Harvard-MIT Division of Health Sciences and Technology
HST.951J: Medical Decision Support, Fall 2005
Instructors: Professor Lucila Ohno-Machado and Professor Staal Vinterbo

Putting it together: Learning Fuzzy Rules

Staal A. Vinterbo

Harvard-MIT Division of Health Science and Technology

Decision Systems Group, BWH

Harvard Medical School

Nov 9 2005: HST 951/MIT 6.873 Class

Motivation

- Put our knowledge into action
- Show a practical classifier development cycle

Fuzzy Rule Classifier

Background: What is a rule?

Fuzzy rule: $(\alpha \rightarrow \beta)$

- α is a conjunction over propositions over a fuzzy sets, i.e, contains only $\wedge$ and fuzzy set membership function names.
- β is a proposition over a fuzzy set.

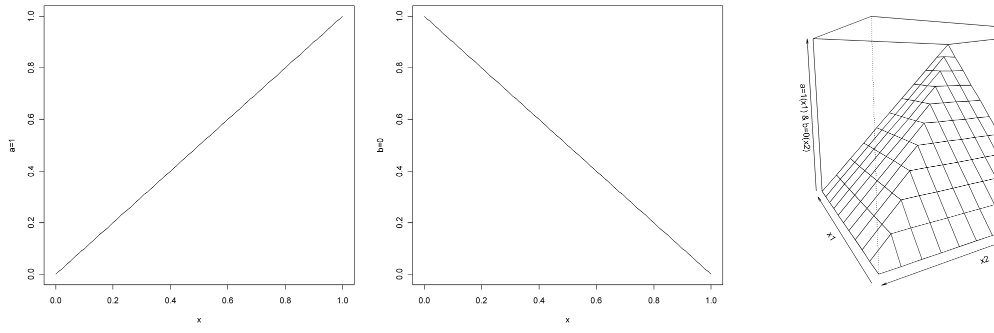
The result of application of $(\alpha \rightarrow \beta)$ is $I(\beta, x) = I(\alpha, x)$.

Fuzzy Rule Classifier

Background: Rule Example

Example

$a = 1 \wedge b = 0 \Rightarrow \text{XOR} = 1$ applied to $x \in [0,1]^2$ results in $I(\text{XOR} = 1, (x_1, x_2)) = \min(\mu_{T(a=1)}(x_1), \mu_{T(b=0)}(x_2))$.



Fuzzy Rule Classifier
Background: What is it?

A fuzzy rule classifier is

- a set of rules
- a way to apply these rules to a data point

Let $(\alpha_1 \to \beta_1)$ and $(\alpha_2 \to \beta_2)$ be two rules. Applying these to a point *x* yields memberships for β_1 and β_2. If $\beta = \beta_1 = \beta_2$, we assign the maximal of the two memberships to β, justified by

$$(\alpha_1 \to \beta) \wedge (\alpha_2 \to \beta) \leftrightarrow ((\alpha_1 \vee \alpha_2) \to \beta)$$

Fuzzy Rule Classifier
Background: De-fuzzyfication

What if we after the application of two rules have two memberships in β_1 and β_2, respectively, but need to choose one?

- We take the Bayesian approach: choose the one with the maximal membership.

Fuzzy Rule Classifier
Process

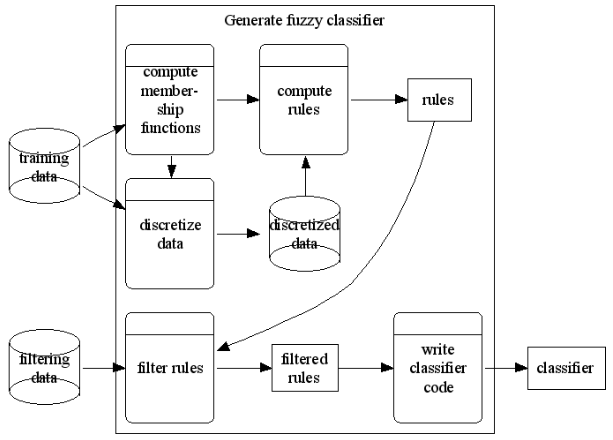
The induction of a fuzzy rule classifier from data process has several steps:

1. Data acquisition
2. Data preparation (aka. preprocessing)
3. Learning of rules
4. Validation of rules

For now, we will concentrate on step 3.

