STA 561 Homework 3

March 20, 2023

Authors
- Alonso Guerrero Castaneda (UID: 1194613)
- Eli Gnesin (UID: 1172961)
- Tommy Misikoff (UID: 1166813)
- Sanskriti Purohit (UID: 1179957)
- Will Tirone (UID: 1130904)

TA: Rick Presman

Question 1

Consider data $(T_i, \delta_i)_{i=1}^n$ as data and consider the Kaplan-Meier estimator $\hat{S}_n(t) = \prod_{j:t_j<t}(1 - \frac{d_j}{n_j})$ where n_j are the number of subjects still alive at time t_j and d_j are the number of subjects who died at time t_j and j is the index of observed event times.

0.0.1 Challenges Faced:

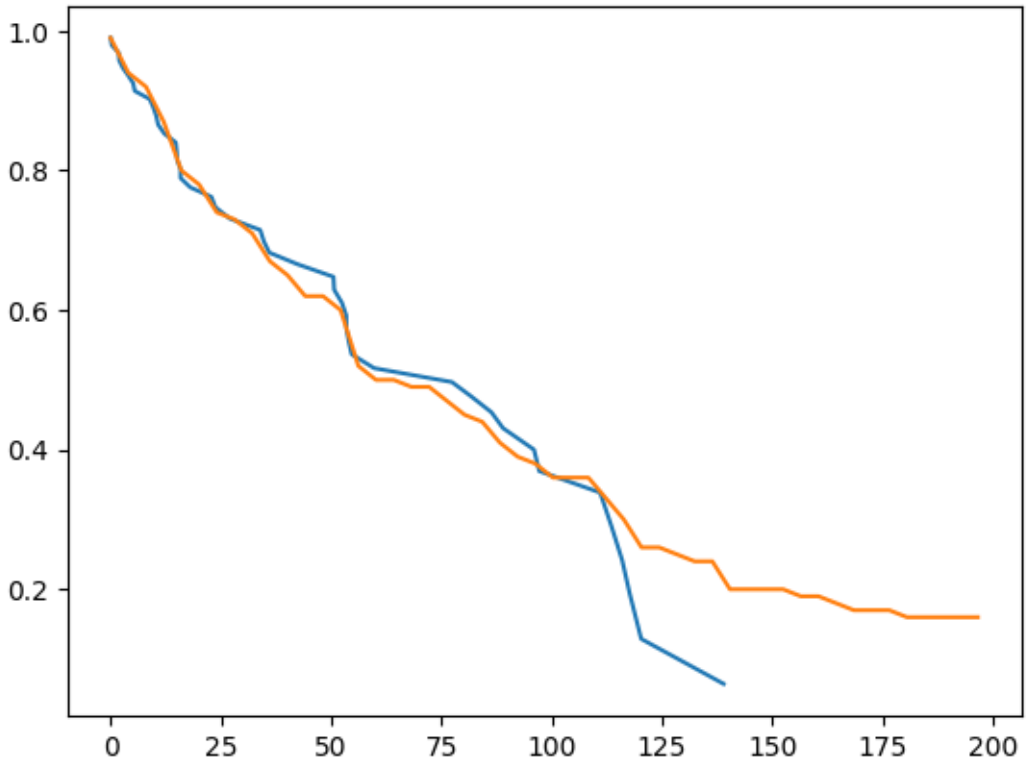
In case of "km_streaming.py", it lacked accountability for risk values where delta = 0 and T_i > t_j appears before t_j in the dataset. One example can be taken from the first two elements of our observed data. They should account to risk values within the dataset but their risk values are never accounted for in the stream since we start storing for the values from our third element.

We attempted to resolve this issue by utilizing Reservoir Sampling to store a subset of our dataset to imitate and account for the behaviour of our data. This improves the estimation of our Kaplan-Meier estimator for most of our data-points with a few outliers as exceptions. This can be attributed to the fact that we are only looking at a subset of our actual data which may not account for the variance.

