Pupylation 位点预测

代码： git clone https://github.com/lipan6461188/GPS_Pupylation.git

1.首先，遵照 PSP(7,7)的特征选取方法进行测试，每一个位点的权重都是 1，也就是对于两条序列 m 和 n。他们的分数：

$$S(m,n) = \sum_{i=0}^{15} w_i * Score(m_i, n_i)$$ 并且 S(m ,n) >= 0

其中$Score(m_i, n_i)$是 BLOSUM62 矩阵中的一个分数。

这里的w_i是每一个位点的权重：

首先，我把他们都设置成 1。即$w_i = \{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1\}$

样本的格式（左边是序列，右边代表正负样本）：

```
HKNKDQAMKQ      -
ARLKNLGVEE      +
TFWKDNQLNI      -
AALKKNDMDK      -
DMDKAKQLIA      -
MQKKNAEKQA      -
**MKILIVED      -
```

2. 预测的方法如下：

	1	2	3	.....	最后一条 N	总得分:
1	S_{11}	S_{21}	S_{31}			S_1
2	S_{12}	S_{22}	S_{32}			S_2
.....				S_{ij}		S_j
最后一条 M						S_M

第一步：（预测正样本，统计真阳性、假阴性）：

上面的表格中横纵坐标都是表示所有的正样本(M =N = 正样本个数)，其中的每一个分数 S_{ij} 都是对应的两条序列的打分，并且 $S_{ij} = S_{ji}$。计算所有的正样本之间的相互打分，得到了这样的一张表。统计最后一列：$Sj = S_{1j} + S_{2j} + ... + S_{Mj}$ 这就是正样本 j 的打分

接下来就可以统计 TP 和 FN 了：

```
j = 1;
while( j < 正样本个数M )
```

```
{
        if(Sj >= 阈值) TP++;
        else FN++;
}
```

第二步：（预测负样本，统计真阴性、假阳性）：

上面的表格横坐标表示正样本，纵坐标表示假样本（N = 正样本个数；M = 负样本个数）。对每一条负样本 i，他与正样本 j 打分以后，值是 S_{ij}。同样统计表格的右边一列 $Sj = S_{1j} + S_{2j} + ... + S_{Mj}$，他表示负样本 j 的得分：

接下来就可以统计 TN 和 FP 了：

```
j = 1;
while( j < 负样本个数M )
{
        if(Sj >= 阈值) TN++;
        else FP++;
}
```

计算 Sn 和 Sp：

$$S_n = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \quad \text{and} \quad S_p = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}.$$

改变阈值，取得不同的 Sn 和 Sp：

```
计算得到两张表格
for(阈值i = 0; i<阈值上限; i += gap)
{
    计算TP和FN
    计算TN和FP
    计算Sn和Sp
}
```