Fuzzy Rule Classifier
Induction

Step 3, learning fuzzy rules from data is a process of its own:

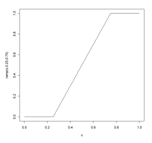


Fuzzy Rule Classifier

Generating Membership functions 1

Let $v < w$ and let ρ be defined as

$$\rho(x, v, w) = \begin{cases} 0 & \text{if } x \leq v \\ 1 & \text{if } x > w \\ \frac{x-v}{w-v} & \text{otherwise} \end{cases}$$



Let $s = v_0, v_1, \ldots, v_{n-1}$ be an increasing sequence of real numbers of size n. We associate n functions with s the following way:

$$\begin{aligned} \mu_0(x) &= 1 - \rho(x, v_0, v_1) \\ \mu_{n-1}(x) &= \rho(x, v_{n-2}, v_{n-1}) \\ \mu_i(x) &= \min(\rho(x, v_{i-1}, v_i), 1 - \rho(x, v_i, v_{i+1})) \end{aligned}$$

Fuzzy Rule Classifier

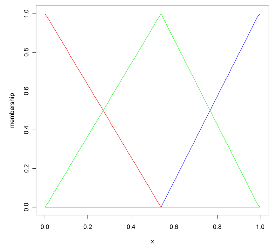
Generating Membership functions 2

How do we select the sequence?

- We use quantiles of the input vector x. For n levels we use the numbers
$$[-\infty, x_1, x_2, \ldots, x_n, \infty]$$
where x_i is the value at which $100 * (i-1)/n$ percent of the values found in x are smaller than x_i.

Example

If x consist of 40 uniformly sampled reals from the unit interval, and we want three fuzzy sets, we get:



Fuzzy Rule Classifier

Discretizing Data

Let f_i, $i \in \{1, 2, \ldots, m\}$ be m membership functions. A discretization of a vector $x = (x_1, x_2, \ldots, x_n)$ according to maximal membership is the vector $(l_1, l_2, \ldots, l_n)$ where l_j is such that $f_{l_j}(x_j)$ is the maximal of $\{f_1(x_j), f_2(x_j), \ldots, f_m(x_j)\}$.

Example

A column x for which we want three levels, i.e., we have the sequence $[-\infty, \min(x), \mathrm{median}(x), \max(x), \infty]$ we assign level $l(v)$ to all values v like:

$$l(v) = \begin{cases} 1 & \text{if } v \leq t_1 = \min(x) + \frac{(\mathrm{median}(x) - \min(x))}{2}, \\ 2 & \text{if } t_1 < v \leq t_2 = \mathrm{median}(x) + \frac{(\max(x) - \mathrm{median}(x)))}{2}, \\ 3 & \text{if } v > t_2. \end{cases}$$

Tables are discretized columnwise, excluding the class column.

Fuzzy Rule Classifier

Rule Induction

For each row i in the discretized data table T we approximate a minimum set A_i of non-outcome attributes that allow us to discern between i and all rows j that have a different outcome than row i. Given this ordered set A_i, we instantiate a rule as $((\wedge_{a \in A} a = T[i, a]) \to c = T[i, c]$ where c is the index of the outcome attribute.