[ ]:
```
import numpy as np
import numpy.random as nr
from typing import Optional
import matplotlib.pyplot as plt
from statsmodels.distributions.empirical_distribution import ECDF
import tqdm
import random
import pdb


def streaming_km(
        observed_times: np.ndarray,
```

```
        censoring_ind: np.ndarray,
        failure_set: Optional[dict] = {},
        risk_set: Optional[dict] = {}
) -> dict:
    """
    Compute the Kaplan-Meier estimator and sufficient statistics with
    single sweep through the data
    :param observed_times: 1D np.array of observed times, i.e., the
        minimum of failure and censoring times
    :param censoring_ind: 1D np.array of indicators that an observation
        was censored
    :param failure_set: a dictionary mapping failure times to
        the number of observed failures at that time, if None an empty
        dictionary is created.  This allows us to sequentially process
        multiple datasets.  Note that we do not changed passed dictionaries.
    :param risk_set: a dictionary mapping failure times to the
        number of individuals at risk.
    :return: dictionary with key --> value pairs:
        failure_set -> dictionary with keys corresponding to unique failure
            times and values corresponding to number of observed failures at
            that (key) time
        risk_set -> dictionary with keys corresponding to unique failure
            times and values corresponding to number of individuals at risk
            at that (key) time
        km -> dictionary with keys corresponding to unique failure times
            and values corresponding to Kaplan-Meier estimator of
            survivor function.
    """

    local_failure_set = failure_set.copy()
    local_risk_set = risk_set.copy()
    res_t = []
    res_delta = []
    K = 10
    for x,(t, delta) in enumerate(zip(observed_times, censoring_indicator)):
        if x < K:
                res_t.append(t)
                res_delta.append(delta)
        else:
            s = random.randint(0,x)
            if s<K:
                res_t[s] = t
                res_delta[s] = delta
        if delta == 1:
            if t not in local_failure_set:
                local_failure_set[t] = 0.0
            local_failure_set[t] += 1.0
```

```
            if t not in local_risk_set:
                local_risk_set[t] = 0.0
            unique_failure_times = np.sort(list(local_failure_set.keys()))
            t_index = np.where(unique_failure_times == t)[0][0]
            if t_index < len(unique_failure_times) - 1: # not last
                local_risk_set[t] += \
                    local_risk_set[unique_failure_times[t_index+1]]
            else:
                local_risk_set[t] += round(x*sum(res_t>=t)/len(res_t))
        risk_times = [w for w in local_risk_set.keys() if w <= t]
        for at_risk_time in risk_times:
            local_risk_set[at_risk_time] += 1.0


    # Compute KM estimator
    unique_failure_times = np.sort(list(local_failure_set.keys()))
    discrete_hazard = np.array(
        [local_failure_set[j]/local_risk_set[j] for j in unique_failure_times]
    )
    km = np.cumprod(1-discrete_hazard)
    km_dict = dict(zip(unique_failure_times, km))
    return {"failure_set": local_failure_set, "risk_set": local_risk_set,
            "km": km_dict}


if __name__ == "__main__":
    # Generate right-censored data
    nr.seed(1) # for replicability
    n = 100
    true_failure_times = nr.exponential(size=n)*100
    censoring_times = nr.exponential(size=n)*100

    # Note that you could do this in one step rather than two
    observed_times = np.array(
        [min(t, c) for t, c in zip(true_failure_times, censoring_times)]
    )
    censoring_indicator = np.array(
        [t <= c for t, c in zip(true_failure_times, censoring_times)]
    )
    km_est = streaming_km(observed_times, censoring_indicator)
    plt.plot(km_est["km"].keys(), km_est["km"].values())
    ecdf = ECDF(true_failure_times)(
        np.linspace(min(observed_times), max(observed_times))
    )
    plt.plot(np.linspace(min(observed_times), max(observed_times)), 1-ecdf)
    plt.show()
```



Question 2

$\hat{\beta}_n^\lambda$ as the LASSO Estimator Consider observed iid data $(\mathbf{X}_i, Y_i)_{i=1}^n$ and with $\lambda > 0$, define $(\hat{\mathbf{u}}_n, \hat{\mathbf{v}}_n) = \arg\min_{(\mathbf{u},\mathbf{v})} \mathbb{P}_n\{Y - (\mathbf{u} \circ \mathbf{v})^T \mathbf{X}\}^2 + \lambda||\mathbf{u}||^2/2 + \lambda||\mathbf{v}||^2/2$ with $\hat{\beta}_n^\lambda = \hat{\mathbf{u}}_n \circ \hat{\mathbf{v}}_n$. Now, we note that since $\circ$ is the element-wise multiplication of vectors, we can also consider the symbol / to be the element-wise division of vectors, such that $\mathbf{u} = \beta/\mathbf{v}$.

