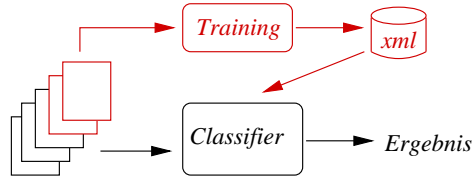


## 5.1 kNN in Gamera (1)

Schema Klassifizierung:



Zwei Schritte:

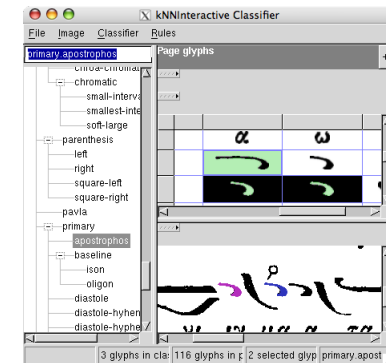
- interaktives *Training*
  - Beispielbildern wird manuell eine Klasse zugewiesen
  - Ergebnis ist eine Datei mit markierten Trainingsdaten (XML-Format in Gamera)
- automatisches *Klassifizieren*
  - *Classifier*-Objekt lädt Trainingsdaten
  - Bilder werden übergeben und Classifier gibt Klassen zurück

1

## 5.1 kNN in Gamera (3)

Der Trainingsdialog

- Glyphen können markiert werden (Ctrl/Shift + Click)
- markierten Glyphen kann Klasse zugewiesen werden (Textfeld links oben)
- Namenshierarchie durch Punkte als Trenner, z.B. "upper.a", "lower.b"
- Ergebnis speichern über "File"-Menü:
  - "Classifier-Glyphs" alle im Classifier gespeicherten (incl. ggf. aus Datei geladenen)
  - "Page-Glyphs" nur die des aktuell geladenen Dokuments



3

## 5.1 kNN in Gamera (2)

Klasse *kNNInteractive*

- definiert im Modul knn

```
from gamera import knn
```
- Konstruktor

```
classifier=knn.kNNInteractive(imagelist,\n                             ["aspect_ratio", "moments"], 0)
```

  - *imagelist* ist Liste zu klassifizierender Bilder
  - zweites Argument ist Liste der Featurenamen
- Laden von Trainingsdaten aus Datei

```
classifier.from_xml_filename("data.xml")
```
- Starten des interaktiven Trainingsdialogs

```
classifier.display(images_to_classify)
```

  - wird typischerweise aus dem Gamera-GUI aufgerufen (Menü "Classify")
  - vollständiges Dokumentbild über Menü "Image" ladbar:
    - Bild wird mit wählbarer Segmentierung zerlegt (typischerweise CC-Analysis)

2

## 5.1 kNN in Gamera (4)

*kNNInteractive* Methoden zum Klassifizieren

- `classify_list_automatic(imagelist)`  
klassifiziert jedes Bild aus *imagelist* und speichert das Ergebnis in Properties des Bildes (s.u.)
- Methoden zum Ändern Parameter:
  - `change_feature_set([feat1; feat2; ...])` zum Ändern Features
  - über Property `num_k` kann Wert von *k* in kNN eingestellt werden

Abfrage der zugewiesenen Klasse

- Liste von Klassen wird als Tupel (*confidence*, *classname*) in Eigenschaft *id\_name* hinterlegt, z.B.

```
[(0.8, 'lower.b'), (0.2, 'lower.d')]
```
- Abfrage mit `glyph.get_main_id()`, was äquivalent ist zu `glyph.id_name[0][1]`

4

## 5.1 kNN in Gamera (5)

### Fragmentierte Zeichen (1)

- zwei mögliche Situationen
  - Zeichen besteht per Definition aus mehreren Komponenten, z.B. Buchstabe "klein i" oder Doppelpunkt
  - Zeichen ist eigentlich nur eine Komponente, aber durch schlechte Bildqualität fragmentiert
- Behandlung beim Training
  - alle Teile des Zeichens markieren, z.B. durch "Rubber Band" Markierung in der Bildanzeige
  - den Klassennamen mit Präfix *\_group*. versehen
  - Gruppe wird als ein Zeichen im Trainingssatz gespeichert

5

## 5.1 kNN in Gamera (7)

### Berührende Zeichen

- Wenn deterministisch zusammengesetzt (z.B. Ligaturen), mit Präfix *\_split.splitalgorithm* trainierbar.  
Angegebener *splitalgorithm* (z.B. *splitly*) wird dann auf diese Glyphen angewandt.
- Alternative: als eigenes Symbol trainieren (z.B. "ligature.ft")
- zufällig berührende Zeichen derzeit nicht in Gamera behandelbar

### Behandlung von Noise

- meist auch nach Preprocessing reichlich vorhanden
- einfachste Lösung: als eigene Klasse trainieren  
funktioniert in Praxis recht zuverlässig

7

## 5.1 kNN in Gamera (6)

### Fragmentierte Zeichen (2)

- Behandlung beim Klassifizieren
  - `group_and_update_list_automatic(imglist, \grouping_function, max_parts_per_group=3)`  
verwenden, wenn gebrochene Zeichen vorkommen
    - ▷ gibt Liste klassifizierter Bilder zurück, in der Gruppen zusammengefasst sind
    - ▷ *grouping\_function* steuert, wann zwei Glyphen als Nachbarn betrachtet werden, die möglicherweise zur selben Gruppe gehören können
      - sinnvoller Wert: *ShapedGroupingFunction(max\_distance)*
      - *max\_parts\_per\_group* gibt Obergrenze für Anzahl Fragmente an
- Algorithmus prüft Graph von Hypothesen
  - schlimmstenfalls exponentielle Laufzeit  
⇒ kann steckenbleiben ⇒ Parameter *max\_distance* und *max\_parts\_per\_group* möglichst klein wählen

6