Problem description: "Feature selection at training test time"

**Goal** Fast classification (no waste of processing) in the following setup

- Complex classification: problem too hard to plan full automation ⇒ confidence estimation is needed (for rejection)
- Several types of features are available, from raw data to high-level features
- Low-level features are much faster to be computed
- Low-level features are sufficient to identify well some particular classes

**Application** Classification of scanned documents mainly for mailroom automation

- Many classes, severe class imbalance, noisy input features: 10%–30% error rate at full automation
- Input features: Sub-Resolution gray-levels, Document Layout Analysis, Pre-defined Patterns Detection, Automatic Text Transcription, ...
- Feature extraction CPU times: from 50 ms/doc. for sub-resolution, to 5 sec/doc. for transcription

A cascade decision tree algorithm

**Inputs**
- one test sample \( x \)
- \( N \) "group-of-features extractors" \( f_i \)
- ranked by increasing CPU needs
- \( N \) confidence-rated classifiers \( C_i \) (*) each trained on first features \( \{f_i\}_{k=1}^{N} \)

(*) assume that the more input features there are, the more accurate the classification is

**Params**
- confidence thresholds \( \{t_i\}_{i=1}^{N-1} \)
- confidence correction functions \( \{s_i\}_{i=1}^{N} \)

**Outputs**
- predicted class \( \hat{c} \)
- confidence \( s(\hat{c}|x) \)

* for \( i = 1 \) to \( N \) do
  * Compute group of features \( f_i(x) \)
  * Compute outputs of classifier \( C_i \) on \( \{f_i\}_{k=1}^{N} \)
  * predicted class \( \hat{c} \) and confidence \( s_i \)
  * if \( i = N \) or \( s_i \geq t_i \) then
    * return \( \hat{c} = C_i, s(\hat{c}|x) = t_i(s_i) \)
  * end if
* end for

**Costs and optimization**

The parameters of the cascade tree are optimized on a labeled validation set

**Confidence thresholds \( \{t_i\}_{i=1}^{N-1} \)**

We define three types of additive costs:

<table>
<thead>
<tr>
<th>Cost</th>
<th>Back-end answer</th>
<th>Could the answer be true with...</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_0 \geq 0 )</td>
<td>TRUE</td>
<td>LESS features</td>
</tr>
<tr>
<td>( C^- &gt; 0 )</td>
<td>FALSE</td>
<td>LESS features</td>
</tr>
<tr>
<td>( C^+ = 1 )</td>
<td>FALSE</td>
<td>MORE features</td>
</tr>
</tbody>
</table>

**Loss function to minimize**

Sum of these costs

**Optimization method**

Grid search on (quantized) possible thresholds combinations

**Confidence correction functions \( \{s_i\}_{i=1}^{N} \)**

Non-parametric approach (may overfit?) vs. Parametric approach (how much engineered?)

Conclusions

- The proposed testing cascade strategy relies on some properties of the core classifier.
- Empirical results are mixed.
- What are the alternative algorithms to select features at test time?