Nonparametric Quantile Regression for Prediction of Parking Data

Tanner Fiez
Department of Electrical Engineering
University of Washington

I. Introduction

In this paper we will explore the approach taken by Juban, Ohlson, Maasoumy, Poirier, and Kolter in their paper, "A multiple quantile regression approach to the wind, solar, and price tracks of GEFCom2014" [1] in detail as well as discuss general nonparametric quantile regression methods. Moreover, we have implemented the algorithms in [1] and created a sci-kit learn style python package[1] to allow for the methods to be applied to generic data sets. We will present and discuss the results of using our implementation of the approach on parking occupancy data from the city of Seattle, WA.

A. Motivation

Regression algorithms typically aim to estimate the conditional mean of a target variable $f(y|x)$, yet a single estimate can be unsatisfying depending on the task. With only a single prediction at each point there is no information about the uncertainty of an estimate. Confidence intervals are one way that the uncertainty in a regression algorithm that attempts to estimate the conditional mean of a target variable can be quantified. Even this approach is limited however, since typically only use a single confidence interval (usually 95%) is given. An alternative approach to regression comes in the form of quantile regression. Quantile regression attempts to estimate the distribution of $y|x$ at various percentiles in order to give a more complete view of the function. There are many prediction tasks where this is an important characteristic. The two examples that we will consider, energy and parking, are well suited for a more complete model. As outlined in [2] the energy industry has been changing rapidly. An artifact of this is that the supply, demand, and price of energy have become more volatile and thus more difficult to predict. This was the motivation for the nonparametric quantile regression approach taken in [1]. Parking management has become very important in dense urban spaces. The two primary reasons for this are the lack of information about parking availability, and the cost that comes from drivers searching for parking. Unlike ubiquitous travel time estimates from handy smartphone applications which can accurately predict how long a trip will take to get to a desired destination, there is typically no information upon arrival about where there will be parking. Recently, Google has made an attempt at simply estimating the degree of difficulty to find parking in an area within select cities, but not availability at specific locations [3]. As a result of the lack of information, drivers spend time circling looking for parking which leads to various inefficiencies. In [4], it was shown that in Seattle, WA, at certain locations and times, over 10% of vehicles on the road are searching for parking and this can lead to over a 60% delay in travel time. For these reasons, a probabilistic prediction of parking occupancy is favorable since parking trends have high variance and a distribution can more easily allow for city planners to design control policies through dynamic pricing in order to mitigate parking related congestion.

[1]The code is hosted on Github at https://github.com/fiezt/nonparametric-quantile-regression.

II. Quantile Regression

Quantile regression has been explored for many years in the literature, and a description is given in [5]. As in regression techniques that minimize the a sum of squared of residuals, the median can be described as a problem of minimizing the sum of absolute residuals. Because the piecewise absolute value function is symmetric, the minimization of the sum of the absolute residuals ensures the same number of observations above and below the median. To estimate the other quantiles, minimizing the sum of asymmetric weighted absolute residuals yields the quantiles. This equates to the optimization problem

$$\min \sum_{i=1}^{m} \psi(y_i - \hat{y}) \tag{1}$$

where ψ is known as the tilted absolute loss function given by

$$\psi(z) = \max\{\alpha z, (\alpha - 1)z\}, \tag{2}$$

where $\alpha \in (0, 1)$ is the quantile.

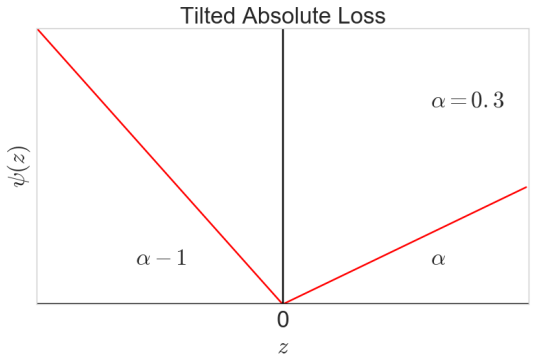

Fig. 1: An example of the tilted absolute loss function with $\alpha = 0.3$.

In a parametric quantile regression formulation, we can replace the estimates y_i by a parametric function ξ and solve the minimization problem

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{m} \psi_\alpha(y_i - \xi(x_i, \beta)) \tag{3}$$

where the notation that we are solving for the parameters of a model with p covariates for the α quantile. Much like in traditional regression models, a regularization term can be added and the optimization problem

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{m} \psi_\alpha(y_i - \xi(x_i, \beta)) + \lambda \|\beta\| \tag{4}$$

can be solved to reduce the potential of overfitting, where λ is the regularization parameter and $\|\cdot\|$ denotes any one of the several options of norms for regularization. This problem can be solved with standard optimization techniques.

