Prototype of an automated classification service: a use case for KDD?

F.-X. Pineau[1]

[1]CDS, Observatoire Astronomique de Strasbourg

Santiago, 28th October, 2017

Université de Strasbourg

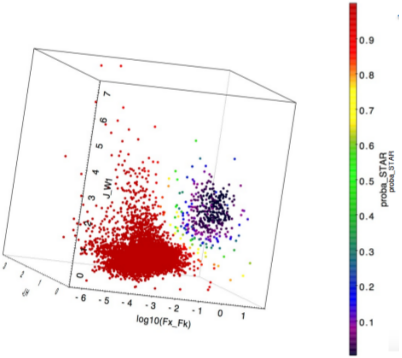
Observatoire astronomique de Strasbourg

Centre de Données Astronomiques de Strasbourg

Motivations

- Probabilistic cross-matches: take into account photometric data to update purely positional posteriors
  - Make two learning samples: real and spurious matches
  - Use kernel smoothing to compute photometric likelihoods (add astrometric priors $\rightsquigarrow$ Kernel Density Classification)

- Meeting a few fellow astronomers one-time needs:
  - Separate stars from extra-galactic sources
  - e.g. XMM-GSC-2MASS (A. Klutsch, P. Guillout)
  - e.g. XMM-2MASS-WISE, params: J-H, Fx/Fk, J-W1 (E. Sanchez, A. Nebot)



Credit: A. Nebot, E. Sanchez

Motivations

- See 4. of KDD Charter: "**Defining requirements for implementing and adding machine learning capabilities to services**".
- Process X-match outputs before downloading results
- Use Kernel Smoothing to define complex query regions:
  - upload a learning sample table (e.g. WD)
  - ask the service for all sources having a likelihood $p(\vec{x}|wd) > 0.01$
  - likelihood $p(\vec{x}|wd)$ computed by Kernel Smoothing
  - (likelihood = local density / numer of points in the sample)
- Use Kernel Density Classification to retrieve objects of given class (see Kai's talk at ADASS)
  - build/upload learning samples
  - ask for all sources having $p(qso|\vec{x}) < 0.8$

Disclaimer

- The prototype service supports two simple classification algo:
  - k-nearest neighbours (kNN)
  - kernel density classification (KDC)
  - (could also support Mean Shift clustering, usefull?)
- Fulfil a part of a classification process
  - Does not support empty values
  - No feature selection / dimentionality reduction
  - Does not automatically select the "optimal" parameter(s)
  - ...

The k-NN classification

- Supervised method: need a learning sample (LS)
- To classify one object:
  - simply look at the *k* nearest neighbour in the LS
  - assigned class is the most common in the neibhours
- Pros
  - No learning stage (lazy classification)
  - Very easy to understand/interpret
  - Very easy to implement and to mutli-thread
  - Fast algorithms available (regular kd-tree for the Euclidean distance)
- Cons
  - Curse of dimentionality: dimentionality reduction first
  - Overfitting/underfitting: how to define the "best" possible value for *k*?
  - Too simple

The Kernel Density Classification

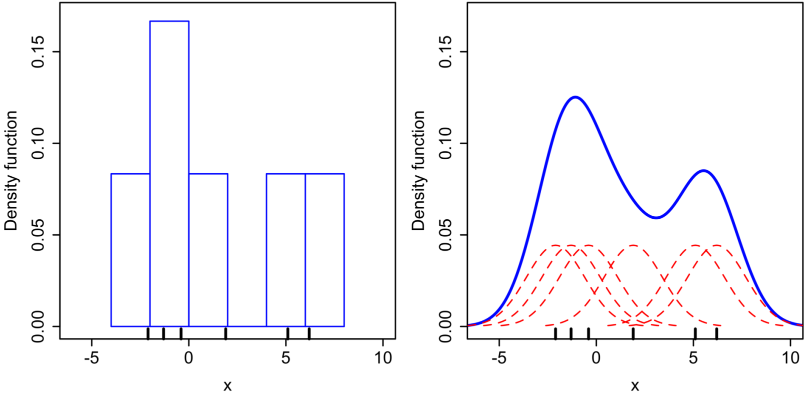
- Original paper: Richards et al. (2004)
  - star/quasar (c_1/c_2) classification from $\vec{x}$ =(u-g, g-r, r-i, i-z)
- Supervised method: requires a learning sample for each class c_i
- Direct application of the Bayes' formula

$$p(c_i|\vec{x}) = \frac{p(c_i)p(\vec{x}|c_i)}{\sum_{j=1}^{n} p(c_j)p(\vec{x}|c_j)} \tag{1}$$

