Machine Learning for Computer Vision
Handwritten Math Recognition

Harold Mouchère

November 6, 2019

1 Context

The aim of this project is to show that deeplearning tools (MLP, CNN, ...) are often embedded in a more complex problem. Here the problem is "Recognition of On-line Handwritten Mathematical Expression" (ROHME), which is one of the task of the CROHME competitions.

In this document, the main aspect of the project are presented, but you have to define your own solution.

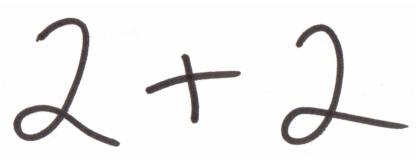
2 Inputs and Outputs

The inputs and outputs for this exercice will be those required for the CROHME competition. By this way, you will be able to use the (python) tools provided in the competition kit.

2.1 Inputs: Inkml

Ink traces with ground-truth information providing the segmentation of strokes into symbols and the symbol layout are provided in the InkML format. Each expression is stored in one InkML file. This XML language is defined by the W3C (http://www.w3.org/TR/InkML/) and can easily be extended. A CROHME InkML file contains:

1. Ink traces: X and Y coordinates of each point sampled nearly equally in time, grouped into strokes with XML identifiers,
2. information about the sample and the writer: unique identifier of the expression (IUD), writer identifier, writer age, gender and handedness if available, and finally
3. ground-truth.



(a) Ink

```
<mrow>
    <mn id="2_1">2</mn>
    <mrow>
        <mo id="_1">+</mo>
        <mn id="2_2">2</mo>
    </mrow>
</mrow>
```

(b) MathML

Figure 1: A simple handwritten expression with four strokes (a), its representation of symbol layout using MathML (b).

The ground-truth contains the following: LaTeX string: often a rendering of this string was shown to the writer when the formula was written, MathML (http://www.w3.org/Math/) representation of the expression using the Presentation encoding, representing the layout of symbols (with their XML identifiers) on baselines, and not their mathematical semantics, Segmentation of strokes into groups with a symbol label and XML identifier; each segment (symbol) is linked to its corresponding MathML symbol by the symbol's identifier. This format contains all the necessary information needed for training a system. Ground-truth information is removed from test files.

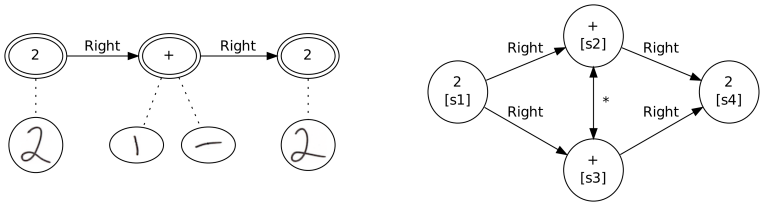
Figure 1(a) displays the ink traces as written by the user. The ground truth information defining the ME "2+2" with its layout is represented in MathML in Figure1(b). The <mrow> tag represents that the contained children are on the same baseline (row). <mn> and <mo> tags define numbers and operators. An additional section in the InkML file contains the segmentation information, which is not shown in Figure 1. Each group of strokes belonging to a symbol are assigned the same label, comprised of the symbol class, '_' and an index to distinguish multiple occurrences of a symbol (e.g. '2_1' and '2_2' in Figure 1(b)).

Inkml files can be rendered with this on-line tool: http://saskatoon.cs.rit.edu/inkml_viewer/

2.2 Output: LG

LG handles for "Labelled Graph". A handwritten expression can be represented by a labeled graph, where nodes are strokes and edges represent relationships between strokes and symbols. Figure 2 provides an example of how label graphs are defined.

From the MathML tree, we can create a symbol relation tree (SRT). Five relations are defined: Right, Above, Below, Sup, Sub and Inside. In Figure 2(a) we can see the two "Right" relationships between the three symbols (shown as nodes with double circles), defined by <mrow> tags in Figure 1(b). Each symbol is associated with its corresponding strokes (shown by dotted edges in Figure



(a) SRT

(b) LG

Figure 2: The symbol layout for the same handwritten expression with four strokes (a), and the stroke label graph (b).

2(a)). From this SRT, the layout graph of strokes can be constructed using the same spatial relation edges plus segmentation edges as in Figure 2(b). The segmentation edges were labeled by '*' in 2013, but in this project, as in the 2014 competition, segmentation edges are labeled with the label of the corresponding symbol. Nodes (strokes) from the same symbol are fully connected (i.e. form a clique), and share the same label. This allows several levels of segmentation in the same graph, with multiple labels on edges and nodes. This feature was used to define the matrix recognition task (the levels were: symbol, cell, row, column, and matrix).

The LG file format for label graphs was designed to be as simple as possible, human-readable, and easily used in any programming language. As shown in Figure 3, the graph is defined by a list of nodes and edges in a Coma Separated Values (CSV) text file. Nodes are defined using stroke identifiers (matching stroke identifiers in the InkML file), and a label. Edges are defined using the node identifiers they connect and a spatial relation or segmentation label. The last value of each line indicates a confidence value for the given label (symbol or relation). For the ground-truth files, these values are always 1.