An interesting and nonintuitive phenomenon that occurs in quantile regression is what is known as the *quantile crossing* problem. This is that for a set of quantiles, estimated conditional quantile functions can overlap. There has been a large amount of research into this problem, and in [6] constraints are proposed in order to solve this problem. However, a simpler solution that often works well, is to simply sort the quantiles so that they are non-crossing.

III. GEFCom 2014

The Global Energy Competition of 2014 was a probabilistic energy forecasting competition with four tracks including load, price, wind, and solar forecasting. As an error measure for the competition the pinball loss function was chosen. This loss function is

$$l(y, \hat{y}_\mathcal{Q}) = \sum_{\alpha \in \mathcal{Q}} \psi_\alpha(y - \hat{y}_\alpha) \tag{5}$$

where ψ is the titled absolute loss function from (2), and $\mathcal{Q}$ is the set of quantiles given by $\mathcal{Q} = \{0.01, 0.02, \ldots, .99\}$. Thus the error is a sum of the tilted absolute loss at each quantile in the set $\mathcal{Q}$. This loss function is the premise for the nonparametric quantile formulation in [1].

IV. Nonparametric Quantile Regression Formulation

Considering the pinball loss function $l(y, \hat{y}_\mathcal{Q}) = \sum_{\alpha \in \mathcal{Q}} \psi_\alpha(y - \hat{y}_\alpha)$, where $\hat{y}_\alpha$ is some estimate, which we can consider to be some function z, and given a set of target variables $y \in \mathbb{R}^m$, for each quantile we want to minimize $\sum_{i=1}^{m} \psi_\alpha(y^i - z)$. Thus this function is minimized by letting z be the α quantile of y. With a set of covariates and target variables $x \in \mathbb{R}^{m \times p}, y \in \mathbb{R}^m$, we want to minimize the sum of the quantile loss functions for the set as

$$L_\mathcal{Q} = \sum_{i=1}^{m} \sum_{\alpha \in \mathcal{Q}} \psi_\alpha(y^i - \hat{y}^i_\alpha(x^i)). \tag{6}$$

The approach to solve this problem given in [1] involves three stages. First an input selection stage using a forward-stepwise selection procedure, radial basis feature creation, and optimization of the loss function based on techniques from the alternating direction method of multipliers (ADMM) algorithm from [7].

V. Forward Stepwise Selection

In the GEFCom 2014 dataset were many available features, and the authors in [1] also derived many useful features. In order to determine the most useful features, a dimensionality reduction algorithm was used. There are many well known feature reduction algorithms including principal component analysis, linear discriminant analysis, minimum redundancy maximum relevancy, and subset selection to name a few. The authors in [1] chose to use a subset selection method of forward stepwise feature selection [8]. This boosted performance but is not strictly necessary for the nonparametric quantile regression method. In best subset selection, for each $k \in \{0, 1, \ldots, p\}$ the subset of k that minimizes the residual sum of squares is chosen. This procedure is computationally intensive however, and for this reason is often only feasible for a p up to 30 or 40. In forward stepwise selection, instead of considering all possible subsets, the algorithm starts with an intercept term and sequentially adds the covariate that produces the minimum sum of squared residuals using a simple least squares estimate and leave one out cross validation (LOOCV). This procedure produces a sequence of models, that is a set of covariates for each k.

Although it may seem as if LOOCV will be very computationally expensive, in linear smoothers LOOCV can be computed very efficiently. Linear smoothers are models where fitted values can be written in the form $\hat{f} = Sy$. In least squares for example $S = X(X^T X)^{-1} X^T$. The typical way LOOCV error is computed is

$$\frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{f}_{-i}(x_i))^2, \tag{7}$$

but for all linear smoothers an equivalent computation is

$$\frac{1}{m} \sum_{i=1}^{m} \left(\frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}\right)^2. \tag{8}$$