  - c_i: object class
  - $\vec{x}$: vector in the parameter space
  - $p(c_i)$: user defined priors
    - iterate while priors $\neq$ posteriors means
  - $p(\vec{x}|c_i)$: likelihoods (p.d.f) computed by kernel smothings (KS)
    - one KS by learning sample class

Histogramming vs KS in 1D

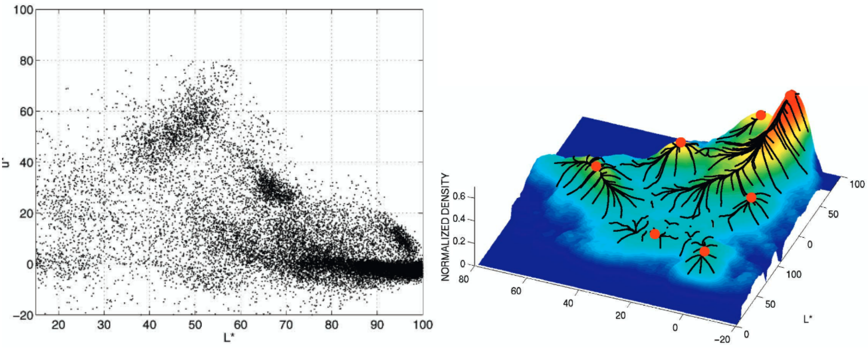
- KS: density = sum of kernels centered around each data point
- Normalized density = probability density function (p.d.f)



Credits: `https://en.wikipedia.org/wiki/File:Comparison_of_1D_histogram_and_KDE.png`

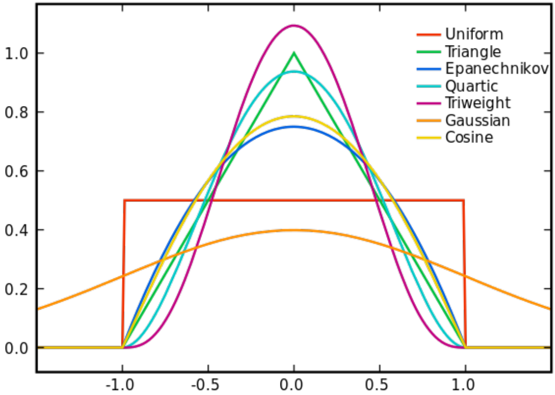
Kernel smoothing in 2D

- KS: density = sum of 2D kernels (e.g. 2D Gaussians) centered around each data point
- Normalized density = probability density function (p.d.f)



Credits: Comaniciu, D. and Meer, P. (1997)

Kernels



Credits: `https://en.wikipedia.org/wiki/File:Kernels.svg`

- We use only the multivariate **Epanechnikov** kernel
  - finite support (unlike Gaussian kernels)
  - theoretically the best (even if it is not that important)

Various Kernel Smoothings

- Fixed bandwidth: all kernels have the same bandwidth
- Variable/Adaptative bandwidth
  - balloon estimator:
    - 1 fixed bandwidth per density estimation
    - bandwidth = distance to the measurement point's k^{th}-NN
  - knn averaging: balloon estimator with a uniform kernel
  - sample-point estimator:
    - 1 bandwidth per data point in the LS
    - data point bandwidth = distance to the data point's k^{th}-NN

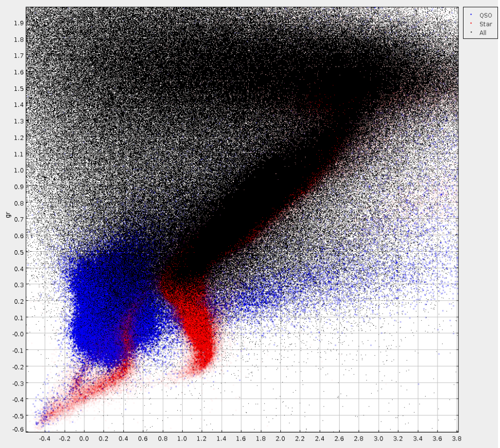
KDC pros and cons

- Pros
  - Easy to understand, to interpret and to implement
  - Natural probabilities in output
  - Fast algorithms (based on kd-tree for exemple), easy to multi-thread
  - No randomness (but results depends on the choosen bandwidth)
- Cons
  - How to choose the "best" bandwidth to avoid under/over-fitting?
  - Curse of dimensionality
  - Not the current trend (random forest)?
- Keep in mind that: the quality/representativity of the LS is often more important than the chosen supervised algo!

