IO Problem Set 2 (GPV)

Chris Ackerman*

December 1, 2021

Abstract

This problem set asked us to write code to implement the GPV algorithm on bid data from John's website. My write-up first outlines the approach I took, then has a brief discussion of data features/playing around with kernels, and finally plots the estimated density of private valuations against the observed bids. The appendix contains the full program that I ran.

1 Overview

For this problem set, I

1. Loaded the bid data from John's website.
2. Graphed the distribution of bids by each bidder to get a sense of what the data looked like.
3. Fit a nonparametric kernel to the aggregate bid data to see how it looked.
4. Implemented the GPV algorithm on the data, by manually implementing a normal kernel density estimator of the distribution of bids and then inverting it to get estimated valuations.

These function calls omit some helper functions for the GPV algorithm.

```
if __name__ == '__main__':
    my_dir = "/home/chris/files/school/ucla/second_year/fall/io/pset2"
    os.chdir(my_dir)
    bid_data = load_bid_data(filename="PS3Data.csv")
    graph_bidder_data(bid_data=bid_data, filename='bidder_histogram.pdf', n_bins=50)
    graph_bids(bid_data=bid_data, filename='bid_density.pdf', bandwidth=0.5)
    density_support = np.linspace(2, 10, 500)
    estimated_density = estimate_values_density(
                                    B = bid_data[['Bidder 1', 'Bidder 2']].to_numpy()/2,
                                    kg = generate_uniform_kernel(),
                                    rhog = 1,
                                    kf = generate_uniform_kernel(),
                                    hg = 0.1,
                                    hf = 0.1,
                                    grid = density_support
    )
    graph_estimated_density(
                            density_support=density_support,
                            estimated_density=estimated_density,
                            filename='estimated_density.pdf'
    )
```