Thus to compute the LOOCV error in a linear smoother only one model needs to be fit instead of m independent models. A proof of this follows for the least squares estimate, and readers can refer to [9] to see how this generalizes to be true for all linear smoothers. In least squares the minimization problem is

$$\min_\beta \sum_{i=1}^{m} (y_i - x_i \beta)^2. \tag{9}$$

Fitting the model to $m - 1$ points contained in y_{-i} gives the following minimization problem

$$\min_\beta \sum_{j=1, j \neq 1}^{m} (y_j - f_{-i}(x_j))^2, \tag{10}$$

which can equivalently be written as

$$\min_\beta \sum_{j=1}^{m} (y_j^\star - f_{-i}(x_j))^2, \tag{11}$$

where

$$y_j^\star = \begin{cases} y_j & \text{if } j \neq i \\ y_i - y_i + f_{-i}(x_i) & \text{if } j = i \end{cases}. \tag{12}$$

The solution of the minimization problem is then given by

$$\beta = (X^T X)^{-1} X^T y^\star. \tag{13}$$

Hence, the estimator is given by $\hat{f} = X(X^T X)^{-1} X^T y^\star$, which can be written as a linear smoother $\hat{f} = Sy^\star$, where $S = X(X^T X)^{-1} X^T$. The LOOCV estimate is then $\hat{f}_{-1}(x_i) = S_i y^\star$, which can equivalently be computed as

$$\begin{aligned} \hat{f}_{-1}(x_i) &= S_i y^\star \\ \hat{f}_{-1}(x_i) &= S_i y - S_{ii} y_i + S_{ii} \hat{f}_{-i}(x_i) \\ \hat{f}_{-1}(x_i) &= \hat{f}(x_i) - S_{ii} y_i + S_{ii} \hat{f}_{-1}(x_i) \\ \hat{f}_{-1}(x_i) &= \frac{\hat{f}(x_i) - S_{ii} y_i}{1 - S_{ii}}. \end{aligned} \tag{14}$$

The error estimate is then

$$y_i - \hat{f}_{-i}(x_i) = \frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}, \tag{15}$$

which plugging back into (7) yields the result for the LOOCV error of

$$\frac{1}{m} \sum_{i=1}^{m} \left(\frac{y_i - \hat{f}(x_i)}{1 - S_{ii}} \right)^2. \tag{16}$$

By the greedy nature of the forward stepwise selection algorithm, it is suboptimal with respect to the training set, but the advantages of it are the computational efficiency compared to the best subset selection method, and slightly higher bias in the model.

Algorithm 1 Forward Stepwise Selection

Input: Covariates $X \in \mathbb{R}^{m \times p}$, targets $y \in \mathbb{R}^m$

Output: $\tilde{X} \in \mathbb{R}^{m \times p}$ with columns ordered by covariate selection order, $J \in \mathbb{R}^p$ the indexes of the original covariates ordered by selection order, $E \in \mathbb{R}^p$ the leave one out cross validation error at each subset selection.

procedure FORWARDSTEPWISESELECTION(X, y)
  Initialize: $\tilde{X} \leftarrow \emptyset, J \leftarrow \emptyset, E \leftarrow \emptyset$
  for $i = 1, \ldots, p$ **do**
    $j^* = \underset{j \notin J}{\text{argmin}}\ \text{LOOCV}(\tilde{X} \cup X_j \sim y)$
    $e^* = \min_{j \notin J} \text{LOOCV}(\tilde{X} \cup X_j \sim y)$
    $J \leftarrow J \cup j^*$
    $E \leftarrow E \cup e^*$
    $\tilde{X} \leftarrow \tilde{X} \cup \tilde{X}_{j^*}$

As a note of implementation, we observed that evaluating the curve of the LOOCV errors against the size of the subset helped make it intuitively clear how large of a subset of $\tilde{X}$ to keep. This value is also tuned via cross validation. From this point forward we will let the subset size kept be q and consider $\tilde{X} \in \mathbb{R}^{m \times q}$.

VI. Radial Basis Feature Creation

Following the selection of covariates to use in the model, the feature space is transformed using radial basis functions (RBF). In particular as in [1] we use squared exponential bias functions. The effect of doing this is to create a nonlinear relation being the feature space and the target variables. Squared exponential bias functions are given by

$$\phi_j(\tilde{x}) = \exp\left(\frac{\|\tilde{x} - \mu_j\|_2^2}{2\sigma_j^2} \right), \ j = 1, \ldots, k \tag{17}$$

where $\tilde{x} \in \mathbb{R}^q, \mu_j \in \mathbb{R}^q, \sigma_j \in \mathbb{R}$, and $\phi(\tilde{x}) \in \mathbb{R}^k$. To choose the parameter value for each μ_j and σ_j the k-means algorithm is used, where the resulting centroids are used as the values for each μ_j respectively. With each μ_j chosen, the bandwidth parameters σ_j are chosen using what is known as the *median trick*, which is to set