ROC 曲线绘图：

横坐标是 1 - Sp；纵坐标是 Sn

脚本运行指南：（在 run.sh 中都有）

1. 数据转换

首先由 IDs 到 Uniprot 中检索到了所有的蛋白质序列

snapshoot：

```
>sp|A0QQH7|PURA_MYCS2 Adenylosuccinate synthetase OS=Mycobacterium smegmatis (strain ATCC
700084 / mc(2)155) GN=purA PE=1 SV=1
MPAIVLIGAQWGDEGKGKATDLLGGRVQWVVRYQGGNNAGHTVVLPTGENFALHLIPSGI
LTPGVTNVIGNGVVVDPGVLLTELKGLEDRGVDTSNLLISADAHLLMPYHVAIDKVVERW
AGSKKIGTTGRGIGPCYQDKIARIGIRVADVLDEQVLAEKIEAALEFKNQVLVKIYNRKA
LEPAEVLENLLEQAEGFKHRIADARLLLNQALENDEAVLLEGSQGTLLDVDHGTYPFVTS
SNPTAGGAAVGSGIGPTRITTVLGILKAYTTRVGSGPFPTELFDEHGAYLAKTGGEVGVT
TGRARRCGWFDAVIARYATRVNGITDYFLTKLDVLSSLETVPVCVGYTVDGKRVDEMPMT
QSDIARAEPVYEELPGWWEDISGAREFEDLPAKARDYVLRLEELAGAYVSCIGVGPGRDQ
TIVRRDVLAAR
>sp|A0QS45|RL11_MYCS2 50S ribosomal protein L11 OS=Mycobacterium smegmatis (strain ATCC
700084 / mc(2)155) GN=rplK PE=1 SV=1
MAPKKKVAGLIKLQIQAGQANPAPPVGPALGQHGVNIMEFCKAYNAATESQRGNVIPVEI
TVYEDRSFTFALKTPPAAKLLLKAAGVQKGSGEPHKTKVAKVTWDQVREIAETKKADLNA
NDIDAAAKIIAGTARSMGITVE
>sp|A0QSE0|RS17_MYCS2 30S ribosomal protein S17 OS=Mycobacterium smegmatis (strain ATCC
700084 / mc(2)155) GN=rpsQ PE=1 SV=1
MADQKGPKYTPAAEKPRGRRKTAIGYVVSDKMQKTIVVELEDRKSHPLYGKIIRTTKKVK
AHDENGEAGIGDRVSLMETRPLSATKRWRLVEILEKAK
>sp|A0QUX7|ILVH_MYCS2 Acetolactate synthase small subunit OS=Mycobacterium smegmatis
(strain ATCC 700084 / mc(2)155) GN=ilvH PE=1 SV=1
MSNGTPTHTLSVLVEDKPGVLARVSSLFSRRGFNIQSLAVGATEQKDMSRMTIVVSVEDS
PLEQITKQLNKLINVIKIVEQEEDNSVSRELALIKVRADATTRGQIIEAVNLFRAKVVDV
STESLTIEATGTPEKLEALLRVLEPYGIREIAQSGVVSVSRGPRGIGAAK
>sp|A0QV10|Y2408_MYCS2 Uncharacterized oxidoreductase MSMEG_2408/MSMEI_2347
OS=Mycobacterium smegmatis (strain ATCC 700084 / mc(2)155) GN=MSMEG_2408 PE=1 SV=1
MSPRITLNDGNSIPQVGLGVWQTPAEDTERAVAAALQAGYRHIDTAAAYRNETETGRAIA
NSGVPREDIFLVTKLWNSDQGYDATLAAFDASVQRLGVDYLDLYLIHWPVPENNKFVDTF
KAFAHLRDQGRIRSIGVSNFEPEHLTTLIEETGIVPAVNQIELHPLLPQQELRDVHAKLG
IATEAWSPLGQGSLLADPVITGIAEQHGKTPAQVLIRWHIQLGNIVIPKSVNPERIASNF
```

