In [1]:
```
from collections import defaultdict
import random
```

In [2]:
```
amino_acid = ["A","R","N","D","C","Q","E","G","H","I","L","K","M","F",
"P","S","T","W","Y","V"]
```

In [3]:
```
alphabet = amino_acid + ["X"]
```

In [4]:
```
f1 = open("pa4train.txt")
data = [line.strip() for line in f1]
data = [line.split() for line in data]
train_data = [["".join([w if w in amino_acid else "X" for w in x])]+[int
(y)] for x,y in data]

f2 = open("pa4test.txt")
data_t = [line.strip() for line in f2]
data_t = [line.split() for line in data_t]
test_data = [["".join([w if w in amino_acid else "X" for w in x])]+[int(
y)] for x,y in data_t]
```

In [5]:
```
def dot(w1,w2):
    sum = 0
    for i in w1:
        if (i in w2):
            sum += w1[i]*w2[i]
    return sum
```

In [6]:
```
def add(w1,w2):
    w_new = dict()
    for i in w1:
        w_new[i] = w1[i]
    for i in w2:
        if i not in w_new:
            w_new[i] = 0
        w_new[i] += w2[i]
    return w_new
```

In [7]:
```
def mult(x,w):
    w_new = dict()
    for i in w:
        w_new[i] = x*w[i]
    return w_new
```

Q1

In [8]:
```
def string_kernel(data,p):
    phi = []

    for i in data:
        phi_x = dict()
        x = i[0]
        y = i[1]
        for j in range(0,len(x) - p + 1):
            if x[j:j+p] not in phi_x:
                phi_x[x[j:j+p]] = 0
            phi_x[x[j:j+p]] += 1
        phi += [[phi_x]+[y]]
    return phi
```

In [9]:
```
def perceptron(data,num_pass):
    w = dict()

    for i in data:
        x = i[0]
        y = i[1]
        temp = y*dot(w,x)

        if temp <= 0:
            w = add(w,mult(y,x))
    return w
```

In [10]:
```
def error(data,phi,w):
    count = 0
    for i in range(len(data)):
        t = dot(phi[i][0],w)
        sign = 1 if t > 0 else -1 if t < 0 else random.choice([-1,1])
        if (sign != data[i][-1]):
            count += 1
    return count/len(data)
```

In [11]:
```
for i in [2,3,4,5]:
    print("p =",i)
    phi = string_kernel(train_data,i)
    w = perceptron(phi,1)
    train_error = error(train_data,phi,w)
    print("train error:",train_error)

    phi_t = string_kernel(test_data,i)
    test_error = error(test_data,phi_t,w)
    print("test error:",test_error)
```

```
p = 2
train error: 0.07107438016528926
test error: 0.08179419525065963
p = 3
train error: 0.013498622589531680
test error: 0.04221635883905013
p = 4
train error: 0.008264462809917356
test error: 0.029023746701846966
p = 5
train error: 0.006336088154269973
test error: 0.04353562005277045
```

Q2

In [12]:
```
def sk_modify(data,p):
    phi = []

    for i in data:
        phi_x = dict()
        x = i[0]
        y = i[1]
        for j in range(0,len(x) - p + 1):
            phi_x[x[j:j+p]] = 1
        phi += [[phi_x]+[y]]
    return phi
```

In [13]:
```
for i in [2,3,4,5]:
    print("p =",i)
    phi = sk_modify(train_data,i)
    w = perceptron(phi,1)
    train_error = error(train_data,phi,w)
    print("train error:",train_error)

    phi_t = sk_modify(test_data,i)
    test_error = error(test_data,phi_t,w)
    print("test error:",test_error)
```

```
p = 2
train error: 0.08264462809917356
test error: 0.09762532981530343
p = 3
train error: 0.012396694214876033
test error: 0.052770448548812667
p = 4
train error: 0.007988980716253443
test error: 0.032981530343007916
p = 5
train error: 0.006060606060606061
test error: 0.04353562005277045
```

Q3

In [14]:
```
phi = string_kernel(train_data,5)
w = perceptron(phi,1)
```

In [15]:
```
len(w) < 21**5
```

Out[15]:
```
True
```

In [16]:
```
max_two = sorted([(w[i],i) for i in w],reverse = True)[:2]
max_two
```

Out[16]:
```
[(3, 'WDTAG'), (3, 'LFLNK')]
```

The corresponding substrings are WDTAG,LFLNK.