$$\sigma_j = \underset{l \neq j}{\text{median}} \|\mu_j - \mu_l\|_2. \tag{18}$$

In addition to using the standard k-means algorithm, the k-means++ initialization introduced in [10] is used. This initialization procedure has been shown to improve results significantly, by a simple seeding technique which attempts to spread out the the initializations of centroids. This is done by initializing the first centroid to a sample of the input data chose uniformly at random. Then the rest of the centroids are chosen by sampling from the data with probability for each sample proportional to the minimum squared distance from the sample to the existing centroids. This helps reduce the chances of finding suboptimal solutions at local minimum. As another technique to reach a better solution we use multiple restarts and choose the solution that minimized the loss function which is given by

$$L = \sum_{j=1}^{k} \sum_{i \in C_k} \|\tilde{x}_i - \mu_j\|_2^2. \tag{19}$$

The number of clusters, which sets the number of basis function that are used, is a parameter that needs to be tuned by cross validation. We can also note that this procedure and method for generating features makes it clear that we are using a nonparametric model since we do not have a class of function we are considering.

Algorithm 2 K-Means++

Input: Covariates $\tilde{X} \in \mathbb{R}^{m \times q}$, k the number of clusters
Output: Centroids $\mu \in \mathbb{R}^{k \times q}$, bandwidths $\sigma \in \mathbb{R}_+^k$

procedure K-MEANS($\tilde{X}, k$)
  Initialize: $\mu_1 \leftarrow x^i, i$ chosen uniformly at random from $i = 1, \ldots, m$
  for $j = 1, \ldots, k$ **do**
    Sample x^i at random with probability $p(x^i) \propto \min_{l<j} \|x^i - \mu_l\|_2^2$
    $\mu_j \leftarrow x^i$
  while (not converged) **do**
    $c_i \leftarrow \underset{j}{\text{argmin}} \|\mu_j - x^i\|_2^2, \ i = 1, \ldots, m$
    $\mu_j \leftarrow \frac{\sum_{i=1}^m \mathbf{1}\{c_i = j\} x^i}{\sum_{i=1}^m \mathbf{1}\{c_i = j\}}, \ j = 1, \ldots, k$
  for $j = 1, \ldots, k$ **do**
    $\sigma_j \leftarrow \underset{l \neq j}{\text{median}} \|\mu_j - \mu_l\|_2$

Note that convergence in k-means is given by no labels changing between iterations. Using the Euclidean distance measure, this convergence is guaranteed.

VII. ADMM Optimization

In this section the ADMM optimization method used by the authors in [1] is explored. Recall from section IV the loss function we want to attempt to minimize was given by

$$L_{\mathcal{Q}} = \sum_{i=1}^{m} \sum_{\alpha \in \mathcal{Q}} \psi_\alpha(y^i - \hat{y}_\alpha^i(x^i)). \tag{20}$$

Now following the first two stages of the machine learning approach we have a feature space given by $\phi \in \mathbb{R}^{m \times k}$. To minimize the loss function, a linear model is used (linear in the features, but nonlinear with respect to the original input), meaning that are predictions are simply a function of the parameters we learn and our feature space. Thus predictions for a quantile are given by $\hat{y}_\alpha(x, \theta_\alpha) = \theta_\alpha^T \phi(x)$, where $\theta \in \mathbb{R}^k$, which demonstrates that each quantile model is independent, and therefore has its own unique set of parameters. Considering this form of linear model as well as adding regularization, the problem to solve becomes

$$\min_{\theta_{\mathcal{Q}}} \sum_{i=1}^{m} \sum_{\alpha \in \mathcal{Q}} \psi_\alpha(\theta_\alpha^T \phi(x^i) - y^i) + \frac{\lambda}{2} \sum_{\alpha \in \mathcal{Q}} \|\theta\|_2^2. \tag{21}$$

Because this is a convex optimization problem it could be solved for each quantile independently using off the shelf solvers. This would require solving 99 separate optimization problems though. For this reason the authors in [1] came up with a novel approach to solving this problem efficiently by formulating the problem in such a way that it could be solved very similarly as the ADMM algorithm. In doing so, all quantiles can be solved for simultaneously and some clever observations about the structure of the problem allow for reuse of computations.