为了便于处理，把他们转换成如下的格式：

```
>A0QV10
MSPRITLNDGNSIPQVGLGVWQTPAEDTERAVAAALQAGYRHIDTAAAYRNETETGRAIANSGVPREDIFLVTKLWNSDQGYDATLAAFDASVQRLGVDYLD
LYLIHWPVPENNKFVDTFKAFAHLRDQGRIRSIGVSNFEPEHLTTLIEETGIVPAVNQIELHPLLPQQELRDVHAKLGIATEAWSPLGQGSLLADPVITGIA
EQHGKTPAQVLIRWHIQLGNIVIPKSVNPERIASNFDVFDFELSGQDITSIASLETGKRLGPDPRTFNFTG
>A0R1Y7
MIVAGARTPVGKLMGSLKDFSGTDLGAIAIRAALEKANVPASMVEYVIMGQVLTAGAGQMPARQAAVAAGIPWDVAALSINKMCLSGIDAIALADQLIRAGE
FDVIVAGGQESMSQAPHLLPKSREGYKYGDATLVDHLAYDGLHDVFTDQPMGALTEQRNDVDKFTRAEQDEYAAQSHQKAAAAWKDGVFADEVVPVSIPQRK
GDPIEFAEDEGIRANTTAESLAGLKPAFRKDGTITAGSASQISDGAAAVIVMNKAKAEELGLTWLAEIGAHGVVAGPDSTLQSQPANAIKKAITREGITVDQ
LDVIEINEAFAAVALASTKELGVDPAKVNVNGGAIAIGHPIGMSGARIALHAALELARRGSGYAVAALCGAGGQGDALVLRR
>A0R7F9
MVILDPTLDERTVAPSLETFLNVIRKDGGTVDKVDIWGRRRLAYEIAKHAEGIYAVIDVKAEPATVSELDRQLNLNESVLRTKVLRTDKH
>O05598
MSRFTEKMFHNARTATTGMVTGEPHMPVRHTWGEVHERARCIAGGLAAAGVGLGDVVGVLAGFPVEIAPTAQALWMRGASLTMLHQPTPRTDLAVWAEDTMT
VIGMIEAKAVIVSEPFLVAIPILEQKGMQVLTVADLLASDPIGPIEVGEDDLALMQLTSGSTGSPKAVQITHRNIYSNAEAMFVGAQYDVDKDVMVSWLPCF
HDMGMVGFLTIPMFFGAELVKVTPMDFLRDTLLWAKLIDKYQGTMTAAPNFAYALLAKRLRRQAKPGDFDLSTLRFALSGAEPVEPADVEDLLDAGKPFGLR
PSAILPAYGMAETTLAVSFSECNAGLVVDEVDADLLAALRRAVPATKGNTRRLATLGPLLQDLEARIIDEQGDVMPARGVGVIELRGESLTPGYLTMGGFIP
AQDEHGWYDTGDLGYLTEEGHVVVCGRVKDVIIMAGRNIYPTDIERAAGRVDGVRPGCAVAVRLDAGHSRESFAVAVESNAFEDPAEVRRIEHQVAHEVVAE
VDVRPRNVVVLGPGTIPKTPSGKLRRANSVTLVT
>P9WGZ9
MDVDLPPPGPLTSGGLRVTALGGINEIGRNMTVFEHLGRLLIIDCGVLFPGHDEPGVDLILPDMRHVEDRLDDIEALVLTHGHEDHIGAIPFLLKLRPDIPV
VGSKFTLALVAEKCREYRITPVFVEVREGQSTRHGVFECEYFAVNHSTPDALAIAVYTGAGTILHTGDIKFDQLPPDGRPTDLPGMSRLGDTGVDLLLCDST
NAEIPGVGPSESEVGPTLHRLIRGADGRVIVACFASNVDRVQQIIDAAVALGRRVSFVGRSMVRNMRVARQLGFLRVADSDLIDIAAAETMAPDQVVLITTG
TQGEPMSALSRMSRGEHRSITLTAGDLIVLSSSLIPGNEEAVFGVIDALSKIGARVVTNAQARVHVSGHAYAGELLFLYNGVRPRNVMPVHGTWRMLRANAK
```

调用脚本：

```
perl covertSeq.pl --input 268Sequences --output sequence
```

--input 输入序列文件
--output 输出序列文件

2. 特征选取

为了便于后面的预测，从这些序列中选出所有的样本：

snapshoot：

```
EVLKGLKVAL      -
PLLKDAWQLS      -
DLTKQAEAYR      -
SLAKINPEAN      -
HKPKSEKPML      -
HKNKDQAMKQ      -
ARLKNLGVEE      +
TFWKDNQLNI      -
AALKKNDMDK      -
DMDKAKQLIA      -
MQKKNAEKQA      -
**MKILIVED      -
DVDKFTRAEQ      -
ETGKVYLDRY      -
DGAKLLKCSE      -
RTFKEVLRDK      -
LAPKGHTVNA      -
QLAKEAGCAQ      -
DMDKPLTTAE      +
QVNKDSGALD      -
VGDKVKTGSL      -
ADGKSTPIGI      -
AVNKDEEAPI      -
ASDKPGFSIV      -
DEGKAFDESI      -
QLEKMLLERA      -
TVDKVDIWGR      +
ESVKNFGEVL      +
ADLKALNDED      -
DAMKEAGINV      -
```

调用脚本：

```
perl fetchSubSeq.pl --upper 7 --down 7 --patter K --delimit 1 --input "sequence" --output "samples" --positive "POSITIVE.txt"
```

参数解释：
--upper 上游氨基酸个数
--down 下游氨基酸个数
--patter 匹配的中间氨基酸的模式
--delimit 氨基酸之间以哪种方式分隔，1 表示无分隔符
--input 输入的 fasta 文件，必须上一步转换的结果
--output 输出文件
--positive 记录正样本的文件

所谓记录正样本的文件是这样的文件，左边是蛋白的名字，右边是蛋白某个序列位点上的位置。

```
A0QQH7  292
A0QS45  101
A0QSE0  96
A0QUX7  46
A0QV10  262
```

3. 预测

脚本：

```
perl gps.pl --ubound 4 --dbound 0 --gap 0.001 --blosum BLOSUM62
--sample samples --output ROC --silent
```

--ubound　阈值上界
--dbound　阈值下界
--gap　阈值gap
--blosum　BLOSUM矩阵的文件
--sample　正负样本的文件
--output　输出文件
--silent　安静模式，不向屏幕输出