*I worked on this problem set with Luna Shen, David Kerns, and Benedikt Graf

2 Bid Data

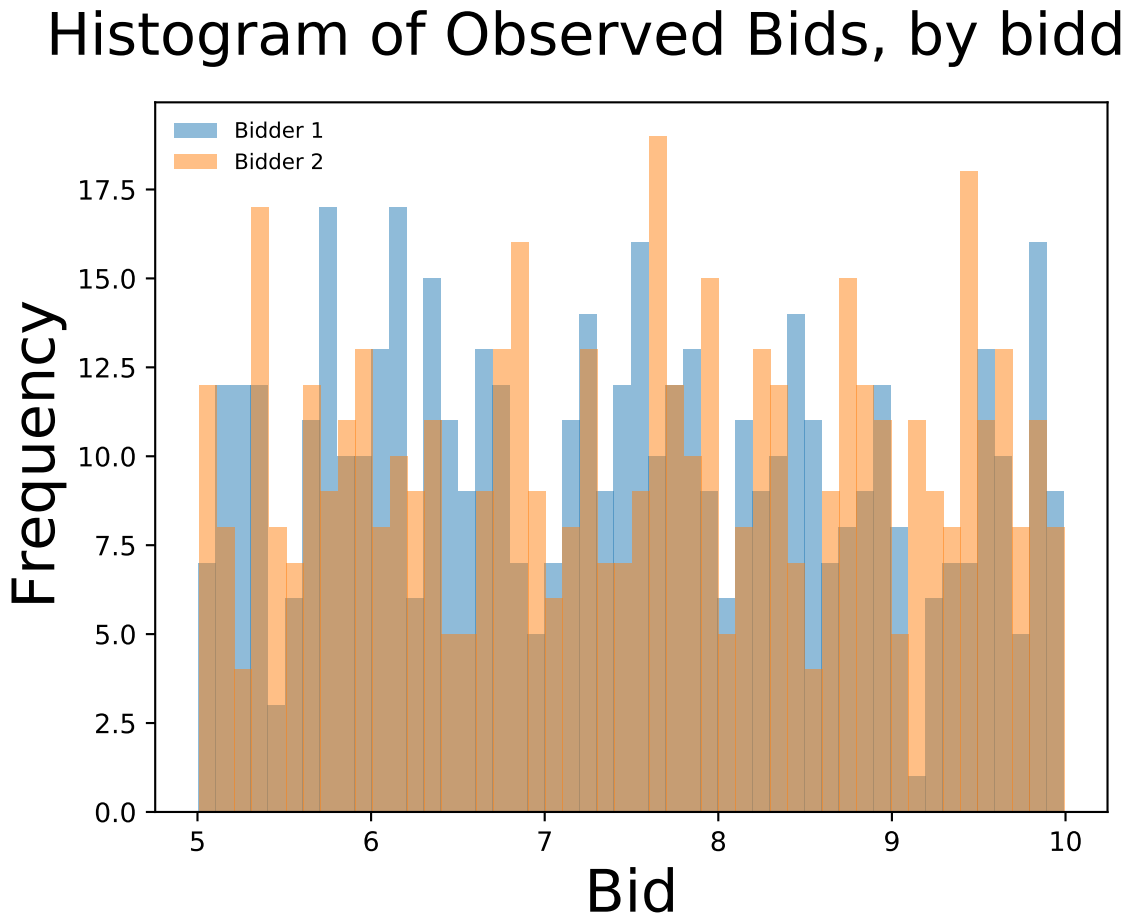

Figure 1: This figure graphs a histogram of each bidder's bids. Each bidder has their own color, and the histograms look pretty much the same. Nothing too crazy here, but reassuring since we're assuming the bidders have independent private values drawn from the same distribution.

3 Estimated Valuations

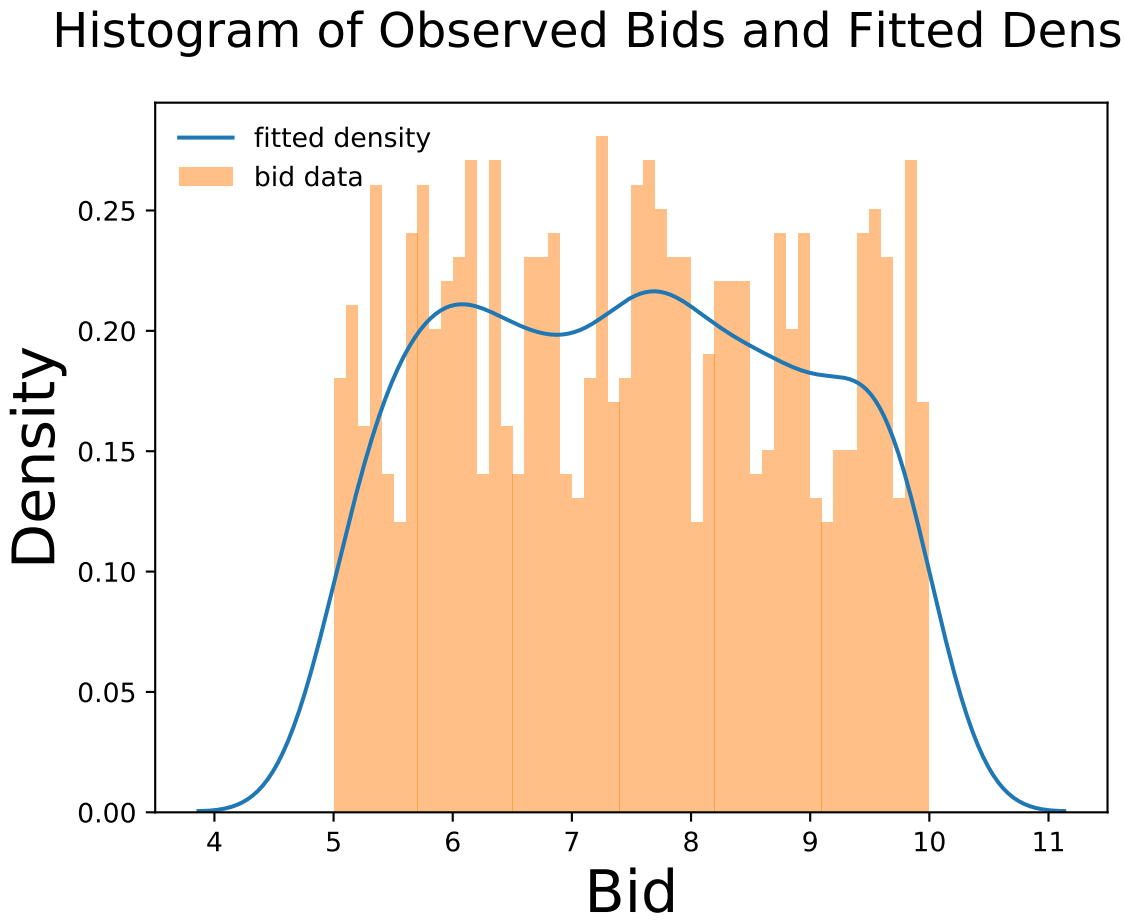

Figure 2: This figure pools the bid data from both bidders and fits a nonparametric density to the data. It is reassuring that the canned kernel density estimator that I'm using seems to do a reasonable job fitting the data.