Before discussing the methods used in [1], we will first review the fundamentals of the ADMM algorithm from [7]. ADMM has its foundations from dual ascent and the augmented Lagrangian and the method of multipliers. We can consider a basic equality constrained convex optimization problem of the form

$$\begin{aligned} &\min \ f(x) \\ &\text{subject to } Ax = b \end{aligned} \tag{22}$$

with the associated Lagrangian and dual respectively

$$L(x, y) = f(x) + y^T(Ax - b) \tag{23}$$

$$g(y) = -f^*(-A^T y) - b^T y \tag{24}$$

where y is the Lagrange multiplier. The primal optimal point is $x^* = \underset{x}{\text{argmin}} \ L(x, y^*)$, and the dual problem of max $g(y)$ is solved by using gradient ascent. The dual ascent method then consists of iterating the following updates

$$\begin{aligned} x^{k+1} &= \underset{x}{\text{argmin}} \ L(x, y^k) \\ y^{k+1} &= y^k + \alpha^k (A x^{k+1} - b), \end{aligned} \tag{25}$$

where the gradient $\nabla g(y) = Ax^{k+1} - b$, is the residual of the equality constraint. The most important property of the dual ascent method is that it can be decomposed so that if a function is separable (e.g., $f(x) = \sum_{i=1}^N f_i(x_i)$), the N different problems can be solved in parallel as

$$\begin{aligned} x_i^{k+1} &= \underset{x}{\text{argmin}} \ L_i(x, y^k) \\ y^{k+1} &= y^k + \alpha^k (A x^{k+1} - b). \end{aligned} \tag{26}$$

This is known as the dual decomposition. The augmented Lagrangian was created in part to give convergence guarantees without strong assumptions about convexity or finiteness of a function. The augmented Lagrangian is

$$L_\rho(x, y) = f(x) + y^T(Ax - b) + (\rho/2)\|Ax - b\|_2^2 \tag{27}$$

where $\rho > 0$ is called the penalty parameter. This problem can also be formulated equivalently as an unaugmented Lagrangian associated with the problem

$$\begin{aligned} &\text{minimize } f(x) + \rho/2 \|Ax - b\|_2^2 \\ &\text{subject to } Ax = b \end{aligned} \tag{28}$$

Applying dual ascent to this problem yields the method of multipliers algorithm which has updates

$$\begin{aligned} x^{k+1} &= \underset{x}{\text{argmin}} \ L_\rho(x, y^k) \\ y^{k+1} &= y^k + \rho(A x^{k+1} - b). \end{aligned} \tag{29}$$

While the method of multipliers has better convergence properties than dual ascent, when f is separable, L_ρ is not separable, so the computations cannot be decomposed to be computed in parallel. ADMM blends the convergence properties of the method of multipliers with the decomposition capabilities of dual ascent. It is for problems of the form

$$\begin{aligned} &\min \ f(x) + g(x) \\ &\text{subject to } Ax + Bz = c \end{aligned} \tag{30}$$

The difference in the constraint here is that the variable x in the constraint has been split into x and z with the objective function also separated across the split. The augmented Lagrangian for this problem is

$$L_\rho(x,z,y) = f(x)+g(z)+y^T(Ax+Bz-c)+(\rho/2)\|Ax+Bz-c\|_2^2 \tag{31}$$

which gives the ADMM iterations of

$$\begin{aligned} x^{k+1} &= \underset{x}{\text{argmin}}\ L_\rho(x, z^k, y^k) \\ z^{k+1} &= \underset{z}{\text{argmin}}\ L_\rho(x^{k+1}, z, y^k) \\ y^{k+1} &= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \end{aligned} \tag{32}$$

Separating the minimization of x and z allows the decomposition when both f and g are separable. An alternative form of ADMM is to used a scaled form by letting $r = Ax + Bz - c$ and a scaled dual variable be $u = (1/\rho)y$ giving the updates of

$$\begin{aligned} x^{k+1} &= \underset{x}{\text{argmin}}\ (f(x) + (\rho/2)\|Ax + Bz^k - c + u^k\|_2^2) \\ z^{k+1} &= \underset{z}{\text{argmin}}\ (g(z) + (\rho/2)\|Ax^{k+1} + Bz - c + u^k\|_2^2) \\ u^{k+1} &= u^k + Ax^{k+1} + Bz^{k+1} - c. \end{aligned} \tag{33}$$

