Batch-mode Supervised Learning

Nearest neighbor and kernel-based methods
   Properties of the NN method
   Refinements of the NN method

Relation between tree-based and kernel-based methods

Relation between kernel-based and linear methods
Batch-mode Supervised Learning

- Objects (or observations): \( LS = \{o_1, \ldots, o_N\} \)
- Attribute vector: \( a^i = (a_1(o_i), \ldots, a_n(o_i))^T, \quad \forall i = 1, \ldots, N. \)
- Outputs: \( y^i = y(o_i) \) or \( c^i = c(o_i), \quad \forall i = 1, \ldots, N. \)
- LS Table

<table>
<thead>
<tr>
<th>( o )</th>
<th>( a_1(o) )</th>
<th>( a_2(o) )</th>
<th>( \ldots )</th>
<th>( a_n(o) )</th>
<th>( y(o) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( a_1^1 )</td>
<td>( a_2^1 )</td>
<td>( \ldots )</td>
<td>( a_n^1 )</td>
<td>( y^1 )</td>
</tr>
<tr>
<td>2</td>
<td>( a_1^2 )</td>
<td>( a_2^2 )</td>
<td>( \ldots )</td>
<td>( a_n^2 )</td>
<td>( y^2 )</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>( N )</td>
<td>( a_1^N )</td>
<td>( a_2^N )</td>
<td>( \ldots )</td>
<td>( a_n^N )</td>
<td>( y^N )</td>
</tr>
</tbody>
</table>
Nearest neighbor methods

Intuition: similar objects should have similar output values.

- NB: all inputs are numerical scalars
- Define distance measure in the input space:

\[
d_a(o, o') = (a(o) - a(o'))^T (a(o) - a(o')) = \sum_{i=1}^{n} (a_i(o) - a_i(o'))^2
\]

- Nearest neighbor:

\[
NN_a(o, LS) = \arg \min_{o' \in LS} d_a(o, o')
\]

- Extrapolate output from nearest neighbor:

\[
\hat{y}_{NN}(o) = y(NN_a(o, LS))
\]
Batch-mode Supervised Learning
Nearest neighbor and kernel-based methods
Relation between tree-based and kernel-based methods
Relation between kernel-based and linear methods

Properties of the NN method
Refinements of the NN method

Qu (Mvars) vs. Pu (MW) plot showing 3000 learning states. Zoom around state 4984 and nearest neighbor (state 2276) at Qu = -20 Mvar and Pu = 1090 MW.
Properties of the NN method

Computational

- Training: storage of the LS \((n \times N)\)
- Testing: \(N\) distance computations \(\Rightarrow N \times n\) computations

Accuracy

- Asymptotically \((N \rightarrow \infty)\): suboptimal (except if problem is deterministic)
- Strong dependence on choice of attributes \(\Rightarrow\) weighting of attributes

\[
d^w_a(o, o') = \sum_{i=1}^{n} w_i (a_i(o) - a_i(o'))^2
\]

or attribute selection...
Refinements of the NN method

1. The $k$-NN method:

- Instead of using only the nearest neighbor, one uses the $k$ (a number to be determined) nearest neighbors:

  $$kNN_a(o, LS) = \text{First}(k, \text{Sort}(LS, d_a(o, \cdot)))$$

- Extrapolate from $k$ nearest neighbors, e.g. for regression

  $$\hat{y}_{kNN}(o) = k^{-1} \sum_{o' \in kNN_a(o, LS)} y(o')$$

  and majority class for classification.

- $k$ allows to control overfitting (like pruning of trees).

- Asymptotically ($N \to \infty$): $k(N) \to \infty$ and $\frac{k(N)}{N} \to 0 \Rightarrow$ optimal method (minimum error)
Refinements of the NN method

2. Condensing and editing of the $LS$:
   - Condensing: remove ‘useless’ objects $LS$
   - Editing: remove ‘outliers’ from $LS$
   - Apply first editing then condensing (see notes)

3. Automatic tuning of the weight vector $w$...

4. Parzen windows and/or kernel methods:

   $$\hat{y}_K(o) = \sum_{o' \in LS} y(o') K(o, o')$$

   where $K(o, o')$ is a measure of similarity
Nearest neighbor, editing and condensing

Initial LS

Edited LS

Condensed LS
Relation between tree-based and kernel-based methods

Kernel defined by a regression tree:

- Let $\mathcal{L}_i, i = 1, \ldots, |\mathcal{T}|$ denote the leaves of $\mathcal{T}$.
- Let $N_i$ denote the number of objects in the sub-LS of $\mathcal{L}_i$.
- Let $K_{\mathcal{T}}(o, o')$ be equal to $N_i^{-1}$ if $o$ and $o'$ reach same leaf $\mathcal{L}_i$, and 0 otherwise.
- Then the approximation of the regression tree may be written as

$$
\hat{y}_{\mathcal{T}}(o) = \sum_{o' \in LS} y(o') K_{\mathcal{T}}(o, o').
$$
Scalar product representation of tree kernels

Kernel defined by a regression tree:

- Let $L_i, i = 1, \ldots, |T|$ denote the leaves of $T$.
- Let $N_i$ denote the number of objects in the sub-LS of $L_i$.
- For each leaf, define a function attribute $a_{L_i}(o)$ by $a_{L_i}(o) = N_i^{-1/2}$ if $o$ reaches $L_i$, and zero otherwise.
- Let $a_T(o) = (a_{L_1}(o), \ldots, a_{L_{|T|}}(o))^T$
- Then we have that

$$K_T(o, o') = a_T(o)^T a_T(o')$$

and

$$\hat{y}_T(o) = \sum_{o' \in LS} y(o') a_T(o)^T a_T(o').$$
Relation between kernel-based and linear methods

Let us consider a two-class classification problem, and define $y(o) = 1$ if $c(o) = c_1$ and $y(o) = -1$ if $c(o) = c_2$.

Let us construct a simple classifier:

- Center of class 1: $c_+ = N_+^{-1} \sum_{o' \in LS_+} a(o')$
- Center of class 2: $c_- = N_-^{-1} \sum_{o' \in LS_-} a(o')$
- Classifier: $\hat{y}(o) = 1$ if $d(c_+, a(o)) < d(c_-, a(o))$.
- Define $c = \frac{c_+ + c_-}{2}$ and $\Delta c = c_+ - c_-$
- With these notations we have $\hat{y}(o) = sgn((a(o) - c)^T \Delta c)$
- In other words:

$$\hat{y}(o) = sgn \left( N_+^{-1} \sum_{o' \in LS_+} a^T(o')a(o) - N_-^{-1} \sum_{o' \in LS_-} a^T(o')a(o) + b \right)$$

where $b = \frac{1}{2}(\|c_-\|^2 - \|c_+\|^2)$