Example

If $A = (a, b)$, $T[i, (a, b, c)] = (1, 1, 5)$, then we get the rule

$$((a = 1 \wedge b = 1) \to c = 5)$$

Fuzzy Rule Classifier

Rule Induction

Example

g_1	g_2	g_3	C
2	1	3	ALL
3	1	3	ALL
3	2	3	AML
2	1	2	AML

$$\begin{aligned} x_1 &: \{\{g_1, g_3\}, \{g_2, g_3\}^*\} \\ x_2 &: \{\{g_1, g_2\}, \{g_2, g_3\}^*\} \\ x_3 &: \{\{g_2\}^*\} \\ x_4 &: \{\{g_3\}^*\} \end{aligned}$$

$$\begin{aligned} M[x_1, x_3] &= \{g_1, g_2\} \\ M[x_1, x_4] &= \{g_3\} \\ M[x_2, x_3] &= \{g_2\} \\ M[x_2, x_4] &= \{g_1, g_3\} \end{aligned}$$

$$\begin{aligned} ((g_2 = 1 \wedge g_3 = 3) &\rightarrow C = ALL) \\ ((g_2 = 1 \wedge g_3 = 3) &\rightarrow C = ALL) \\ (g_2 = 2 &\rightarrow C = AML) \\ (g_3 = 2 &\rightarrow C = AML) \end{aligned}$$

Fuzzy Rule Classifier

Rule Filtering

The objective of rule filtering is to remove redundant rules. We determine redundancy relative to the combination of a set of rules R, a data set T, and a measure of classification quality q:

Definition

A rule r is *redundant* wrt. to R, T and Q if $Q((R \cup \{r\})(T), T) \leq Q(R(T), T)$, i.e, if adding rule r does not improve the classification of T.

Fuzzy Rule Classifier

Rule Filtering 1

Given a set of rules $R = \{r_i\}_{i=1}^m$ and a set of data points $T\{x_i\}_{i=1}^n$, we can form a matrix $V_{m \times n}$ by letting $V[i,j] = r_i(x_j) = I(\alpha_i, x)$ for the rule $r_i = (\alpha_i \rightarrow \beta_i)$. Further let J be the $m \times n$ matrix given by

$$J[i,j] = \begin{cases} 1 & \text{if } \beta_i \text{ agrees with the class of } x_j\text{, and} \\ -1 & \text{otherwise.} \end{cases}$$

Let $W[i,j] = V[i,j] * J[i,j]$.

Fuzzy Rule Classifier

Rule Filtering 2

Definition

Let R, T, W be defined as above. Let $\mathcal{I}$ be the smallest set of indexes such that

$$\sum_{j=1}^{n} I(\max_{i \in \mathcal{I}} W[i,j] + \min_{i \in \mathcal{I}} W[i,j] > 0) \geq \sum_{j=1}^{n} I(\max_{i=1}^{m} W[i,j] + \min_{i=1}^{m} W[i,j] > 0)$$

where $I(p)$ returns the truth value of the proposition p. The set $R' = \{r_i\}_{i \in \mathcal{I}}$ is a *filtering* of R relative to T.

In other words, a filtering of a rule set is the smallest rule set that does not classify T worse than R.

Fuzzy Rule Classifier

Writing Classifier Code

Once a set of rules has been found it is useful to be able to distribute this as an applicable package. We approach this by having a program transform our internal representation of our rules into a stand-alone program text that can be applied to data. Programming languages that

- allow computing on the "language", i.e., let you treat programs as data and vice versa, and
- can deal with higher order functions, i.e., functions that return functions,

make this relatively painless. Examples of such languages are Scheme, Lisp, and R.

Tuning of Filtering Parameter

Overview

Standard situation:

- One data set
- Want to create a classifier from the data set
- Have to tune learning parameters, and
- Validate our classifier

How do we deal with this having one data set? What follows is a strategy based on a standard way of eliciting model performance: cross validation (CV).

Tuning of Filtering Parameter

Cross Validation

Let M_θ be a model building process, i.e, $M_\theta(T) = C$, where C is a model (classifier) instance, T is training data, and θ are model building parameters. Also, let Q be a classification quality measure function such that $Q(C, T)$ is the quality of the classification of T by model instance (classifier) C.

