Lisa Yan
CS 109

Section #7
November 13-15, 2019

# Section #7 Solutions

---

**This Week in Review**: Last week, we wrapped up a major portion of the class by using probabilistic models (with given parameters) to answer probability questions. Now we're moving into a new framework: can we start from data and look at it to learn the parameters themselves? We've seen a few specific cases before of approximating parameters; the Beta distribution approximates a Bernoulli $p$, and sample means and variances can approximate parameters of a normal. Models that attempt to capture complex data will have lots of parameters, though, and we now need a more general approach to approximating them from data. We're going to explore MLE and gradient ascent today in the context of something we're all familiar with: the Stanford Honor Code.

**The Honor Code**: The Stanford Student Honor Code is one of the policies that make our University so unique. Beyond the practical implications, it has the beautiful effect of creating a society where we all operate on the basis of mutual trust. In departments from English to Computer Science it has been decided that automated tools should be used to identify if two submissions are suspiciously similar. (Aside: these tools are never used as a basis for community standards cases–that always requires professional human opinion.) Automated tools are never perfect, and students and teachers alike worry of even the tiny probability of a false accusation. That becomes especially important for short assignments, like the ones at the start of CS106A. As active Stanford citizens we would like to further explore the chance of a false match in automated tools. In today's section we are going to specifically explore the Breakout assignment, one of the shortest in CS106A.

1. **Single Match**

   Say there are 1000 decision points when writing Breakout. Assume: Each decision point is binary. Each student makes all 1000 decisions. For each decision there is a probability $p$ that a student takes the more popular choice. What is the probability distribution for the number of matching decisions (we are going to refer to this as the "score")? If possible, could you approximate this probability?

   Let $A_i$ be the event that decision point $i$ is matched. We note that a match occurs when both students make the more popular choice or when both students make the less popular choice. $P(A_i) = P(\text{Both more popular}) + P(\text{Both less popular}) = p^2 + (1-p)^2$.

   Let $M$ be a random variable for the number of matches. It is easy to see that each of the 1000 decisions is an independent Bernoulli experiment with probability of success $p' = p^2 + (1-p)^2$. Therefore $M \sim Bin(1000, p')$.

   We can use a Normal distribution to approximate a binomial. We approximate $M \sim Bin(1000, p')$ with Normal random variable $Y \sim N(1000p', 1000p'(1-p'))$.

2. **Maximum Match**

When we look at two assignments, the probability of a false match is exceedingly small. However what happens when the score reported for one student is the max score between that student and 5000 historical Breakout submissions? Let $X_i$ be the similarity score between a student who worked on their own and student $i$. Let $Y$ be the highest match score between the student and all other submissions:

$$Y = \max_i X_i$$

The Central Limit Theorem tells us about the distribution of the sum of IID random variables. A more obscure theorem, the Fisher-Tippett-Gnedenko theorem, tells us about the *max* of IID random variables. It says that the max of IID exponential or normal random variables will be a "Gumbel" random variable.

$$Y \sim \text{Gumbel}(\mu, \beta) \qquad \text{The max of IID vars}$$

$$f(Y = k) = \frac{1}{\beta} e^{-(z+e^{-z})} \text{ where } z = \frac{k-\mu}{\beta} \qquad \text{The Gumbel PDF}$$

You are given data of 1300 students' max scores from three quarters (we believe they all worked independently): $y^{(1)} \ldots y^{(1300)}$. Come up with a method for finding the values of $\mu$ and $\beta$. You may assume you have a working version of Gradient Ascent.

For this problem, we use Maximum Likelihood Estimator (MLE) to estimate the parameters $\theta = (\mu, \beta)$.

$$L(\theta) = \prod_{i=1}^{n} f(Y^{(i)} = y^{(i)} \mid \theta)$$

$$LL(\theta) = \log \prod_{i=1}^{n} f(Y^{(i)} = y^{(i)} \mid \theta)$$

$$= \sum_{i=1}^{n} \log f(Y^{(i)} = y^{(i)} \mid \theta)$$

$$= \sum_{i=1}^{n} \log \frac{1}{\beta} e^{-(z_i + e^{-z_i})} \qquad \text{where } z_i = \frac{y^{(i)} - \mu}{\beta}$$

$$= \sum_{i=1}^{n} \log \frac{1}{\beta} + \sum_{i=1}^{n} -(z_i + e^{-z_i})$$

$$= -n \log(\beta) + \sum_{i=1}^{n} -(z_i + e^{-z_i})$$

Now we must choose the values of $\theta = (\mu, \beta)$ that maximize our log-likelihood function. To solve this argmax, we will use Gradient Ascent. First, we need to find the first derivative of the log-likelihood function with respect to our parameters.