Then:

$$\begin{aligned}(\hat{\mathbf{u}}_n, \hat{\mathbf{v}}_n) &= \arg\min_{(\mathbf{u},\mathbf{v})} \mathbb{P}_n\{Y - (\mathbf{u} \circ \mathbf{v})^T \mathbf{X}\}^2 + \lambda||\mathbf{u}||^2/2 + \lambda||\mathbf{v}||^2/2 \\ &= \min_{,\mathbf{v}} \mathbb{P}_n\{Y - ((\beta/\mathbf{v}) \circ \mathbf{v})^T \mathbf{X}\}^2 + \lambda||\beta/\mathbf{v}||^2/2 + \lambda||\mathbf{v}||^2/2 \\ &= \min_{,\mathbf{v}} \mathbb{P}_n\{Y - \beta^T \mathbf{X}\}^2 + \lambda||\beta/\mathbf{v}||^2/2 + \lambda||\mathbf{v}||^2/2\end{aligned}$$

Now, rather than minimize both β and $\mathbf{v}$ simultaneously, we can first consider minimizing over $\mathbf{v}$ and then over β:

$$\begin{aligned}&= \min_\beta(\min_\mathbf{v} \mathbb{P}_n\{Y - \beta^T \mathbf{X}\}^2 + \lambda||\beta/\mathbf{v}||^2/2 + \lambda||\mathbf{v}||^2/2) \\ &= \min_\beta(\mathbb{P}_n\{Y - \beta^T \mathbf{X}\}^2 + (\min_\mathbf{v}(\lambda||\beta/\mathbf{v}||^2/2 + \lambda||\mathbf{v}||^2/2)) \\ &= \min_\beta(\mathbb{P}_n\{Y - \beta^T \mathbf{X}\}^2 + \lambda(\min_\mathbf{v}(||\beta/\mathbf{v}||^2 + ||\mathbf{v}||^2)/2)\end{aligned}$$

To find the minimum over $\mathbf{v}$ for $||\beta/\mathbf{v}||^2 + ||\mathbf{v}||^2$, we take the derivative with respect to $\mathbf{v}$ and set it to 0.

$$\begin{aligned}
0 &= \frac{\delta}{\delta \mathbf{v}} ||\beta/\mathbf{v}||^2 + ||\mathbf{v}||^2 \\
0 &= -\frac{\beta}{\mathbf{v}}\frac{\beta}{\mathbf{v}^2} + 2v = -\frac{2\beta^2}{\mathbf{v}^3} + 2v = 2v - \frac{2\beta^2}{v^3} \\
0 &= 2\mathbf{v}^4 - 2\beta^2\mathbf{v}^4 = \beta^2 \\
\mathbf{v}^4 &= \beta^2 \\
\mathbf{v}^2 &= |\beta|
\end{aligned}$$

Thus, $||\beta/\mathbf{v}||^2 + ||\mathbf{v}||^2$ is minimized at $\mathbf{v}^2 = |\beta|$. Now we find the actual value of $\min_{\mathbf{v}}(||\beta/\mathbf{v}||^2 + ||\mathbf{v}||^2)/2$ by using this minimum $\mathbf{v}$ which we will refer to as $\tilde{\mathbf{v}}$.

$$\begin{aligned}
\min_{\mathbf{v}}(||\beta/\mathbf{v}||^2 + ||\mathbf{v}||^2) &= ||\beta/\tilde{\mathbf{v}}||^2 + ||\tilde{\mathbf{v}}||^2 \\
&= \sum_{j=1}^{p} (\beta_j^2/\tilde{\mathbf{v}}_j^2 + \tilde{\mathbf{v}}_j^2) \\
&= \sum_{j=1}^{p} (\beta_j^2/|\beta_j| + |\beta_j|) = \sum_{j=1}^{p} (|\beta_j| + |\beta_j|) \\
&= \sum_{j=1}^{p} 2|\beta_j| = 2\sum_{j=1}^{p} |\beta_j| \\
&= 2||\beta||_1
\end{aligned}$$

