# Applied inductive learning - Lecture 4

Louis Wehenkel

Department of Electrical Engineering and Computer Science
University of Liège

Montefiore - Liège - October 10, 2005

*Find slides: http://montefiore.ulg.ac.be/∼lwh/AIA/*

# Batch-mode Supervised Learning (Notations)

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

| $o$ | $a_1(o)$ | $a_2(o)$ | $\ldots$ | $a_n(o)$ | $y(o)$ |
|---|---|---|---|---|---|
| 1 | $a_1^1$ | $a_2^1$ | $\ldots$ | $a_n^1$ | $y^1$ |
| 2 | $a_1^2$ | $a_2^2$ | $\ldots$ | $a_n^2$ | $y^2$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $N$ | $a_1^N$ | $a_2^N$ | $\ldots$ | $a_n^N$ | $y^N$ |

# Nearest neighbor methods

Intuition: similar objects should have similar output values.

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

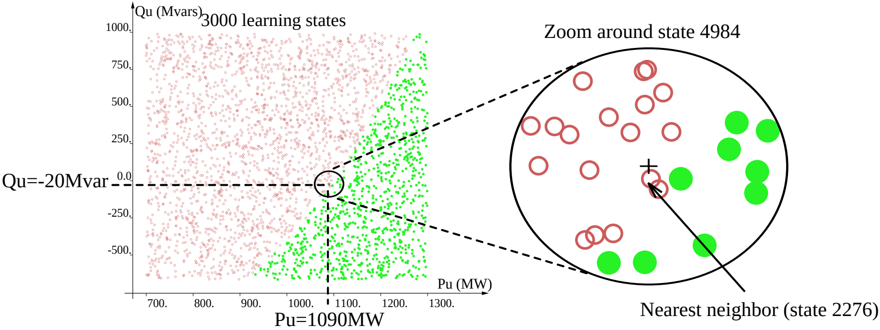
$$d_a(o, o') = (\mathbf{a}(o) - \mathbf{a}(o'))^T (\mathbf{a}(o) - \mathbf{a}(o')) = \sum_{i=1}^{n} (a_i(o) - a_i(o'))^2$$

- Nearest neighbor:

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

- Extrapolate output from nearest neighbor:

$$\hat{y}_{NN}(o) = y(NN_a(o, LS))$$


Qu (Mvars)
3000 learning states
1000.
750.
500.
250.
0.0
-250.
-500.
Qu=-20Mvar
Pu (MW)
700.
800.
900.
1000.
1100.
1200.
1300.
Pu=1090MW
Zoom around state 4984
Nearest neighbor (state 2276)

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

$$d_a^w(o, o') = \sum_{i=1}^{n} w_i (a_i(o) - a_i(o'))^2$$

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:

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

- 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)

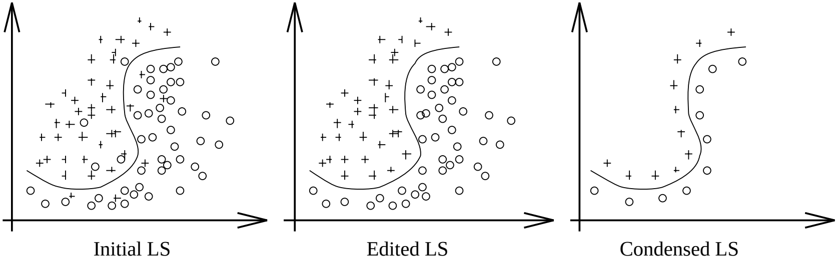
# Refinements of the NN method

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



# 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 $\mathcal{L}_i, i = 1, \ldots, |\mathcal{T}|$ denote the leaves of $\mathcal{T}$.
- Let $N_i$ denote the number of objects in the sub-LS of $\mathcal{L}_i$.
- For each leaf, define a function attribute $a_{\mathcal{L}_i}(o)$ by $a_{\mathcal{L}_i}(o) = N_i^{-1/2}$ if $o$ reaches $\mathcal{L}_i$, and zero otherwise.
- Let $\mathbf{a}_{\mathcal{T}}(o) = (a_{\mathcal{L}_1}(o), \ldots, a_{\mathcal{L}_{|\mathcal{T}|}}(o))^T$
- Then we have that

$$K_{\mathcal{T}}(o, o') = \mathbf{a}_{\mathcal{T}}^T(o)\mathbf{a}_{\mathcal{T}}(o')$$

- and

$$\hat{y}_{\mathcal{T}}(o) = \sum_{o' \in LS} y(o')\mathbf{a}_{\mathcal{T}}^T(o)\mathbf{a}_{\mathcal{T}}(o').$$

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

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

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