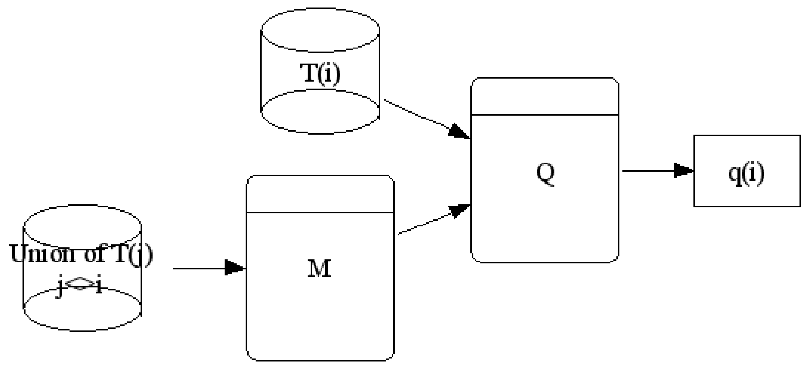
Definition

Let i be a positive integer. Partition T into i equally sized sets T_i. The vector $(q_1, q_2, \ldots, q_i)$, where $q_i = Q(M_\theta(\cup_{j\neq i} T_j), T_i)$ is the result of *i-fold crossvalidation*.

We can then look at the sample mean and sample variance of the values q_i.

Tuning of Filtering Parameter

Cross Validation



Crossvalidation: computing q(i)

Tuning of Filtering Parameter

Comparing Cross Validations

Let q and p be the results of two i-fold crossvalidations such that q_j and p_j were computed on the same training and evaluation data split. If we let $d_i = q_i - p_i$, we can then compare q and p in terms of a t-test for sample mean equality using

$$t = \frac{\overline{d}\sqrt{i}}{s_d},$$

a statistic that is T distributed with $i - 1$ degrees of freedom. Of interest is the $p = P(T > t)$ value. This value can be interpreted as the probability that the observed differences in means is by chance alone.

Tuning of Filtering Parameter

Procedure

Recall that our fuzzy classifier synthesis process (FCS) had two inputs:

- training data, and
- filtering data.

Given a data set T', let θ be the fraction of this data set that is randomly split off into a filtering data set F, leaving T''. Denote this process by $S_\theta(T') = (T'', F)$. We can now construct our classifier as $C = \mathrm{FCS}(T'', F)$. We can now empirically compare two different settings of θ, θ_1 and θ_2, by i-fold crossvalidation by letting

$$\begin{aligned} q_i &= Q(\mathrm{FCS}(S_{\theta_1}(\cup_{j \neq i} T_j)), T_i) \\ p_i &= Q(\mathrm{FCS}(S_{\theta_2}(\cup_{j \neq i} T_j)), T_i) \end{aligned}$$

Validation

Classifier Instance

A classifier instance can only be evaluated on data that has not been seen during the construction process. Unlike model performance evaluation, cross-validation is not a strategy we can apply. Having one data set, we cannot but

- making a simple split into training data T and *validation* data V.

Hence, our evaluation of our classifier C is

$$Q(C, V).$$

Overall Process

Having one data set D and one classifier learning method M, we can envision at least two ways of producing a tuned and validated classifier. They differ in how we view validation, whether we require the particular model instance to be validated or if we accept model evaluation.

Overall Process

Instance Validation

1. Split the data D into training set T and validation set V.
2. Perform CV parameter tuning procedure using only T.
3. Compute final model instance C from T using tuned parameters.
4. Validate model instance C on V as $q = Q(C, V)$.

Denote the final classifier resulting from this process as $C = \mathrm{TII}(M, T)$ (TII = "tuned instance induction").

Overall Process

Model Validation

The strategy here is to validate the induction process instead of the model instance, allowing the use of all the data for the construction of the final model.

1. Perform i fold crossvalidation on D where $q_i = Q(\mathrm{TII}(M, \cup_{j \neq i} D_i), D_i)$
2. Let the result of the crossvalidation be the validation of the final model $C = \mathrm{TII}(M, D)$.