Probabilistic Selection for Boosting

Given a set of examples $X = [x_1, \ldots, x_n]$, with corresponding selection probabilities $P = [p_1, \ldots, p_n]$, how can we randomly select a sample from $X$ according to $P$?
Probabilistic Selection for Boosting

If $X = [x_1, x_2, x_3]$ and $P = [0.89, 0.1, 0.01]$ which is a more likely sample of $X$ according to $P$?

$s_1 = [x_1, x_1, x_1]$  
$s_2 = [x_1, x_2, x_3]$  
$s_3 = [x_3, x_3, x_3]$
Probabilistic Selection for Boosting

If \( X = [x_1, x_2, x_3] \) and \( P = [0.89, 0.1, 0.01] \) which is a more likely sample of \( X \) according to \( P \)?

\[ s_1 = [x_1, x_1, x_1] \]
\[ s_2 = [x_1, x_2, x_3] \]
\[ s_3 = [x_3, x_3, x_3] \]

\( s_1 \), of course, with a probability of \((0.89)^3 = 0.70\)
Probabilistic Selection for Boosting

How to implement it?
Roulette wheel analogy
P=[0.07,0.13,0.2,0.27,0.33]
# Probabilistic Selection for Boosting

<table>
<thead>
<tr>
<th>P</th>
<th>Cumulative Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>0.13</td>
<td>0.20</td>
</tr>
<tr>
<td>0.20</td>
<td>0.40</td>
</tr>
<tr>
<td>0.27</td>
<td>0.67</td>
</tr>
<tr>
<td>0.33</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Probabilistic Selection for Boosting

<table>
<thead>
<tr>
<th>P</th>
<th>Cumulative Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>0.13</td>
<td>0.20</td>
</tr>
<tr>
<td>0.20</td>
<td>0.40</td>
</tr>
<tr>
<td>0.27</td>
<td>0.67</td>
</tr>
<tr>
<td>0.33</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Idea: Generate random number \( r \) between 0 and 1 and if \( r < 0.07 \), select \( x_1 \), else if \( r < 0.20 \) select \( x_2 \), else if \( r < 0.40 \) select \( x_3 \). etc.
Probabilistic Selection for Boosting

Select(p,n): // Returns a multiset with n examples according to P

cs = np.cumsum(p)
S=[]
for i in range(n):
    r = np.random()
    i = 0
    while r>cs[i]:
        i+=1
    S.append(i)
return S
Probabilistic Selection for Boosting

We can speed up the process if we generate all random numbers outside the loop and sort them.

Select($p,n$): // Returns a multiset with $n$ examples according to $P$

```python
    cs = np.cumsum(p)
    R = sort(np.random(n))
    S = []
    i = 0
    for r in R:
        while r > cs[i]:
            i += 1
        S.append(i)
    return S
```

Running time: $O(n \log n)$
Probabilistic Selection for Boosting

Example:
Suppose $P = [0.2, 0.1, 0.3, 0.4]$
Suppose the sorted random selection is $R = [0.15, 0.35, 0.65, 0.9]$
Trace the execution of the previous algorithm

$Cs = [0.2, 0.3, 0.6, 1]$
$S = [0, 2, 3, 3]$