Heuristic Prototype Selection for Regression

Debraj Basu, Deepak Pai, Joshua Sweetkind-Singer

Motivation

***k*-NN regression** faces
- High storage costs
- Expensive computation
- Sensitivity to outliers

A smaller set of **prototypes** can afford
- Lower storage and compute costs
- Robustness to outliers
- Better generalization

Description: Methods which leverage the separation of samples based on class have verified the above benefits for classification

This work generalizes the set-cover based approach by *Bien et. al. 2012* to identify representative points in the regression setting as prototypes

Benchmark: Decremental Instance Selection for *K*-NN Regression (DISKR) *Song et. al. 2017*

Properties

- Prototypes are representative points that exhibit certain desirable properties
- The neighborhood of a prototype is determined by an ϵ covering ball around
- These covering balls are expected to
  - Cover many points with similar labels
  - Avoid covering points with dissimilar labels
- Proximity in label space is defined by a customizable metric
- These properties are governed by the size of the covering balls, and automatically induce sparsity in the data

Formulation

For n samples, the properties are encoded into a mixed-integer program with indicator variables α_i for each sample, and slack variables η_i and ξ_i where $i \in \{1, \ldots, n\}$

$$\min_{\alpha_i, \xi_i, \eta_i} \sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \eta_i + \lambda \sum_{i=1}^{n} \alpha_i$$

$$\sum_{j:\mathbf{x}_i \in B(\mathbf{x}_j)} \alpha_j \left(1 - \frac{\|\mathbf{y}_i - \mathbf{y}_j\|_2}{\Delta \mathbf{y}_{\max}}\right) \geq 1 - \xi_i \qquad \forall i \in [n]$$

$$\sum_{j:\mathbf{x}_i \in B(\mathbf{x}_j)} \alpha_j \frac{\|\mathbf{y}_i - \mathbf{y}_j\|_2}{\Delta \mathbf{y}_{\max}} \leq \eta_i \qquad \forall i \in [n]$$

$$\alpha_i \in \{0, 1\}, \quad \xi_i, \eta_i \geq 0 \qquad \forall i \in [n]$$

Heuristic Prototype Selection

The set of prototypes is successively refined by greedily selecting data points based on the objective

It terminates when the objective cannot be improved anymore

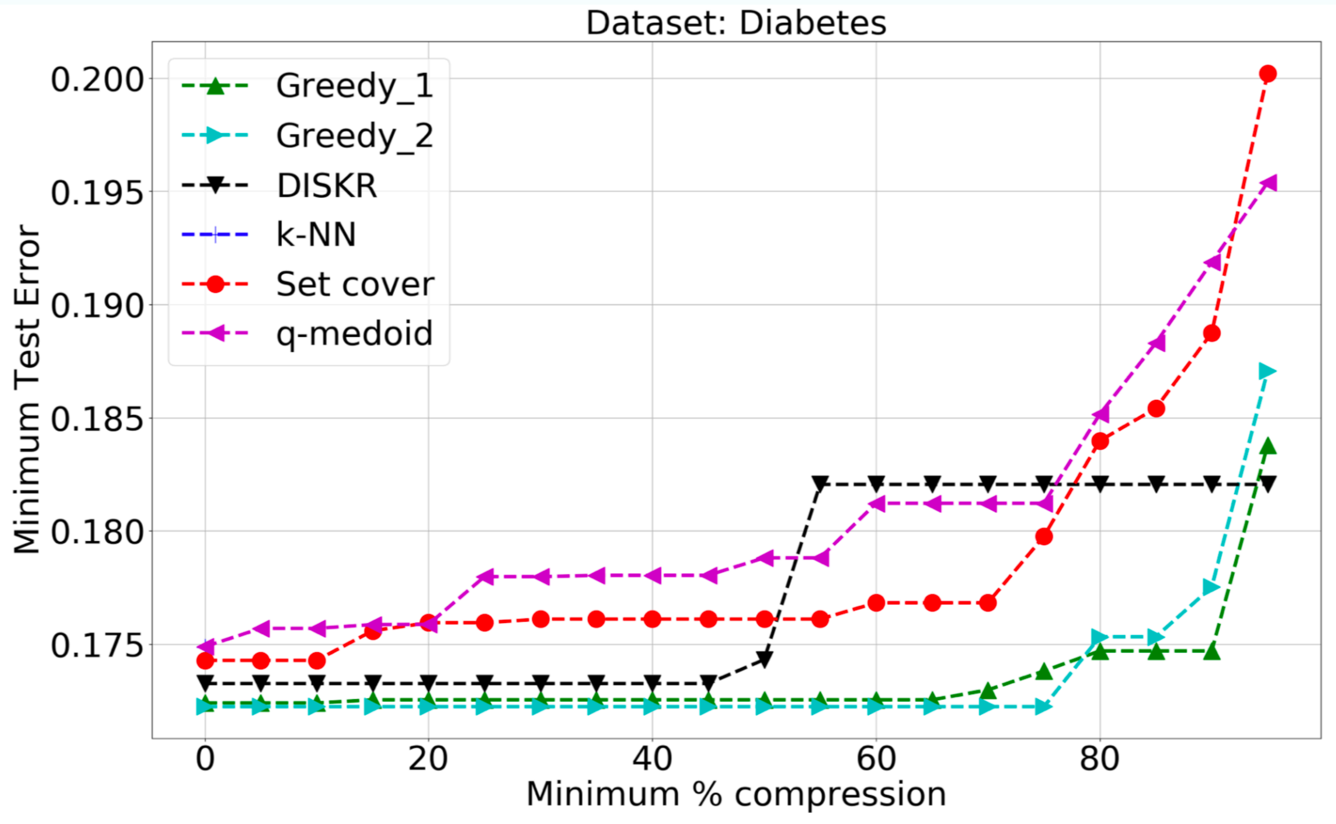
1: Initialize $\mathcal{P} = \phi$
2: **do**
3: Find $i' \leftarrow \arg\max_{i \in [n] \setminus \mathcal{P}} \Delta \mathrm{Obj}(\mathbf{x}_i)$
4: **If** $\Delta \mathrm{Obj}(\mathbf{x}_{i'}) > 0$ **then do** $\mathcal{P} \leftarrow \mathcal{P} \cup \{ i' \}$
5: **while** $\Delta \mathrm{Obj}(\mathbf{x}_{i'}) > 0$
6: **return** $\mathcal{P}$

Comments

- One hot encoding reduces the formulation and solution to multi-class classification (*Bien et. al. 2012*)
- Time Complexity $O(n\, P\, max_i C(i))$; P is the number of prototypes and $C(i)$ is the number of neighbors of x_i
- Varying λ enables data set condensation by controlling the number of prototypes

Experiments

Prediction Performance



For each minimum allowable % compression, our greedy solution exhibits the best prediction performance.

This plot summarizes results from extensive cross validation done across a range of k nearest neighbors, ϵ covering balls

Tolerance to Outliers

Prototypes are selected from increasingly corrupted data sets

Prototypes selected by our method exhibit more robust performance

Methods which utilize the labels are observed to be more tolerant to outliers