Plugging this result into the earlier equation, we get:

$$\begin{aligned}
\min_{\beta}(\mathbb{P}_n\{Y - \beta^T\mathbf{X}\}^2 + \lambda(\min_{\mathbf{v}}(||\beta/\mathbf{v}||^2 + ||\mathbf{v}||^2)/2) &= \min_{\beta}(\mathbb{P}_n\{Y - \beta^T\mathbf{X}\}^2 + \lambda(2||\beta||_1)/2) \\
&= \min_{\beta}(\mathbb{P}_n\{Y - \beta^T\mathbf{X}\}^2 + \lambda||\beta||_1
\end{aligned}$$

This matches the forumula for lasso with λ as the tuning parameter, thus $\hat{\beta}_n^{\lambda} = \hat{\mathbf{u}}_n \circ \hat{\mathbf{v}}_n$ is the lasso estimator with λ as the tuning parameter.

Alternating Ridge Regression Once we recognize that $\hat{\beta}_n^{\lambda}$ is the LASSO estimator with tuning parameter λ, we can now consider how to implement a lasso estimator from this representation. First, note that since multiplication is commutative under the real numbers, $\mathbf{u} \circ \mathbf{v} = \begin{pmatrix} u_1 * v_1 & ... & u_p * v_p \end{pmatrix} = \begin{pmatrix} v_1 * u_1 & ... & v_p * u_p \end{pmatrix} = \mathbf{v} \circ \mathbf{u}$, and therefore $(\mathbf{u} \circ \mathbf{v})\mathbf{X} = (\mathbf{v} \circ \mathbf{u})\mathbf{X}$. Now consider the initial definition of the minimization function given above. Consider an initial $\mathbf{v}_0$. Now, we have:

$$\begin{aligned}
\hat{\mathbf{u}_1} &= \arg\min_{\mathbf{u}} \mathbb{P}_n\{Y - (\mathbf{u} \circ \mathbf{v}_0)^T\mathbf{X}\}^2 + \lambda||\mathbf{u}||^2/2 + \lambda||\mathbf{v}_0||^2/2 \\
&= \arg\min_{\mathbf{u}} \mathbb{P}_n\{Y - (\mathbf{u} \circ \mathbf{v}_0)^T\mathbf{X}\}^2 + \lambda||\mathbf{u}||^2/2 + const.
\end{aligned}$$

Now, we can consider the matrix $\mathbf{X}^*$, defined by taking the elements v_i of $\mathbf{v}_0$ and multiplying v_i by the i^{th} row of $\mathbf{X}$. In this construction, $\mathbf{uX}^* = (\mathbf{u} \circ \mathbf{v}_0)\mathbf{X}$. Then we have $\hat{\mathbf{u}_1} = \arg\min_{\mathbf{u}} \mathbb{P}_n\{Y - \mathbf{u}^T\mathbf{X}^*\}^2 + \lambda||\mathbf{u}||^2/2 + const.$. Now, since we are minimizing with respect to $\mathbf{u}$, and the constant $\lambda||\mathbf{v}_0||^2/2$ does not contain $\mathbf{u}$, we can ignore it with respect to the minimization (since for every $\mathbf{u}$, we would simply be adding on the same extra term). Then, we have $\hat{\mathbf{u}_1} = \arg\min_{\mathbf{u}} \mathbb{P}_n\{Y - \mathbf{u}^T\mathbf{X}^*\}^2 + \lambda||\mathbf{u}||^2/2$, which is simply the equation that minimizes to Ridge Regression (with $\lambda/2$ instead of λ, but we could simply let $\lambda^* = \lambda/2$ to avoid this). As such, we have used this representation to minimize $\mathbf{u}$ given a fixed $\mathbf{v}$ by Ridge Regression.

Now, with $\hat{\mathbf{u}_1}$, consider the matrix $\mathbf{X}^\circ$, defined by taking the elements u_i of $\hat{\mathbf{u}_1}$ and multiplying u_i by the i^{th} row of $\mathbf{X}$. With this construction, $\mathbf{vX}^\circ = (\mathbf{v} \circ \hat{\mathbf{u}_1})\mathbf{X}$. Then, we can consider the minimization of $\mathbf{v}$:

$$\begin{aligned}\hat{\mathbf{v}_1} &= \arg\min_{\mathbf{v}} \mathbb{P}_n\{Y - (\mathbf{v} \circ \hat{\mathbf{u}_1})^T\mathbf{X}\}^2 + \lambda||\hat{\mathbf{u}_1}||^2/2 + \lambda||\mathbf{v}||^2/2 \\ &= \arg\min_{\mathbf{v}} \mathbb{P}_n\{Y - \mathbf{v}^T\mathbf{X}^\circ\}^2 + \lambda||\mathbf{v}||^2/2 + \lambda||\hat{\mathbf{u}_1}||^2/2\end{aligned}$$