Now considering the optimization problem for a single quantile α which is of the form

$$\min_{\theta\in\mathbb{R}^k}\ \sum_{i=1}^m \psi_\alpha(\theta^T\phi(x^i) - y^i) + \frac{\lambda}{2}\|\theta\|_2^2 \tag{34}$$

This problem can be reformulated equivalently to a form that is similar to (28).

$$\begin{aligned} &\min_{\theta\in\mathbb{R}^k, z\in\mathbb{R}^m}\ \psi_\alpha(z) + \frac{\lambda}{2}\|\theta\|_2^2 \\ &\text{subject to } z = \phi\theta - y \end{aligned} \tag{35}$$

The associated augmented Lagrangian for the problem is then

$$L_\rho(\theta,z,w) = \psi_\alpha(z)+\frac{\lambda}{2}\|\theta\|_2^2+w^T(\phi\theta-y-z)+\frac{\rho}{2}\|\phi\theta-y-z\|_2^2 \tag{36}$$

where here w is the dual variable for the constraint on z. As before this allows for the ADMM iteration, which we consider in the scaled form from (33), with a new scaled dual variable $u = \frac{1}{\rho}w$ to give the iteration

$$\begin{aligned} \theta^{k+1} &= \underset{x}{\text{argmin}}\ \frac{\lambda}{2}\|\theta\|_2^2 + (\rho/2)\|\phi\theta - y - z^k + u^k\|_2^2 \\ z^{k+1} &= \underset{z}{\text{argmin}}\ \psi_\alpha(z) + (\rho/2)\|\phi\theta^{k+1} - y - z + u^k\|_2^2 \\ u^{k+1} &= u^k + \phi\theta^{k+1} - y - z^{k+1}. \end{aligned} \tag{37}$$

In this form, both the updates for θ^{k+1} and z^{k+1} can be solved in closed form. The update for θ^{k+1} is a regularized least squares problem that has the solution

$$\theta^{k+1} = (\phi^T\phi + \frac{\lambda}{\rho}I)^{-1}\phi^T(y + z^k - u^k), \tag{38}$$

where I is the identity matrix $\in \mathbb{R}^{k\times k}$. The update for z^{k+1} has the form of a proximal operator, which is of the form $\underset{x}{\text{argmin}}\frac{1}{2}\|x - y\|_2^2 + f(x)$. To solve this the authors in [1] consider a simpler problem of

$$\underset{z}{\text{argmin}}\frac{1}{2}(z - y)^2 + \gamma\psi_\alpha(z) \tag{39}$$

which has the solution

$$S_{\gamma,\alpha}(y) \equiv \max\{0, y - \gamma\alpha\} + \min\{0, y - \gamma(\alpha - 1)\}. \tag{40}$$

A proof of this is given in [1]. With the closed form solutions to the updates the ADMM iteration becomes

$$\begin{aligned} \theta^{k+1} &= (\phi^T\phi + \frac{\lambda}{\rho}I)^{-1}\phi^T(y + z^k - u^k), \\ z^{k+1} &= S_{1/\rho,\alpha}(\phi^T\theta^{k+1} - y + u^k) \\ u^{k+1} &= u^k + \phi\theta^{k+1} - y - z^{k+1}. \end{aligned} \tag{41}$$

An important observation from [1] is that computations can be reused across iterations and quantiles. Note that the most computationally expensive task of inverting the matrix $(\phi^T\phi + \frac{\lambda}{\rho}I)$ only needs to be performed once since it is static over all iterations and quantiles. Furthermore, instead of a naive matrix inversion, the Cholesky factorization can be used in the single computation. Finally, all quantiles can be solved at once. This is done by stacking the quantiles columnwise, to form $\Theta \in \mathbb{R}^{k\times|\mathcal{Q}|}, Z \in \mathbb{R}^{m\times|\mathcal{Q}|}, U \in \mathbb{R}^{m\times|\mathcal{Q}|}$. To illustrate this consider Θ as

$$\Theta = \begin{bmatrix} | & | & & | \\ \theta_{\alpha_1} & \theta_{\alpha_2} & \cdots & \theta_{\alpha_{|\mathcal{Q}|}} \\ | & | & & | \end{bmatrix}. \tag{42}$$

Then the computations of all quantiles at each iteration are given by the iteration

$$\begin{aligned} \Theta^{k+1} &= (\phi^T\phi + \frac{\lambda}{\rho}I)^{-1}\phi^T(y1^T + Z^k - U^k), \\ Z^{k+1} &= S_{1/\rho,\mathcal{Q}}(\phi^T\Theta^{k+1} - y1^T + U^k) \\ U^{k+1} &= U^k + \phi\Theta^{k+1} - y1^T - Z^{k+1}. \end{aligned} \tag{43}$$

In the update for U, the operator $S_{1/\rho,\alpha}$ does need to applied individually for each quantile column for $\alpha \in \mathcal{Q}$. The vector 1 is $\in \mathbb{R}^{|\mathcal{Q}|}$. In our implementation of the algorithm we follow as the authors of [1] do and use $\rho = 1$ and instead of using one of the various stopping criteria for ADMM, use a fixed number of iterations near 100.

