N-------
K-----
A0A1G8U4Y5_9RHOB/145-284            --V---R---L---V---A--------R---A----G-------
A-----

V7EPJ0_9RHOB/141-283                ---A-----V---L-V-T--GES-GT-GK----S----L-I-A
----K--
B1ZTM1_OPITP/145-296                ---P-----V---L-L-I--GEN-GS-GK----S----A-V-A
----E--
W3ANH6_9FIRM/219-355                ---P-----K---L-I-V--EPV-GN-LH----R----A-F-I
----N--
Q6LNI3_PHOPR/144-289                ---D-----V---L-I-D--GES-GT-GR----R----T-V-S
----K--
A0A1G8U4Y5_9RHOB/145-284            ---E-----V---L-V-T--GPT-GS-GT----A----K-V-A
----E--



KENKE
```

-

13.0.4 Turn the alignment object into an array of letters

[7]:
```
import numpy as np
from Bio import AlignIO
#align = AlignIO.read("PF05371_seed.sth", "stockholm")
align = AlignIO.read("datasets/opuntia.aln", "clustal")
align_array = np.array([list(rec) for rec in align], np.character)
print("Align shape %i by %i" % align_array.shape)
print(align_array)
```

```
Align shape 7 by 906
[[b'T' b'A' b'T' … b'A' b'G' b'A']
 [b'T' b'A' b'T' … b'A' b'G' b'A']
 [b'T' b'A' b'T' … b'A' b'G' b'A']
 …
 [b'T' b'A' b'T' … b'A' b'G' b'A']
 [b'T' b'A' b'T' … b'A' b'G' b'A']
 [b'T' b'A' b'T' … b'A' b'G' b'A']]
```

[8]:
```
[rec for rec in align]
```

[8]:
```
[SeqRecord(seq=Seq('TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAA…AGA
', SingleLetterAlphabet()), id='gi|6273285|gb|AF191659.1|AF191', name='<unknown
name>', description='gi|6273285|gb|AF191659.1|AF191', dbxrefs=[]),
 SeqRecord(seq=Seq('TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAA…AGA
', SingleLetterAlphabet()), id='gi|6273284|gb|AF191658.1|AF191', name='<unknown
name>', description='gi|6273284|gb|AF191658.1|AF191', dbxrefs=[]),
 SeqRecord(seq=Seq('TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAA…AGA
', SingleLetterAlphabet()), id='gi|6273287|gb|AF191661.1|AF191', name='<unknown
name>', description='gi|6273287|gb|AF191661.1|AF191', dbxrefs=[]),
 SeqRecord(seq=Seq('TATACATAAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAA…AGA
', SingleLetterAlphabet()), id='gi|6273286|gb|AF191660.1|AF191', name='<unknown
name>', description='gi|6273286|gb|AF191660.1|AF191', dbxrefs=[]),
 SeqRecord(seq=Seq('TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAA…AGA
', SingleLetterAlphabet()), id='gi|6273290|gb|AF191664.1|AF191', name='<unknown
name>', description='gi|6273290|gb|AF191664.1|AF191', dbxrefs=[]),
 SeqRecord(seq=Seq('TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAA…AGA
', SingleLetterAlphabet()), id='gi|6273289|gb|AF191663.1|AF191', name='<unknown
name>', description='gi|6273289|gb|AF191663.1|AF191', dbxrefs=[]),
 SeqRecord(seq=Seq('TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAA…AGA
', SingleLetterAlphabet()), id='gi|6273291|gb|AF191665.1|AF191', name='<unknown
name>', description='gi|6273291|gb|AF191665.1|AF191', dbxrefs=[])]
```

Note that this leaves the original Biopython alignment object and the NumPy array in memory as separate objects - editing one will not update the other!

[9]:
```
align_array.shape
```

[9]:
```
(7, 906)
```

-

13.0.5 SummaryInfo

-

13.0.6 consensus sequence, position specific score matrix

-

13.0.7 information content substitution

[26]:
```
summary_align = AlignInfo.SummaryInfo(align)
consensus = summary_align.dumb_consensus()
print(consensus)
my_pssm = summary_align.pos_specific_score_matrix(consensus, chars_to_ignore =␣
 ↪['N', '-'])
```

