Chair III: Database Systems, Professorship of Data Mining and Analytics
Chair XXV: Data Science and Engineering
Department of Informatics
Technical University of Munich

ML2SQL

Compiling a Declarative Machine Learning Language to SQL and Python

Maximilian E. Schüle, Matthias Bungeroth, Dimitri Vorona, Alfons Kemper, Stephan Günnemann, Thomas Neumann
{m.schuele,matthias.bungeroth}@tum.de, {vorona,kemper,guennemann,neumann}@in.tum.de

ML2SQL

MLearn: Example 1 — TRANSLATE

```
A =
  [[1.1,0.98,87.3,3],
  [0.1,3.15,42.05,3.3],
  [100.5,26.8,10.1,225.1],
  [1097.5,23000,10.1,24850.1]]
X = A[: , 0:2]
y = A[: , 3]
bias[1,len(X,0)] : 1
X = (bias::X.T).T
Xt = X.T
w = (Xt*X)^(-1) * Xt * y
print '%' , w
```

Target Language: Python — EXECUTE

```
import numpy as np
  def main():
      A = np.array([ np.array([ 1.1,0.98,87.3,3]),
np.array([0.1,3.15,42.05,3.3]),np.array([ 100.5,26.8,
10.1,225.1]),np.array([1097.5,23000,10.1,24850.1])])
      X = A[ :,0:2 +1]
      y = A[ :,3:3 +1]
      bias = np.full( ( 1,np.size(X ,0 )), 1)
      X = (np.append(bias,X.T, axis=0)).T
      Xt = X.T
      w = np.dot( np.dot( np.linalg.matrix_power(
(np.dot( Xt,X)), (-1)),Xt),y)
      print( '{}'.format(  w))
  if __name__ == "__main__": main()
```

```
$ python < main.py

[[ 0.12238861]
[ 1.9759119 ]
[ 0.98614977]
[-0.00300479]]
```

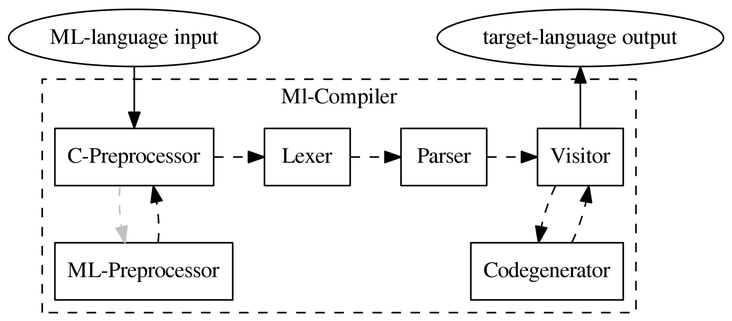
Web interface for an interactive exploration of the *MLearn* language: Above left, the text editor allows to specify tasks, which are translated into the selected target language (above right). The code will be executed in the terminal.

Abstract

- MLearn: Declarative machine learning language
- ML2SQL: Compiler to translate MLearn to SQL and Python
- Destination Systems: PSQL, HyPer, Python with TensorFlow and NumPy
- Features: Gradient descent and algebra including tensors

Conclusion

- Potentials of a declarative ML language:
  - expressing tasks compactly
  - being independent of the underlying engine
- Future work:
  - development of efficient array data types
  - standardised integration of optimisation methods such as gradient descent inside of database systems



The compilation process: the *MLearn* language first gets preprocessed twice for handling includes, then the language gets tokenised and parsed. For each target platform, a generator allows to translate the abstract syntax tree into the target language.

Grammar

```
expression: INJECT | 'print' mathexplist | 'if' '(' mathexp ')' '{' explist '}' ['else' '{' explist '}'] | ('continue' | 'break')
 | 'while' '(' mathexp ')' '{' explist '}' | 'for' VARNAME ('from' mathexp 'to' mathexp | 'in' interval) '{' explist '}' | functions
 | 'create' 'tensor' VARNAME 'from' VARNAME '(' varlist ')'
 | 'save' 'tensor' VARNAME 'to' (VARNAME|STRING) [':' STRING] '(' varlist ')'
 | VARNAME '[' accessor (',' accessor)* ']' ('=' | ':') mathexp | VARNAME (',' VARNAME)* '=' VARNAME '(' [mathexp (',' mathexp)*] ')'
 | VARNAME '[' accessor (',' accessor)* ']' '~' VARNAME '(' [mathexp (',' mathexp)*] ')' | 'import' VARNAME
 | VARNAME '=' (mathexp | 'distribution' '(' VARNAME ',' VARNAME ')') | VARNAME '~' VARNAME '(' [mathexp (',' mathexp)*] ')'
 | returnType* 'function' VARNAME '(' [VARNAME (',' VARNAME)*] ')' '{' explist '}' | 'return' mathexp (',' mathexp)*
 | ('readcsv'|'writecsv') '{' (('name:' VARNAME) | ('file:' STRING) | ('columns:' varlist) | ('replace empty entries:' mathexp)
                              |('delimiter:' STRING) | ('replace:' '{' (STRING ':' STRING)+ '}') | ('delete empty entries'))+ '}'
 | 'gradientdescent' '{' (('function:' STRING) | ('data:' varlist) | ('optimize:' [nameshape (',' nameshape)*])
                         | ('learningrate:' mathexp) | ('maxsteps:' mathexp) | ('batchsize:' mathexp) | ('threshold:' mathexp))+ '}'
 | 'plot''{'(('xData:'mathexp) | ('yData:'mathexp) | ('xLabel:'STRING) | ('yLabel:'STRING) | ('type:'STRING) | ('filename:'STRING))+'}'
 | 'crossvalidate' '{' (('minfun' ':' VARNAME '=' fun) | ('kernel' ':' VARNAME ':' VARNAME ':' mathexp)
                       | ('data' ':' VARNAME (',' VARNAME)*) | ('n' ':' mathexp) | ('lossfun' ':' fun)
                       | ('folds' ':' mathexp) | ('test' '{' (VARNAME '=' interval)+ '}'))+ '}' ;
```

Listing 1: Grammar of the MLearn language: Predefined building blocks for gradient descent, cross validation, CSV file handling as well as procedural control blocks, function calls and the declaration of variables.