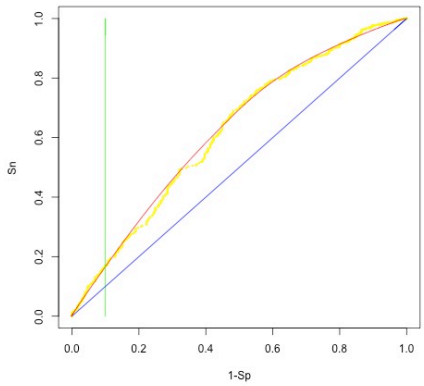
输出文件中的三列分别代表：1-Sp、Sn 和 Pr

```
0       0       0.0737071048815853
0       0.00026089225150013     0.0737249214406575
0       0.00026089225150013     0.0737249214406575
0       0.00026089225150013     0.0737249214406575
```

4. 绘图

脚本：

```
Rscript plot.R
```



为什么结果会这么差？

参考论文：GPS-PUP: computational prediction of pupylation sites in prokaryotic proteins

使用Motif length selection (MLS) 和 Weight training (WT) 的方法来进行提高：

1. Motif长度选取

为了做合适的特征选取，得到一个合适的 PSP(m, n)。把 m 和 n 从 3 开始，一直到 30 做一个测试，每一次都采用 LOO 的方法来评判结果（也就是上面的方法）。把 Sp 固定在 0.9，看 Sn 的值，越高就认为越好。

- 关于为什么要从 3 开始？

论文里面有如下的描述：

> **1) Motif length selection (MLS).** In this step, the combinations of PSP(*m*, *n*) (*m* = 1, …, 30; *n* = 1, …, 30) were extensively tested, while the optimized combination of PSP(*m*, *n*) with the highest leave-one-out (LOO) performance was determined. We fixed the *Sp* at 80% to compare the *Sn* values. The PSP(8, 18) was determined in this study.

当选取的序列是 PSP(1, ,1)时，发现正样本和负样本的数量都非常少，原因很简单：设 motif 的序列是 [XKY] 则 X 只有 20 种情况、Y 只有 20 种情况一共只有 400 种情况，也就是正样本和负样本加起来一共也只有 400（除去重复的，且一种情况既有正又有负时，判定为正样本）。这样的预测结果显然是不准确的，所以左边从 3 开始、右边从 3 开始。

这里的时间很长，结果是这样的：

阈值	m	n	Sp	Sn	Pr	正负总得分差值
0.700000	5	8	0.8984	0.2130	0.0834	71.318
0.362000	6	14	0.8925	0.2129	0.0821	58.213
0.289000	9	13	0.8961	0.2151	0.0827	59.806

这是所有的结果中最优的结果了，当 Sp 接近于 0.9 时，Sn 只有 0.21。

2. 权重选择

前面我的权重都是1，这里基于上一步的结果再对这些权重进行选择。

方法如下：

```
#权重分数
our @scores = ( 1,1,1,1,1,
                1,1,1,1,1,
                1,1,1,1,1,
                1,1,1,1,1,
                1,1,1);
```

```
do
    for(i=0;i<motif长度; i++)
    {
        @score[$i]++;
        if(LOO性能提高) { 保留;  }
            else
            {
                @score[$i] -= 2;
                if(LOO性能提高) { 保留; }
                else { @score[$i]++; }
            }
    }
until @score不再改变
```

结果如下：

①上游 5 个下游 8 个：

1 1 1 1 0 1 1 0 1 0 1 1 1 1 0.248823836905384

②上游 6 个，下游 14 个

1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 0 0 1 1   0.234206471494607

③上游 9 个，下游 13 个

1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
   0.222051282051282

提高不显著！

论文的结果是这样的：

Method	Threshold	Ac (%)	Sn (%)	Sp (%)	Pr (%)	MCC
GPS 2.2	High	85.44	33.07	90.18	23.33	0.1991
	Medium	82.51	44.88	85.91	22.35	0.2279
	Low	78.85	63.78	80.21	22.56	0.2864
GPS 2.1		85.25	31.50	90.11	22.35	0.1854
		81.40	37.80	85.34	18.90	0.1715
		77.35	44.09	80.36	16.87	0.1636

当 Sp 是 0.90 时，Sn 是 0.33。这与这个结果相差甚远。