# CIPER — a system for Constrained Induction of Polynomial Equations for Regression: User's manual

Peter Ljubič, Jožef Stefan Institute, Ljubljana, Slovenia

## 1 Introduction to CIPER

CIPER is a system for induction of regression models of a designated dependent variable $v_d$ from measured values of a set of numeric variables $V$, where $v_d \in V$. The regression models induced by CIPER are polynomial equations of the form $v_d = P$, where $P$ is a polynomial on variables in $V \setminus \{v_d\}$. It performs heuristic search through the space of candidate polynomial equations to find the one that has an optimal value of the heuristic function. The search strategy is based on refinement operator that orders the space of polynomial equations starting with the simplest one (i.e., $v_d = \text{const}$) and increasing its complexity at each step by adding a variable (1) to an existing term or (2) as a new term. The heuristic function is based on the minimal description length principle: it combines the degree of fit of the equation to the data with the complexity of the equation. CIPER can take into account syntactic subsumption (sub- and super-polynomial) constraints on induced polynomials.

This manual contains brief instructions for using CIPER. Further details about the refinement operator, heuristic function, and search strategy used in CIPER can be found in [2, 3, 1].

## 2 Implementation

CIPER is implemented in the C++ programming language using the Microsoft Visual C++ 6.0 development environment. It was developed on an Intel based PC under Microsoft Windows 2000 operating system.

## 3 Using CIPER

The CIPER program can be run using the following command:

```
ciper [options]
```

If CIPER is ran with no options set, it prints the options description. Option parameters are:

- `-l <class name>`, where `<class name>` specifies the left-hand side term of equation. If this option is not specified, the last attribute is set as left-hand side term.
- `-t <training set>`, where `<training set>` is the name of the file used as training set, containing attribute specification and attribute data. See section 5 for the details on the file format.
- `-T <testing set>`, where `<testing set>` is the name of the file used as testing set, containing attribute specification and attribute data. See section 5 for the details on the file format.
- `-b N`, where `N` is the the size of the beam used for beam search over a space of equations. If this option is omitted, beam size is set to 1 (greedy search).
- `-d N`, where `N` is the maximum depth of a single term on the right-hand side of an equation. If this option is omitted, the depth is unlimited.
- `-r N`, where `N` specifies the maximum number of terms on the right-hand side of an equation. If this option is omitted, the maximum number of terms on right-hand side is unlimited.
- `-p <sub-polynomial>`, where `<sub-polynomial>` specifies the sub polynomial of the right-hand side of the equation. For example "x" is sub polynomial of "x+y". One can interpret this constraint as what attributes we would like to get on the right-hand side of an equation. If not specified no constraints on sub polynomial are used.
- `-P <super-polynomial>`, where `<super-polynomial>` specifies the super polynomial of the right-hand side of the equation. For example "x*x+y*y" is super polynomial of "x+y". One can interpret this constraint as what is the maximum polynomial we would like to see on the right-hand side of an equation. If not specified no constraints on super polynomial are used.

The following are some examples of use. Let's assume we have a training file named data.arff containing five attributes, a, b, c, d, and e. We want to predict the value of attribute `d` from the rest of the attributes. Let's say we want to find an equation using beam size of 10. We use the following command:

```
ciper -l "d" -t data.arff -b 10
```

data.arff is specified to be a training file, `d` is specified to be the dependent variable (which is the left-hand side of an equation), beam size is set to 10, and no constraints are used (depth and max terms are not set, therefore they default to unlimited, sub-polynomial and super-polynomial are not set as well). Let's assume the system finds the following equation:

```
d = 1.39*a*a*b*c + 0.326*b*b*c + 1.5421*c*a + 1.254
```

Note there are 3 terms on the right-hand side, the depth of the equation is 4 (since the first term contains 4 variables), and the equation the system finds contains no attribute `e`. There are two particular reasons why one uses constraints. First is the complexity of computation. If the system spends more than the desired amount of time searching for the equation, we can limit the search space using constraints. The other reason is we are not satisfied with the equation the system finds (for example no attribute `e` in equation from previous example). For details on constraints see Section 4.

When we intend to check the performance of equation found on testing set (for prediction purposes), we must specify one with `-T` option. With following command line

```
ciper -l "d" -t training-file.arff -T testing-file
```

produces output, which gives us error information about the prediction made on testing set. The output contains the following measures:

- `RE`: is defined as the mean square error, relative to mean square error of constant prediction.
- `MSE`: mean square error on testing set.
- Correlation: gives us the correlation coefficient between the predicted and actual values of an attribute.

# 4 Constraints

## Depth

With the use of the depth constraint we limit the system to find equations with depth equal to or less than the depth we specify. Using the command line

```
ciper -l "d" -t data.arff -b 10 -d 2
```

the system finds equation where any term of the equation has at most 2 variables (attributes). So its degree is at most 2.

## Max terms

Using the max terms constraint we limit the system to find equations where the number of terms on the right-hand side is at most the one specified. With the command line

```
ciper -l "d" -t data.arff -b 10 -r 3
```

we limit the system to find equation with a total number of terms on the right-hand side that does not exceed 3.

### Sub polynomial

Let's say we are not satisfied with the equation the system found in the example above, because it contains no e attribute. We can force it to include e in the equations found by using the following command

```
ciper -l "d" -t data.arff -b 10 -p "e"
```

If we wanted attributes e and a to be in the equation, we use the command

```
ciper -l "d" -t data.arff -b 10 -p "e+a"
```

Similarly we can express even stronger conditions, for example

```
ciper -l "d" -t data.arff -b 10 -p "a*a + b*c"
```

where we force the system to find equation containing both a*a and b*c.

### Super polynomial

With this constraint we tell the system what is the most complex equation we would like to find. With next command

```
ciper -l "d" -t data.arff -b 10 -P "a*a + b"
```

the system finds equation, which is sub-polynomial of the equation a*a + b.

By combining those four constraints, we can profit even more. But be careful when specifying constrains, since it's very easy to specify a combination of constraints impossible to satisfy. Such an example is: max terms at most 2 and sub polynomial "a+b+c". Since a+b+c already consists of 3 terms, we cannot get the equation containing less then 3 terms, but with max terms constraint of 2, we would like to achieve exactly this impossible goal.

## 5 Input file format

CIPER reads WEKA (.arff) files. Values must be comma delimited. CIPER reads integer or real (numeric) attributes. If attribute is nominal or binary, error occurs. See `http://www.cs.waikato.ac.nz/ ml/weka/arff.html` for more information.

## References

[1] S. Džeroski, L. Todorovski, and P. Ljubič. Inductive databases of polynomial equations. In *Proceedings of the Second International Workshop on Knowledge Discovery in Inductive Databases (at ECML/PKDD-2003)*, pages 28–43. Rudjer Bošković Institute, Zagreb, Croatia, 2003.

[2] L. Todorovski, S. Džeroski, and P. Ljubič. Discovery of polynomial equations for regression. In *Proceedings of the Sixth Conference on Information Society (Intelligent and Computer Systems Volume)*, pages 151–154. Jožef Stefan Institute, Ljubljana, Slovenia, 2003.

[3] L. Todorovski, P. Ljubič, and S. Džeroski. Inducing polynomial equations for regression. Submitted to *The Fifteenth European Conference on Machine Learning (ECML-2004)*.