Now we have $\hat{\mathbf{v}_1} = \arg\min_{\mathbf{v}} \mathbb{P}_n\{Y - \mathbf{v}^T\mathbf{X}^\circ\}^2 + \lambda||\mathbf{v}||^2/2 + const.$, and as with the minimization of $\mathbf{u}$ previously, since we are minimizating with respect to $\mathbf{v}$, $\lambda||\hat{\mathbf{u}_1}||^2/2$ is a constant, and so we can ignore it for the sake of minimization, and thus we instead have $\hat{\mathbf{v}_1} = \mathbb{P}_n\{Y - \mathbf{v}^T\mathbf{X}^\circ\}^2 + \lambda||\mathbf{v}||^2/2$, which is the equation that minimizes Ridge Regression. As such, we have also minimized $\mathbf{v}$ given $\hat{\mathbf{u}_1}$ by Ridge Regression.

We can now repeat the above process to find $\hat{\mathbf{u}_i}, \hat{\mathbf{v}_i}$ through a series of alternating Ridge Regressions, first minimizing $\mathbf{u}$ using $\hat{\mathbf{v}_{i-1}}$, and then minimizing $\mathbf{v}$ using $\hat{\mathbf{u}_i}$ It is then reasonable to expect that at each iteration i, we would then check if $(\hat{\mathbf{u}_i}, \hat{\mathbf{v}_i})$ actually minimizes the original equation, and we would stop when the original equation was minimized (without a closed form, we could do multiple starts of the with different $\mathbf{v}_0$ and accept the best minimized function to give us $(\hat{\mathbf{u}}_n, \hat{\mathbf{v}}_n)$, which can then be used to calculate $\hat{\beta}_n^\lambda$. As such, we have shown that, using this representation, we can implement lasso as a sequence of alternating Ridge Regressions using stochastic approximation methods.

It is important to note that the above method does not include any dampening. We could choose to include dampening, within the above framework, by initializing $\mathbf{u}_0, \mathbf{v}_0$ and then, for any iteration i, calculating $\mathbf{u}_i = \mathbf{u}_{i-1} + \alpha_i\mathbf{u}^*$ where $\mathbf{u}^*$ is the solution of the minimization with respect to $\mathbf{u}$ using $\mathbf{v}_{i-1}$, and then calculating $\mathbf{v}_i = \mathbf{v}_{i-1} + \alpha_i\mathbf{v}^*$ where $\mathbf{v}^*$ is the solution of the minimization with respect to $\mathbf{v}$ using $\mathbf{u}_i$, with decreasing step sizes α_i that satisfy the conditions $\sum_{i\geq 1}\alpha_i = \infty$ and $\sum_{i\geq 1}\alpha_i^2 < \infty$.

Resources and Notes

1. Peter D. Hoff, Lasso, fractional norm and structured sparse estimation using a Hadamard product parametrization, Computational Statistics & Data Analysis, Volume 115, 2017, Pages 186-198, ISSN 0167-9473, https://doi.org/10.1016/j.csda.2017.06.007. (https://www.sciencedirect.com/science/article/pii/S0167947317301469) Abstract: Using a multiplicative reparametrization, it is shown that a subclass of Lq penalties with q less than or equal to one can be expressed as sums of L2 penalties. It follows that the lasso and other norm-penalized regression estimates may be obtained using a very simple and intuitive alternating ridge regression algorithm. As compared to a similarly intuitive EM algorithm for Lq optimization, the proposed algorithm avoids some numerical instability issues and is

also competitive in terms of speed. Furthermore, the proposed algorithm can be extended to accommodate sparse high-dimensional scenarios, generalized linear models, and can be used to create structured sparsity via penalties derived from covariance models for the parameters. Such model-based penalties may be useful for sparse estimation of spatially or temporally structured parameters. Keywords: Cyclic coordinate descent; Generalized linear model; Linear regression; Optimization; Ridge regression; Sparsity; Spatial autocorrelation