```
#print(my_pssm)
# your_pssm[sequence_number][residue_count_name]
print(my_pssm[1])
print(my_pssm[1]["A"])
```

```
TATACATTAAAGXAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAAATGAATCTAAATGATATAXGATTCCA
CTATGTAAGGTCTTTGAATCATATCATAAAAGACAATGTAATAAAGCATGAATACAGATTCACACATAATTATCTGATAT
GAATCTATTCATAGAAAAAAGAAAAAAGTAAGAGCCTCCGGCCAATAAAGACTAAGAGGGTTGGCTCAAGAACAAAGTTC
ATTAAGAGCTCCATTGTAGAATTCAGACCTAATCATTAATCAAGAAGCGATGGGAACGATGTAATCCATGAATACAGAAG
ATTCAATTGAAAAAGATCCTAATXXXTCATTGGGAAGGATGGCGGAACGAACCAGAGACCAATTCATCTATTCTGAAAAG
TGATAAACTAATCCTATAAAACTAAAATAGATATTGAAAGAGTAAATATTCGCCCGCGAAAATTCCTTTTTTATTTAAAT
TGCTCATATTTTXTTTTAGCAATGCAATCTAATAAAATATATCTATACAAAAAAAXATAGACAAACTATATATATATATA
TATATAATATATTTCAAATTXCCTTATATATCCAAATATAAAAATATCTAATAAATTAGATGAATATCAAAGAATCTATT
GATTTAGTGTATTATTAAATGTATATXTTAATTCAATATTATTATTCTATTCATTTTTATTCATTTTCAAATTTATAATA
TATTAATCTATATATTAATTTAXAATTCTATTCTAATTCGAATTCAATTTTTAAATATTCATATTCAATTAAAATTGAAA
TTTTTTCATTCGCGAGGAGCCGGATGAGAAGAAACTCTCATGTCCGGTTCTGTAGTAGAGATGGAATTAAGAAAAAACCA
TCAACTATAACCCCAAXAGAACCAGA
{'A': 7.0, 'C': 0, 'G': 0, 'T': 0}
7.0
```

[28]:
```
instances = [al.seq for al in align[:10]]
print(instances)
```

```
[Seq('TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAA…AGA',
SingleLetterAlphabet()),
Seq('TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAA…AGA',
SingleLetterAlphabet()),
Seq('TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAA…AGA',
SingleLetterAlphabet()),
Seq('TATACATAAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAA…AGA',
SingleLetterAlphabet()),
Seq('TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAA…AGA',
SingleLetterAlphabet()),
Seq('TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAA…AGA',
SingleLetterAlphabet()),
Seq('TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAA…AGA',
SingleLetterAlphabet())]
```

14 14. Biopython - Motif

-

14.0.1 Bio.motifs package included in Biopython 1.61

[36]:
```
from Bio import motifs
from Bio.Seq import Seq
```

[37]:
```
instances = [Seq("TACAA"),
             Seq("TACGA"),
             Seq("TACAA"),
             Seq("TAGAA"),
             Seq("TACAA"),
             Seq("AACGA"),
            ]
```

[38]:
```
m = motifs.create(instances)
print(m)
```

```
TACAA
TACGA
TACAA
TAGAA
TACAA
AACGA
```

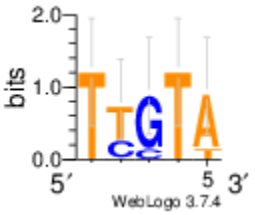
[39]:
```
m.counts
```

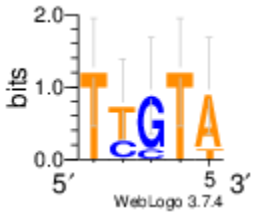
[39]:
```
{'A': [1, 6, 0, 4, 6],
 'C': [0, 0, 5, 0, 0],
 'G': [0, 0, 1, 2, 0],
 'T': [5, 0, 0, 0, 0]}
```

[41]:
```
m.counts["A", 1]
r = m.reverse_complement()
print(r.consensus)
#r.weblogo("mymotif.png")
```

```
TTGTA
```

[42]:
```
from IPython.display import Image, display
display(Image(filename="mymotif.png"))
```





-

14.0.2 Position-weight matrices

-

14.0.3 .counts

[43]:
```
pwm = m.counts.normalize(pseudocounts=0.5)
print(pwm)
```