KDC example

Input data

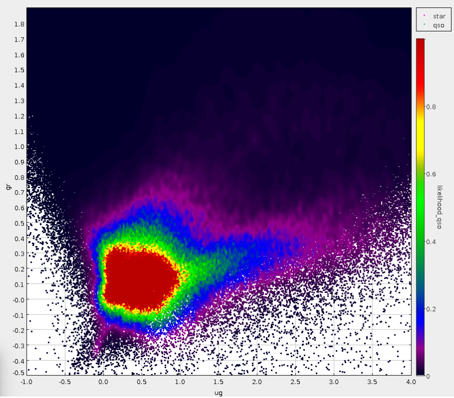
- Unresolved SDSS sources with a good photometric quality;
- Parameters: $u - g$, $g - r$;
- LS: 727 000 stars (red); 402 000 quasars (blue)
- 5 000 000 unknow sources

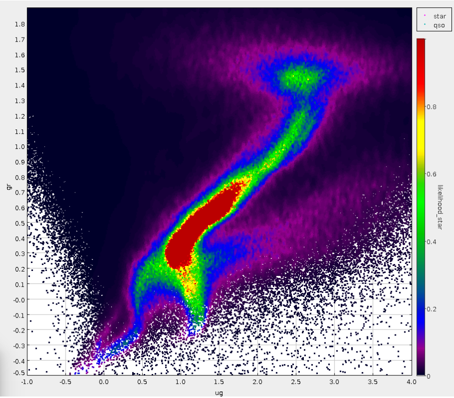


Disclaimer: raw data selection (no quality check), no tests to select the best bandwidth, LS not representative of the data, ...

KDC example

Normalized local densities (p.d.f.) of both learning samples – quasar (left) and star (right) – in the parameter space $(u-g, g-r)$.

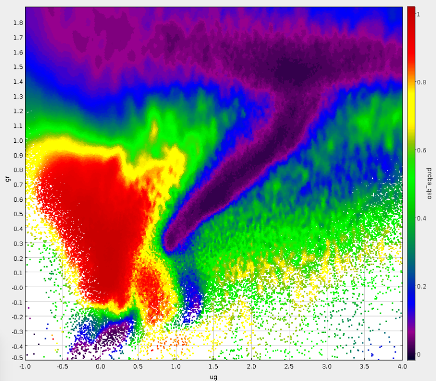


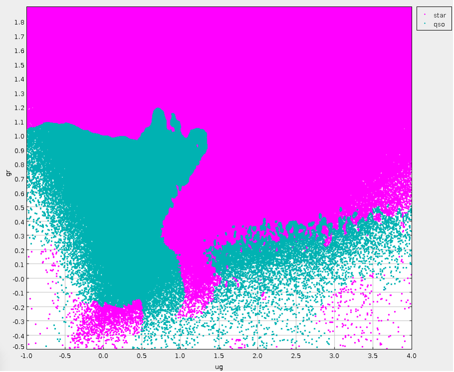


$$p(\vec{x}|qso) \qquad p(\vec{x}|star)$$

KDC example

Probability of being a quasar (left) and repartition of sources classes as quasar/star (right).





$$p(qso|\vec{x}) = \frac{p(qso)p(\vec{x}|qso)}{p(qso)p(\vec{x}|qso)+p(star)p(\vec{x}|star)}$$

Cyan: $p(qso|\vec{x}) < 0.5$

Pink: $p(qso|\vec{x}) \geq 0.5$

Running the example

Using a script calling the REST HTTP API of the service

```
# Create a new working dir on the server
> ./classif.bash mkdir
SlHHdHKh8DRLUKUhniusua109p
# Configure the script to use the created directory
> ./classif.bash setdir SlHHdHKh8DRLUKUhniusua109p
# Put data on the server (LS + to be classified)
> ./classif.bash put qso   qso.2d.csv
> ./classif.bash put star star.2d.csv
> ./classif.bash put data data.2d.csv
# Give a name to parameters (optional)
> ./classif.bash put labels ug,gr
# Start automated classifciation
> ./classif.bash kdc samplepoint -k 100 \
             -p qso:0.35\;star:0.65 -ho > res.2d.csv
```

It took 58 s to classify 5 000 000 sources ("training" included) (the 15-NN classification took 10 s).