2.3 Evaluation

Evaluation is performed by comparing the respective label graphs for the ground-truth and your system output. Consequently, your system should output LG files.

The provided evaluation allows to measure the recall and precision of node (stroke)'s labels and edge (relation)'s labels. Furthermore, it also compute recall and precision metric at the symbol level considering the segmentation deduced from your LG file. Thus, remember to produce valid LG files with fully connected nodes of symbols.

The script to use is `evaluate` which can evaluate two complete directories (all files) of a partial list of files.

```
N, s1, 2, 1.0
N, s2, +, 1.0
N, s3, +, 1.0
N, s4, 2, 1.0
E, s1,s2, Right, 1.0
E, s1,s3, Right, 1.0
E, s2,s3, +, 1.0
E, s3,s2, +, 1.0
E, s2,s4, Right, 1.0
E, s3,s4, Right, 1.0
```

(a) Normal

```
O, sym1, 2, 1.0, s1
O, sym2, +, 1.0, s2, s3
O, sym3, 2, 1.0, s4
R, sym1,sym2, Right, 1.0
R, sym2,sym3, Right, 1.0
```

(b) Short

Figure 3: The two versions of the same LG file for the expression "2+2" of Figure 2(b).

3 Global recognition process

This section describes briefly the 3 main steps of a ROHME.

3.1 Segmentation

This step consists in grouping the strokes to form hypotheses of symbols. It should be noted that in on-line expression, most of the symbols are written with consecutive strokes, but sometimes there are "delayed strokes" which come back to complete an existing stroke.

Thus, two main strategies are possible:

1. stroke order dependent: the ROHME is able to recognize only symbols written with consecutive strokes;
2. stroke order free system: the ROHME is able to deal with symbols written with multi-strokes symbols not written consecutively; you can reduce the complexity by adding constraints as allowing only one "time jump" in a symbol.

You can use a segmentation classifier to decide if an hypothesis should be kept or not. This is not mandatory.

3.2 Recognition

This step is simply an isolated symbol recognition task. Each hypothesis of symbol should be recognized and associated to one of several labels.

Several classifiers can be used: HMM using the time information of the strokes, MLP using extracted features, or a CNN using a image version of the symbol. The CNN solution is the simplest. The classifier can also have a reject class (junk) to reject samples which are probably wrong segmentations.

You can use the 'CROHME-train-iso.py' script as an example of symbol classifier.

The output of this step can be save in a "short version" of LG file: one file per expression, one line per symbol hypothesis (the same hypothesis can appear several times with different labels).

3.3 Layout Analysis

Depending of your progress, this step can be skipped.

This step recognizes the spatial relation between 2 hypothesis of symbols among: Right, Sub, Sup, Above, Below, Inside.

This recognition is generally based on features extracted from the bounding boxes of each symbol (difference of baseline, of height, distances...) with a simple classifier (SVM or MLP). It exists also some image based solutions using an image containing both symbols in the same small image (high resolution is not needed of that).

The output of this step can be saved by just add "R" relations in a "short version" LG files.

3.4 Final decision

Normally this step should use a grammar or another language model to select the best complete and valid expression using the available strokes and hypothesis (symbols + relations).

In the frame of this project, your system should use a strategy to select the set of symbols and relations which minimizes the global cost of the expression (sum of all costs of used symbols and relations). You can propose a greedy algorithm or a dynamic programming solution. If you have not recognized the spatial relations, used simply Right relations between ordered symbols.

The simplest appraoch is a greedy algorithm which select the best symbol hypothesis, then remove all incompatible hypothesis (because of the used strokes) and repeat until the set of remaining strokes is empty.

4 Data

The CROHME dataset is available from TC11 (Technical Committee 11, Reading Systems) website on CROHME's page [1]. You will find this content:

- Full expressions inkml and LG files split in Train, Validation and Test;
- Isolated symbol samples split in train, validation and test sets. Note that each isolated sample come from an existing full expression.

[1] or full expressions and isolated symbol datasets can be downloaded locally here

5 Tools

Both libraries ‘lgeval‘ and ‘crohmelib‘ are available on-line extradoc and DPRL website[2] which contains also several useful explanations. Remember to define the environment variable, as explain in the documentation:

```
export CROHMELibDir=[path_to_CROHMELib]
export LgEvalDir=[path_to_LgEval]
export PATH=$PATH:$CROHMELibDir/bin:$LgEvalDir/bin
```

For each tool, a short documentation is available in the sources or if you call it without any parameter.

We provide a global chain process in several steps:

- `segmenter.py` : Generate from an inkml file hypotheses of symbol in a LG file
- `segmentSelect.py` : Keep or not each segment hypotheses and generate a new LG file
- `symbolReco.py` : Recognize each hypothesis and save all acceptable recognition in a LG file
- `selectBestSeg.py` : From an LG file with several hypotheses, keep only one coherent global solution (greedy sub-optimal)

To run an evaluation, you will need the tool ‘`evaluate`‘ from ‘`lgeval`‘ which will compare 2 directories containing both LG files, one with ground-truth files, the other with your output. If you want to evaluate a subpart of the dataset, you have to build the list of corresponding files with ‘`listExistingLG.sh`‘

[2] https://www.cs.rit.edu/~dprl/CROHMELib_LgEval_Doc.html