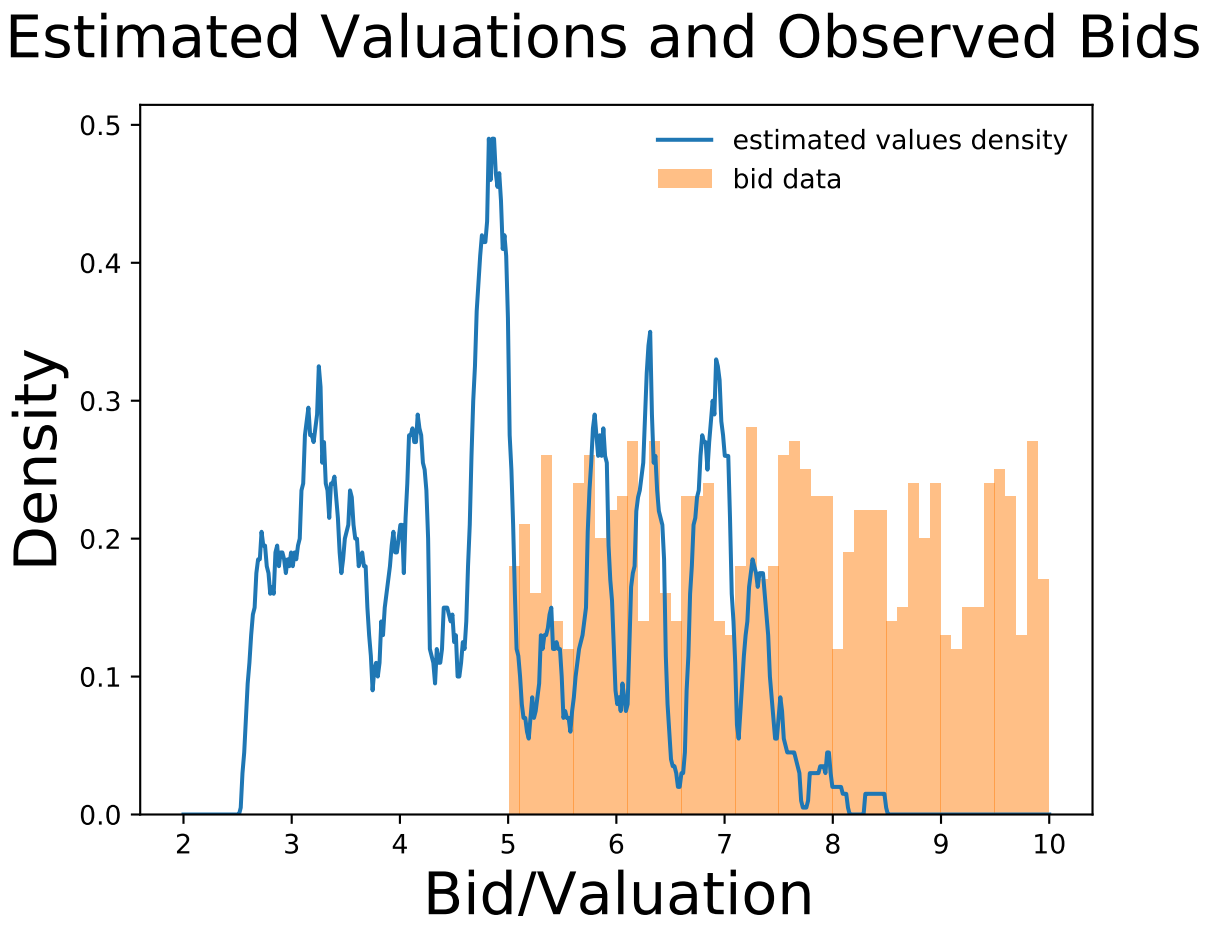

Figure 3: This figure graphs the estimated valuations from the GPV algorithm against the observed bid data (included as a sanity check). It is concerning that the estimated density doesn't reach the top of the support of bids, since that indicates that bidders are sometimes bidding more than their valuation (which should never happen in equilibrium with a first price auction). However, the GPV paper does mention that their algorithm is biased near the edges of the distribution, so that is a possible explanation for the lack of estimated valuations above ~ 8.5.

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.neighbors import KernelDensity


def load_bid_data(filename="PS3Data.csv"):
    """
    Load problem set data
    Downloaded from http://www.johnasker.com/IO.html
    """
    df = pd.read_csv(filename)
    return df


def graph_bidder_data(bid_data=bid_data, filename='bidder_histogram.pdf', n_bins=50):
    """
    Graph a histogram of bids by each bidder;
    Takes bid data and number of bids as arguments
    Currently hard-coded for 2 bidders
    """
    b1 = bid_data["Bidder 1"].values
    b2 = bid_data["Bidder 2"].values
    fig, ax = plt.subplots()
    fig.suptitle('Histogram of Observed Bids, by bidder', fontsize=22)
    ax.hist(b1, bins=n_bins, alpha=0.5, label='Bidder 1')
    ax.hist(b2, bins=n_bins, alpha=0.5, label='Bidder 2')
    ax.legend(frameon=False, loc='upper left', fontsize=8)
    ax.set_xlabel("Bid", fontsize=22)
    ax.set_ylabel("Frequency", fontsize=22)
    plt.savefig(filename)
    plt.close('all')


