Priority sampling test

Jacob Hoover Vigly

2025-02-26

```
%load_ext autoreload
%autoreload complete --print
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

```
from priority import PrioritySampler, normalize
import scipy, numpy as np, pandas as pd, plotly.express as px
pd.options.plotting.backend = "plotly"
import plotly.io as pio
pio.renderers.default = 'svg'
```

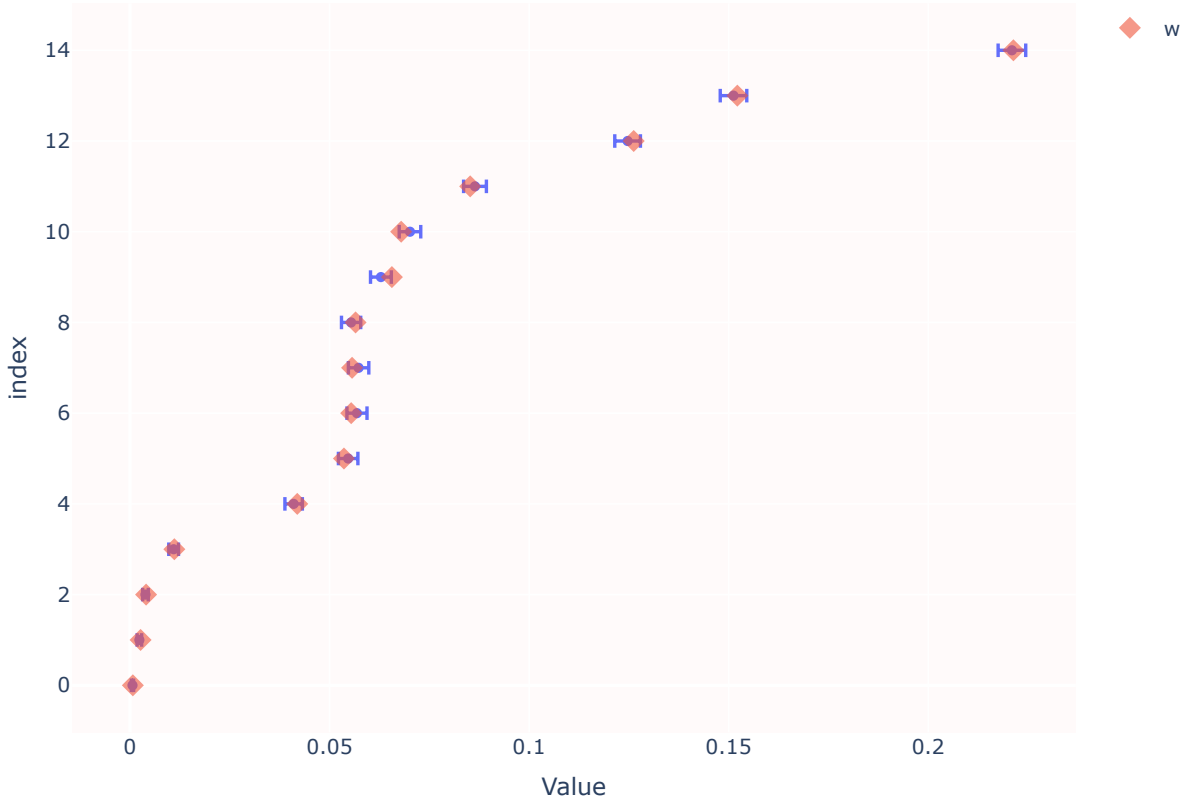
```
# initialize a priority sampler
PS = PrioritySampler(kind="gumbel", use_scipy=False)

dim, sparse_m = 15, 5
w = np.random.dirichlet(np.ones(dim))
w.sort()

samples = np.array([PS.weighted_sample(w, sparse_m) for _ in range(5000)])

stats = scipy.stats.describe(samples, axis=0)
df = pd.DataFrame({
    'Actual': w,
    'Estimate mean': stats.mean,
    'Estimate sem': 1.96 * np.sqrt(stats.variance / stats.nobs)
}).reset_index()
fig = px.scatter(
    df, x='Estimate mean', y='index', error_x='Estimate sem',
```

```
    labels = {"Estimate mean": "Value"}
).add_scatter(
    x=df['Actual'], y=df.index,
    mode='markers', name='w', marker=dict(symbol='diamond', size=10, opacity=0.6),
)
fig.update_layout(plot_bgcolor='snow').show()
```




```
if 0:
    rng = np.random.default_rng()
    bootstrap_results = scipy.stats.bootstrap((samples,), np.mean, confidence_level=0.95, ra
    df['m'] = np.mean(bootstrap_results.bootstrap_distribution, axis=1)
    df['CIl'] = bootstrap_results.confidence_interval.low
    df['CIh'] = bootstrap_results.confidence_interval.high
    df['se'] = bootstrap_results.standard_error
    display(df)
    fig.add_scatter(
        x=df.m,
```