```
       0     1     2     3     4
A:  0.19  0.81  0.06  0.56  0.81
C:  0.06  0.06  0.69  0.06  0.06
G:  0.06  0.06  0.19  0.31  0.06
T:  0.69  0.06  0.06  0.06  0.06
```

[44]:
```
pssm = pwm.log_odds()
print(pssm)
```

```
       0     1     2     3     4
A: -0.42  1.70 -2.00  1.17  1.70
C: -2.00 -2.00  1.46 -2.00 -2.00
G: -2.00 -2.00 -0.42  0.32 -2.00
T:  1.46 -2.00 -2.00 -2.00 -2.00
```

[45]:
```
background = {"A":0.3,"C":0.2,"G":0.2,"T":0.3}
pssm = pwm.log_odds(background)
print(pssm)
```

```
       0     1     2     3     4
A: -0.68  1.44 -2.26  0.91  1.44
C: -1.68 -1.68  1.78 -1.68 -1.68
G: -1.68 -1.68 -0.09  0.64 -1.68
```

```
T:   1.20  -2.26  -2.26  -2.26  -2.26
```

-

14.0.4

[46]:
```
test_seq=Seq("TACACTGCATTACAACCCAAGCATTA")
for pos, seq in m.instances.search(test_seq):
    print("%i %s " % (pos, seq))
print(m)
```

```
10 TACAA
TACAA
TACGA
TACAA
TAGAA
TACAA
AACGA
```

[47]:
```
for pos, seq in r.instances.search(test_seq):
    print("%i %s " % (pos, seq))
print(r)
```

```
TTGTA
TCGTA
TTGTA
TTCTA
TTGTA
TCGTT
```

-

14.0.5 Using the PSSM score

[48]:
```
for pos, score in pssm.search(test_seq, threshold=3.0):
    print("%d, %f " % (pos, score))
pssm.calculate(test_seq)
```

```
0, 3.643981
10, 6.759458
```

[48]:
```
array([  3.643981  ,  -8.560285  ,  -2.4004133 ,  -5.6533937 ,
        -4.2748823 ,  -0.05645879, -10.145247  ,  -3.3293302 ,
        -5.9753222 ,  -3.5703382 ,   6.759458  ,  -5.3903594 ,
```

```
        -5.8598447 ,  -0.81545067,  -0.81545067,   0.7695118 ,
        -6.3903594 ,  -3.5379167 ,   0.4255574 ,  -1.9309279 ,
       -10.145247  ,  -3.3293302 ], dtype=float32)
```

[49]:
```
m.pseudocounts = 0.1
print(m.counts)
print(m.pwm)
print(m.pssm)
```

```
       0     1     2     3     4
A:  1.00  6.00  0.00  4.00  6.00
C:  0.00  0.00  5.00  0.00  0.00
G:  0.00  0.00  1.00  2.00  0.00
T:  5.00  0.00  0.00  0.00  0.00

       0     1     2     3     4
A:  0.17  0.95  0.02  0.64  0.95
C:  0.02  0.02  0.80  0.02  0.02
G:  0.02  0.02  0.17  0.33  0.02
T:  0.80  0.02  0.02  0.02  0.02

       0     1     2     3     4
A: -0.54  1.93 -4.00  1.36  1.93
C: -4.00 -4.00  1.67 -4.00 -4.00
G: -4.00 -4.00 -0.54  0.39 -4.00
T:  1.67 -4.00 -4.00 -4.00 -4.00
```

15 15.

-

15.0.1 K-Nearest Neighbor (KNN)

[160]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.datasets import load_breast_cancer
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import seaborn as sns
sns.set()
```

[54]:
```
breast_cancer = load_breast_cancer()
```

[55]:
```
print(breast_cancer.data.shape)
X = pd.DataFrame(breast_cancer.data, columns=breast_cancer.feature_names)
```

```
(569, 30)
```