Classification algorithm

Basic algorithm using the sample-point estimator:

```
# Structure creation
for learning sample classes (quasar,  stars)
  load classes objects
  build a kd-tree
  for each object
    perform a k-NN query
    put the object with computed extent in a M-tree
# Classification
for all object to classify
  for each learning sample class
    get overlapping objects in the M-tree
    compute the normalized local density
  apply Bayes' formula
```

Confusion matrix

```
# Compute the confusion matrix in percentages
./classif.bash kdc samplepoint -k 100
        -p qso:0.35\;star:0.65 -cr
```

	predicted quasar	predicted star
actual quasar	80.7%	19.3%
actual star	11.6%	88.42%

- A part of white dwarfs are classified as quasars
- The LS should have been cleaned (magnitude errors, ...)
- TODO: divide the learning sample in two to remove biases

The computation of the confusion matrix took 18 s (build trees on 1 100 000 objects and classify 1 100 000 objects).

The service REST HTTP API

```
# Create/remove a new working directory on the server
POST/DELETE ${url}/mkdir

# Create|replace/add/get/remove ls classes/data
PUT/POST/GET/DELETE ${url}/${dir}/learningsample/${clas}
PUT/POST/GET/DELETE ${url}/${dir}/data

# Perform an ‘ls’ on the learning sample
# - possible outputs: txt/xml/json
GET ${url}/${dir}/learningsample/ls

# Create|replace/get/remove parameter names
PUT/GET/DELETE ${url}/${dir}/header
```

The service REST HTTP API

```
# Perform a k-NN classification
# - output: csv
GET ${url}/${dir}/knn
GET ${url}/${dir}/knn/learningsample
# - possible outputs: txt/xml/json
GET ${url}/${dir}/knn/confusionmatrix

# Perform a Kernel Density Classification
GET ${url}/${dir}/kdc/${algo}
GET ${url}/${dir}/kdc/${algo}/learningsample
GET ${url}/${dir}/kdc/${algo}/confusionmatrix
# Params: priors, likelihoods thresholds, ...

# 4 kernel smoothing algorithms
${algo} = knn|fixedbandwidth|balloon|samplepoint
```

Compatibility with DALI

```
# Pure DALI URLs
GET ${url}/availability
GET ${url}/capabilities
GET ${url}/examples        # Facultative

# VOTables responses available for errors and all
# queries (except a GET returning a CSV file).
# Response format depends on the HTTP header:
Accept: text/plain         # For cmd line tools
Accept: application/json   # For a web pages
Accept: application/xml    # DALI compatible VOTable
Accept: text/xml           # DALI compatible VOTable
```

About the service

- Reuse CDS cross-match codes (mutli-threaded kd-tree and M-tree)
- Performances depends on
  - the number of classes (only for KDC)
  - the number of objects returned by kd/M-tree queries
  - the smoothing algo (M-tree slower than kd-tree)
- Support concurrent access (Read/Write locks)
- Support failures during the writting phase (rollback to the previous state)
- Working directory removed if not used during 10 days
- Implementation full Java using the Jersey framework (JAX-RS)

Come see me for a demo.

Thank you!

Richards, G. T., Nichol, R. C., Gray, A. G., Brunner, R. J., Lupton, R. H., Vanden Berk, D. E., Chong, S. S., Weinstein, M. A., Schneider, D. P., Anderson, S. F., Munn, J. A., Harris, H. C., Strauss, M. A., Fan, X., Gunn, J. E., Ivezić, Ž., York, D. G., Brinkmann, J., & Moore, A. W. 2004, The Astrophysical Journal, Supplement, 155, 257. astro-ph/0408505