```
        y=df.index,
        error_x=dict(
            array=df.CIh - df.m,
            arrayminus=df.m - df.CIl
        ),
        mode='markers',
        showlegend=False
    )
    fig.show()
```

The "Gumbel-max trick" is the same idea as the "exp-min trick", which is the same idea again as the idea in "Priority sampling". These are all the same algorithm.

Both you perturb your probabilities with random noise, then sort them by size, record the $(k+1)^{\text{th}}$ value as a threshold, and return a sample consisting of the k items whose weights surpassed that threshold, and weight them by dividing the weight by an inclusion probability.

There are only two steps in the algorithm where the differences are, they are (1) where we sample a perturbed version (call it z_i) of each weight (w_i), and (2) where we calculate the inclusion probability $q_i(\tau)$ to make the importance weight for the sample, given the threshold τ. That is the steps that look like this, for each i:

$$\begin{aligned} z_i &\sim \text{SomeDistribution}(w_i) \\ \tau &\leftarrow \text{top/bottom-(k+1)}^{\text{th}} \text{ value in } z \\ S &\leftarrow \{i : z_i \text{ is past threshold } \tau\} \\ q_i(\tau) &\leftarrow \Pr(z_i \text{ passes threshold} \mid \text{threshold} = \tau) \end{aligned}$$

For each case (dropping the $_i$ subscripts):

Priority sampling (or, I guess "Uniform-min")

$$\begin{aligned} z &\sim \text{Uniform}(\text{low} = 0, \text{high} = \frac{1}{w}) \\ q(\tau) &\leftarrow \Pr(z < \tau) = \text{CDF}_z(\tau) \\ &= \min(1, w \cdot \tau) \end{aligned}$$

Exp-min

$$\begin{aligned} z &\sim \text{Exponential}(\text{scale} = \frac{1}{w}) \\ q(\tau) &\leftarrow \Pr(z < \tau) = \text{CDF}_z(\tau) \\ &= 1 - e^{-w \cdot \tau} = -\text{expm1}(-w \cdot \tau) \end{aligned}$$

Gumbel-max

$$
\begin{aligned}
z &\sim \cdot\text{Gumbel}_r(\text{loc} = \log(w), \text{scale} = 1) \\
q(\tau) \leftarrow \Pr(z > \tau) &= 1 - \text{CDF}_z(\tau) = \text{sf}_z(\tau) \\
&= 1 - e^{-e^{\log(w)-t}} = -\text{expm1}(-w \cdot e^{-\tau})
\end{aligned}
$$

Coding a nice common version that makes the similarities obvious, you run into one annoying thing: the Gumbel max trick uses the *top* k (largest values), while the other two use *bottom* k (smallest values). This means that while q is the CDF in the first two cases, it is the survival function (1-CDF) in the Gumbel-max case.

To make them all fit in the same pattern, we can just change the Gumbel-max to Gumbel-min. Use *negative* of our Gumbel-perturbed weights (meaning we'll have our threshold be the $k+1^{\text{th}}$ from lowest value). And then that means our z's are distributed according to a so-called 'right-skewed' Gumbel distribution rather than the standard left-skewed one. So, the

Gumbel-min

$$
\begin{aligned}
&(z \sim -1 \cdot \text{Gumbel}_r(\text{loc} = \log(w), \text{scale} = 1)) \\
\implies\ & z \sim \text{Gumbel}_l(\text{loc} = \log(\frac{1}{w}), \text{scale} = 1) \\
q(\tau) \leftarrow \Pr(z < \tau) &= \text{CDF}_z(\tau) \\
&= 1 - e^{-e^{\log(w)+\tau}} = -\text{expm1}(-w \cdot e^{\tau})
\end{aligned}
$$

Here's a plot of the difference. It would be easy to make the mistake of just using the same CDF, but you'd get slightly biased weights.

Here's a plot of the Gumbel left- vs right-skewed.