[67]:
```
print(X.loc[0:3,])
```

```
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597

   mean fractal dimension  …  worst radius  worst texture  worst perimeter  \
0                 0.07871  …         25.38          17.33           184.60
1                 0.05667  …         24.99          23.41           158.80
2                 0.05999  …         23.57          25.53           152.50
3                 0.09744  …         14.91          26.50            98.87

   worst area  worst smoothness  worst compactness  worst concavity  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869

   worst concave points  worst symmetry  worst fractal dimension
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300

[4 rows x 30 columns]
```

[163]:
```
X[['mean area', 'mean compactness']]
```

[163]:
```
   mean area  mean compactness
0      1001.0           0.27760
1      1326.0           0.07864
2      1203.0           0.15990
```

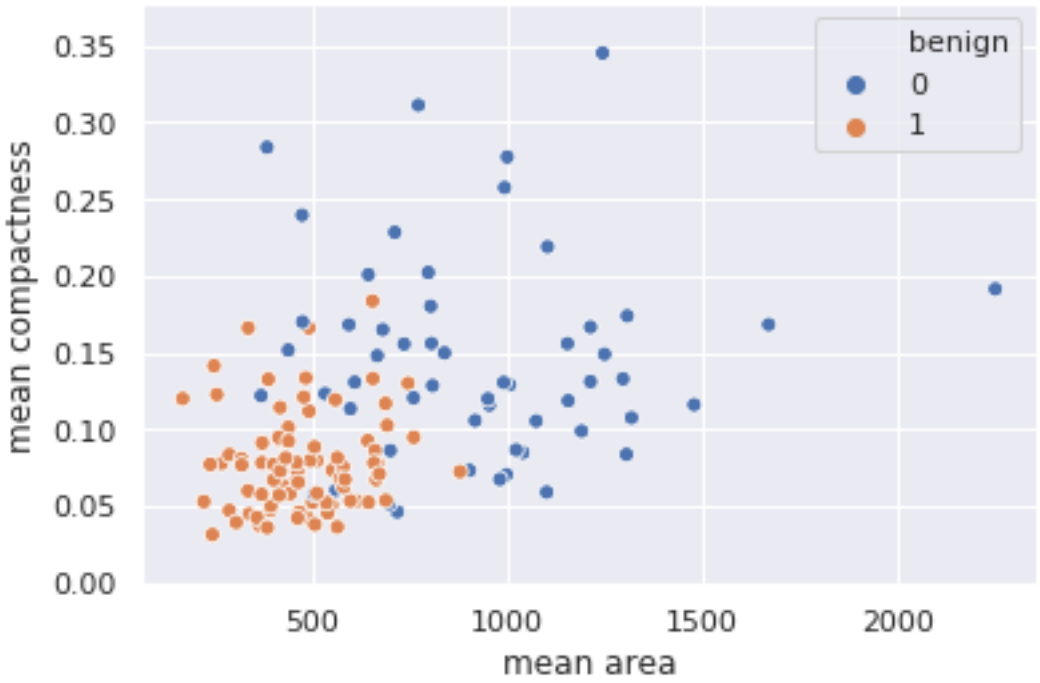
```
3        386.1           0.28390
4       1297.0           0.13280
..         …               …
564     1479.0           0.11590
565     1261.0           0.10340
566      858.1           0.10230
567     1265.0           0.27700
568      181.0           0.04362

[569 rows x 2 columns]
```

[164]:
```
y = pd.Categorical.from_codes(breast_cancer.target, breast_cancer.target_names)
y = pd.get_dummies(y, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
```

[165]:
```
knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean')
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
sns.scatterplot(
    x='mean area',
    y='mean compactness',
    hue='benign',
    data=X_test.join(y_test, how='outer')
)
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:2:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
```

[165]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f01b83efd10>
```



[166]:
```
results = pd.DataFrame([y_test["benign"].values, y_pred], index=("test","predict")).T
print(results)
```

```
     test  predict
0       1        0
1       0        0
2       1        1
3       0        0
4       0        0
..    ...      ...
138     1        1
139     1        1
140     0        0
141     0        0
142     1        1

[143 rows x 2 columns]
```

[167]:
```
results[results["test"]+results["predict"] == 1]
```

[167]:

	test	predict
0	1	0
33	0	1

```
38      0       1
63      1       0
76      0       1
77      0       1
110     0       1
127     1       0
137     1       0
```

[161]:
```
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9370629370629371
```