$$\begin{aligned}
\frac{\partial LL(\theta)}{\partial \mu} &= \frac{\partial}{\partial \mu}\Big[ -n\log(\beta) + \sum_{i=1}^{n} -(z_i + e^{-z_i})\Big] \\
&= \sum_{i=1}^{n} \frac{\partial}{\partial \mu}\Big[ -(z_i + e^{-z_i})\Big] \\
&= \sum_{i=1}^{n} \frac{\partial}{\partial z_i}\Big[ -(z_i + e^{-z_i})\Big]\frac{\partial z_i}{\partial \mu} && \text{By the Chain Rule} \\
&= \sum_{i=1}^{n} \Big[ -1 + e^{-z_i}\Big]\Big[ -\frac{1}{\beta}\Big] \\
&= \frac{1}{\beta}\sum_{i=1}^{n} 1 - e^{-z_i}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial LL(\theta)}{\partial \beta} &= \frac{\partial}{\partial \beta}\Big[ -n\log(\beta) + \sum_{i=1}^{n} -(z_i + e^{-z_i})\Big] \\
&= -\frac{n}{\beta} + \sum_{i=1}^{n} \frac{\partial}{\partial \beta}\Big[ -(z_i + e^{-z_i})\Big] \\
&= -\frac{n}{\beta} + \sum_{i=1}^{n} \frac{\partial}{\partial z_i}\Big[ -(z_i + e^{-z_i})\Big]\frac{\partial z_i}{\partial \beta} && \text{By the Chain Rule} \\
&= -\frac{n}{\beta} + \sum_{i=1}^{n} \Big[ -1 + e^{-z_i}\Big]\Big[\frac{\mu - y^{(i)}}{\beta^2}\Big] && \text{Where the last term equals } \frac{\partial z_i}{\partial \beta}
\end{aligned}$$

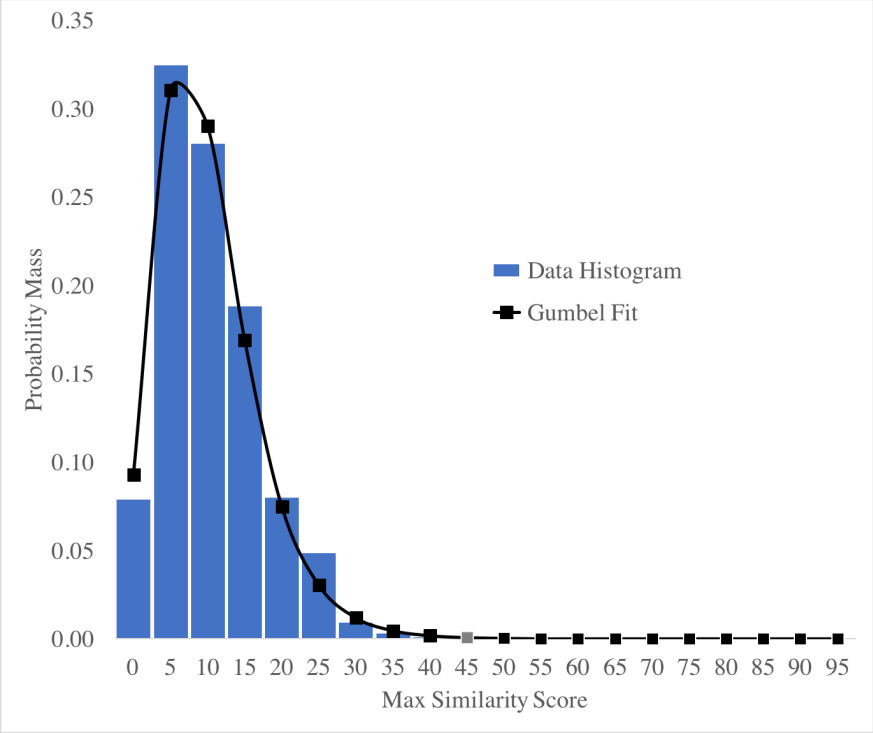
Now that we know the derivative of the log-likelihood function with respect to each parameter, we have the information we would need to perform gradient ascent. We would initialize our values of $\theta$, either to zero or to random values, and then iteratively take a small step in the direction of the gradient for each variable in $\theta$ ($\mu$ and $\beta$) and recalculate the gradient until the gradient approaches zero.

3. **Understanding**

After running your algorithm from part two, you are left with values for parameters. When you plot the Gumbel distribution with those parameters it gives a convincing fit for the histogram of our data: $y^{(1)} \ldots y^{(1300)}$. The parameters you found were:

$$Y \sim \text{Gumbel}(\mu = 9.0, \beta = 5.2)$$

And plotted against the histogram of scores from the three quarters, there is a nice fit:




A student has a max similarity score of 90. What is the probability of a similarity score that high (or higher) given that they worked on their own? You can look up the Gumbel distribution online if it helps!

```
from scipy.stats import gumbel_r
print(1 - gumbel_r.cdf(90, 9, 5.2))
```

So, $P(Y >= 90) = 0.00000017180200395650047$, or nearly 1 in 6 million.