```
# Plot PDFs and CDFs of Gumbel distributions
import plotly.express as px
import numpy as np

def plot_gumbel_funcs(m=0):
    x = np.linspace(-5, 5, 200)
    # PDFs
    pdf_right = np.exp(m - x - np.exp(m - x))
    pdf_left = np.exp(-m + x - np.exp(-m + x))
    # CDFs
    cdf_right = np.exp(-np.exp(m - x))
    cdf_left = 1 - np.exp(-np.exp(-m + x))

    df = pd.DataFrame({
        'x': np.concatenate([x, x, x, x]),
```

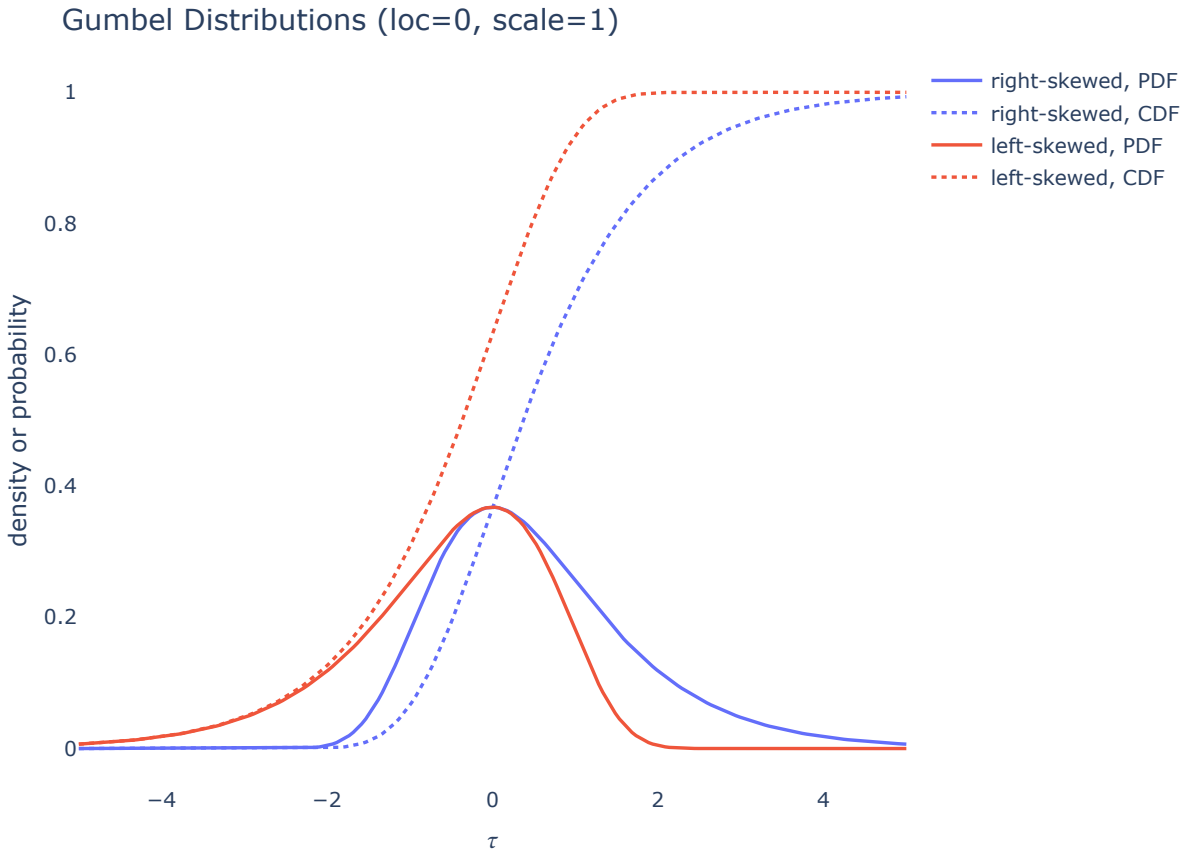
```
        'y': np.concatenate([pdf_right, pdf_left, cdf_right, cdf_left]),
        'type': ['PDF']*len(x) + ['PDF']*len(x) + ['CDF']*len(x) + ['CDF']*len(x),
        'distribution': ['right-skewed']*len(x) + ['left-skewed']*len(x) +
                        ['right-skewed']*len(x) + ['left-skewed']*len(x)
    })

    fig = px.line(df, x='x', y='y', color='distribution', line_dash='type',
                  title=f'Gumbel Distributions (loc={m}, scale=1)')

    fig.update_layout(
        xaxis_title=' ',
        yaxis_title='density or probability',
        legend_title=None,
        plot_bgcolor='rgba(0,0,0,0)',
        paper_bgcolor='rgba(0,0,0,0)'
    )

    return fig

# Create interactive plot with default m=0
plot_gumbel_funcs(m=0)
```



$$\text{Gumbel}_r\text{pdf}(\text{loc}=m,\text{scale}=1)=e^{m-x-e^{m-x}}$$
$$\text{Gumbel}_r\text{CDF}(\text{loc}=m,\text{scale}=1)=e^{-e^{m-x}}$$
$$\text{Gumbel}_l\text{pdf}(\text{loc}=m,\text{scale}=1)=e^{m+x-e^{m+x}}$$
$$\text{Gumbel}_l\text{CDF}(\text{loc}=m,\text{scale}=1)=1-e^{-e^{m+x}}$$