Algorithm 3 ADMM for Multiple Quantile Regression

Input: Covariates $\phi \in \mathbb{R}^{m \times k}$, target variables $y \in \mathbb{R}^m$, regularization parameter λ, set of quantiles $\mathcal{Q}$, step size ρ, max number of iterations T

Output: Learned parameters $\Theta \in \mathbb{R}^{k \times \mathcal{Q}}$

procedure ADMM($\phi, y, \lambda, \mathcal{Q}, \rho, T$)
  Initialize: $\Theta^0 \leftarrow 0, U^0 \leftarrow 0, Z^0 \leftarrow 0$
  $LL^T = (\phi^T\phi + \frac{\lambda}{\rho} I)$
  for $k = 1, \ldots, T$ **do**
    $\Theta^{k+1} \leftarrow L^{-T}L^{-1}\phi^T(y1^T + Z^k - U^k)$
    $Z^{k+1} \leftarrow S_{1/\rho,\mathcal{Q}}(\phi^T\Theta^{k+1} - y1^T + U^k)$
    $U^{k+1} \leftarrow U^k + \phi\Theta^{k+1} - y1^T - Z^{k+1}$

This problem does potentially have the quantile crossing problem which is handled by sorting the quantiles so that they do not cross. Predictions for the full data set over all quantiles can easily be computed as $\hat{y} = \phi\Theta$, where $\hat{y} \in \mathbb{R}^{m \times |\mathcal{Q}|}$. It is important to note that for the feature space ϕ the parameters that were learned for the feature selection as well as RBF parameters in the training set are used directly for a new test set of covariates. We also use cross validation in choosing the regularization parameter and the number of iterations in ADMM.

VIII. Implementation Details

A major objective of this project was to create a versatile python package to be able to apply this three stage model to any general data set. Another goal was to be able to create a software package that encapsulated many of the details of the algorithm. Thus cross validation of hyper-parameters is done internally using a grid search over the four parameters being tuned which are the number of features to select in the forward stepwise selection algorithm, the number of basis functions to use for the RBF feature creation, the regularization parameter in ADMM, and the number of iterations to run ADMM. Five-fold cross validation is used over the grid search of these parameters. In [2] for the GEFCom a scoring function was used as follows

$$L(q_\alpha, y) = \begin{cases} (1 - \frac{\alpha}{100})(q_\alpha - y), & y < q_\alpha \\ \frac{\alpha}{100}(y - q_\alpha), & y \geq q_\alpha \end{cases} \tag{44}$$

where q_α is a quantile forecast and this is consider over $\mathcal{Q}$ previously, and all predictions for all quantiles are averaged over. A smaller score indicates a better score. The best scoring parameters with the cross validation are then what are used to train the model, in order to make predictions on the test set. To demonstrate the ease in which the package allows for this model to be used, a snippet of code is included below.

```
import algorithms

nonpar = algorithms.NonparametricRegression()

nonpar.fit(x_train, y_train)
nonpar.predict(x_test, y_test)

nonpar.plot_results()
```

These few lines with a data set train the full model including finding a good set of hyper-parameters, makes predictions on a test set, and plots the results.

IX. Application

A. Parking Data

All parking transactions for on street parking are available from the Seattle Department of Transportation (SDOT) from 2011 to present. This data set can be found on the Seattle Open Data site. This data set contains millions of transactions. For this example, we have decided to focus on just a few on street blockfaces in the Belltown neighborhood in Seattle, WA. For each of the blockfaces we examine we have processed the transactions to form a data set from April, 2015 - October, 2015 that consists of the occupancy load at 15 minute intervals. The load is defined as the number vehicles parked at a given time on the block divided by the total capacity of the block, which is also a piece of data available from the Seattle Open Data site. In this regression task we aim to use morning occupancy data from the opening of paid parking at 8am until 1pm to predict parking at 2pm. One issue with this data set is there is not a lot of information at this time that is relevant to prediction with the exception of the occupancy. Since the authors in [1] showed that the algorithm discussed in this paper, showed good generalization properties, we wanted to explore its effectiveness on a data set with a less rich data set without adding exogenous features.