def graph_bids(bid_data=bid_data, filename='bid_density.pdf', bandwidth=0.5):
    """
    Plot actual bid data for both bidders
    Along with an estimated density of bids
    Using nonparametric density
    """
    bids = bid_data.drop("Auction #", axis=1).values.flatten().reshape(-1, 1)
    fig, ax = plt.subplots()
    X = rng.random_sample((100,1))
    kde = sm.nonparametric.KDEUnivariate(bids)
    kde.fit()
    ax.plot(kde.support, kde.density, label='fitted density')
    fig.suptitle('Histogram of Observed Bids and Fitted Density', fontsize=18)
    ax.hist(bids, bins=50, density=True, alpha=0.5, label='bid data')
    ax.legend(frameon=False, loc="upper left")
    ax.set_xlabel("Bid", fontsize=22)
    ax.set_ylabel("Density", fontsize=22)
    plt.savefig(filename)
```

```
    plt.close('all')


def kernel_smoothing(X, y, K, h, grid):
    """
    Helper function for GPV algorithm
    """
    out = np.empty((0,1))
    dv = np.prod(h)
    for x in grid:
        X_h = np.array(x - X) / np.array(h)
        K_h = 1/dv * K(X_h)
        est = np.mean(y*K_h)
        out = np.vstack((out, est))
    return out


def density_estimation(X, K, h, grid):
    """
    Helper function for GPV algorithm
    """
    n = np.shape(X)[0]
    return kernel_smoothing(X, np.ones(n), K, h, grid)


def generate_uniform_kernel():
    """
    Helper function for GPV algorithm
    """
    return lambda x: np.array(np.array(1/2*(np.abs(x)<=1)).T, ndmin=2).prod(0)


def bids_inversion(b, I, G, g):
    """
    Invert bids to get valuations
    """
    return b+1/(I-1)*G(b)/g(b)


def estimate_values_density(B, kg, hg, rhog, kf, hf, grid):
    """
    Estimate the density of valuations
    From observed bid data
    Return estimated density that valuations
    Are drawn from
    """
    I = np.shape(B)[1]
    L = np.shape(B)[0]
    B = B.ravel()
    G = lambda b: np.array([np.sum(B<=x)/(L*I) for x in b])
    g = lambda b: np.reshape(density_estimation(B, kg, hg, b),(-1,))
    V = bids_inversion(B, I, G, g)
    c = (np.min(B)+rhog*hg/2<=B)*(np.max(B)-rhog*hg/2>=B)
    V = V[c]
```

```
    f = density_estimation(V, kf, hf, grid)*np.size(V)/(I*L)
    return f


def graph_estimated_density(
        density_support,
        estimated_density=estimated_density,
        filename='estimated_density.pdf'
):
    fig, ax = plt.subplots()
    ax.plot(density_support, estimated_density, label='estimated values density')
    bids = bid_data.drop("Auction #", axis=1).values.flatten().reshape(-1, 1)
    ax.hist(bids, bins=50, density=True, alpha=0.5, label='bid data')
    ax.legend(frameon=False)
    ax.set_xlabel(xlabel='Bid/Valuation', fontsize=22)
    ax.set_ylabel(ylabel='Density', fontsize=22)
    fig.suptitle('Estimated Valuations and Observed Bids', fontsize=22)
    plt.savefig(filename)
    plt.close('all')


if __name__ == '__main__':
    my_dir = "/home/chris/files/school/ucla/second_year/fall/io/pset2"
    os.chdir(my_dir)
    bid_data = load_bid_data(filename="PS3Data.csv")
    graph_bidder_data(bid_data=bid_data, filename='bidder_histogram.pdf', n_bins=50)
    graph_bids(bid_data=bid_data, filename='bid_density.pdf', bandwidth=0.5)
    density_support = np.linspace(2, 10, 500)
    estimated_density = estimate_values_density(
                                    B = bid_data[['Bidder 1', 'Bidder 2']].to_numpy()/2,
                                    kg = generate_uniform_kernel(),
                                    rhog = 1,
                                    kf = generate_uniform_kernel(),
                                    hg = 0.1,
                                    hf = 0.1,
                                    grid = density_support
    )
    graph_estimated_density(
                            density_support=density_support,
                            estimated_density=estimated_density,
                            filename='estimated_density.pdf'
    )
```