The results ended up being surprisingly good. Below is an example of the results of training the model on over four months of data, where again our set of covariates included the load at every 15 minutes up until 1pm, and our target variables are the load at 2pm, and predicting on a month of data.

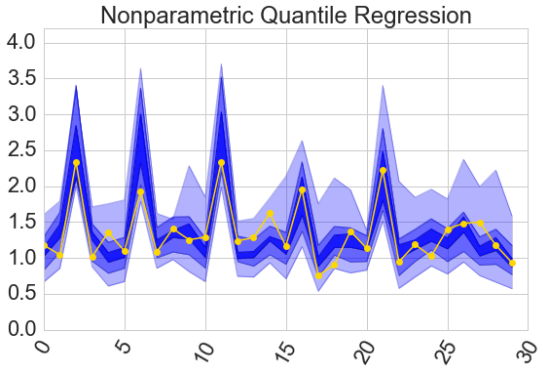

Fig. 2: Predictions on 1 continuous month of parking data on a single block in the Belltown neighborhood of Seattle, WA. The gold line is the true occupancy at each day and the the different shading indicates a range of quantiles. The lightest shade of blue includes the range of quantiles from the 1^{st} to 20^{th} and the 80^{th} to 100^{th} quantiles, the second lightest shading includes the range of quantiles from the 20^{st} to 40^{th} and the 60^{th} to 80^{th} quantiles, and the darkest shade includes the range of quantiles 40^{st} to 60^{th} quantiles.

We can see in the figure that the true occupancy often lies within the 40^{st} to 60^{th} quantiles indicating a good prediction.

Additionally, we see that the highest uncertainty tends to be on Saturdays (the big spikes), which makes intuitive sense since various events and activities can drive up demand. This result had a score of .18 using the scoring function from the previous section which is very good. It is important to note that the occupancy goes over 1 frequently since this is not the true occupancy, but rather the occupancy based off transactions. There are several unobserved variables, such as drivers leaving before their time is up, more cars being able to fit than the designated capacity, etc., which cause this.

X. Discussion

In this paper we thoroughly discussed quantile regression, and in particular the nonparametric quantile regression model used by the authors in [1] which had a three layer approach and used some clever methods in order to make the problem amenable to the popular ADMM optimization procedure. We also implemented the algorithms described, and were able to discuss and show our results in prediction of parking occupancy in Seattle, WA. Results in the prediction seem to be very respectable which backs up the authors claim in [1] that one of the main advantages of this algorithm seemed to be how well it generalized. Future work will involve looking at attempting to improve the feature representation that goes into the ADMM model.

References

[1] R. Juban, H. Ohlsson, M. Maasoumy, L. Poirier, and J. Z. Kolter, "A multiple quantile regression approach to the wind, solar, and price tracks of GEFCom2014," *International Journal of Forecasting*, vol. 32, no. 3, pp. 1094–1102, 2016.

[2] T. Hong, P. Pinson, S. Fan, H. Zareipour, A. Troccoli, and R. J. Hyndman, "Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond," *International Journal of Forecasting*, vol. 32, no. 3, pp. 896–913, 2016.

[3] J. Cook, Y. Li, and R. Kumar, "Using machine learning to predict parking difficulty," https://research.googleblog.com/2017/02/using-machine-learning-to-predict.html, 2017.

[4] C. Dowling, T. Fiez, L. J. Ratliff, and B. Zhang, "How much urban traffic is searching for parking?" *CoRR*, vol. abs/1702.06156, 2017.

[5] R. Koenker and K. F. Hallock, "Quantile regression," *Journal of Economic Perspectives*, vol. 15, no. 4, pp. 143–156, December 2001.

[6] I. Takeuchi, Q. V. Le, T. D. Sears, and A. J. Smola, "Nonparametric quantile estimation," *Journal of Machine Learning Research*, vol. 7, pp. 1231–1264, 2006.

[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[8] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. Springer New York Inc., 2001.

[9] J. Wakefield, *Bayesian and Frequentist Regression Methods*, ser. Springer Series in Statistics. Springer, 2013.

[10] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07, 2007, pp. 1027–1035.