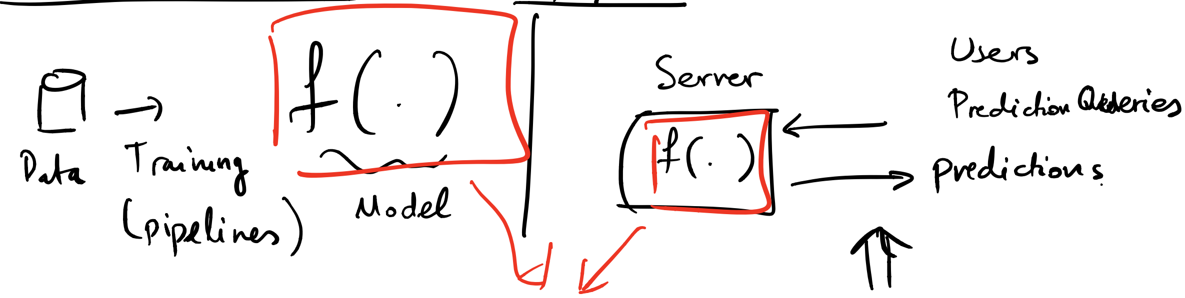
Today: Efficient model serving.

→ Compression, Relational data (embeddings)

Prediction Serving



Serving Systems

Models: as a black box
(reuse the same code as training)

Standard tricks for efficient serving for user requests.

Problem: Agnostic to ML.

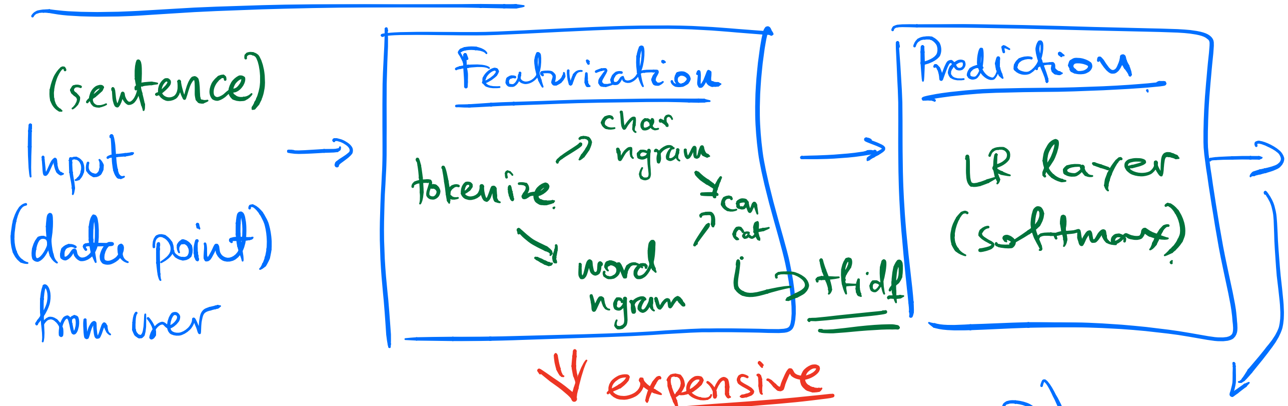
Focus only on prediction serving problems.

1) User requests
   * batching
   * result caching.
2) Increase user capacity
   * replicate the model. (parallel evaluation)

external optimization.

Q: What is the most expensive operation when serving machine learning models?

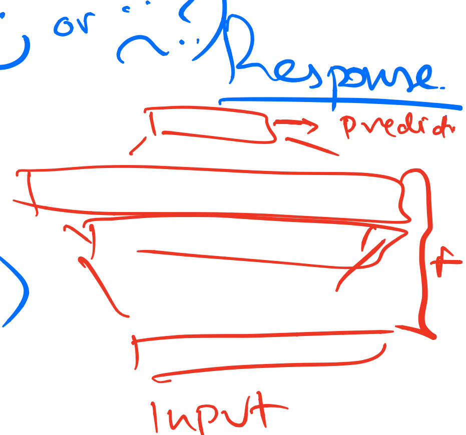
Model computation →



Task: sentiment analysis (☺ or ☹?)

Model 1: Logistic Regression

Model 2: LSTM (RNN)



Scenario: Multiple requests from users

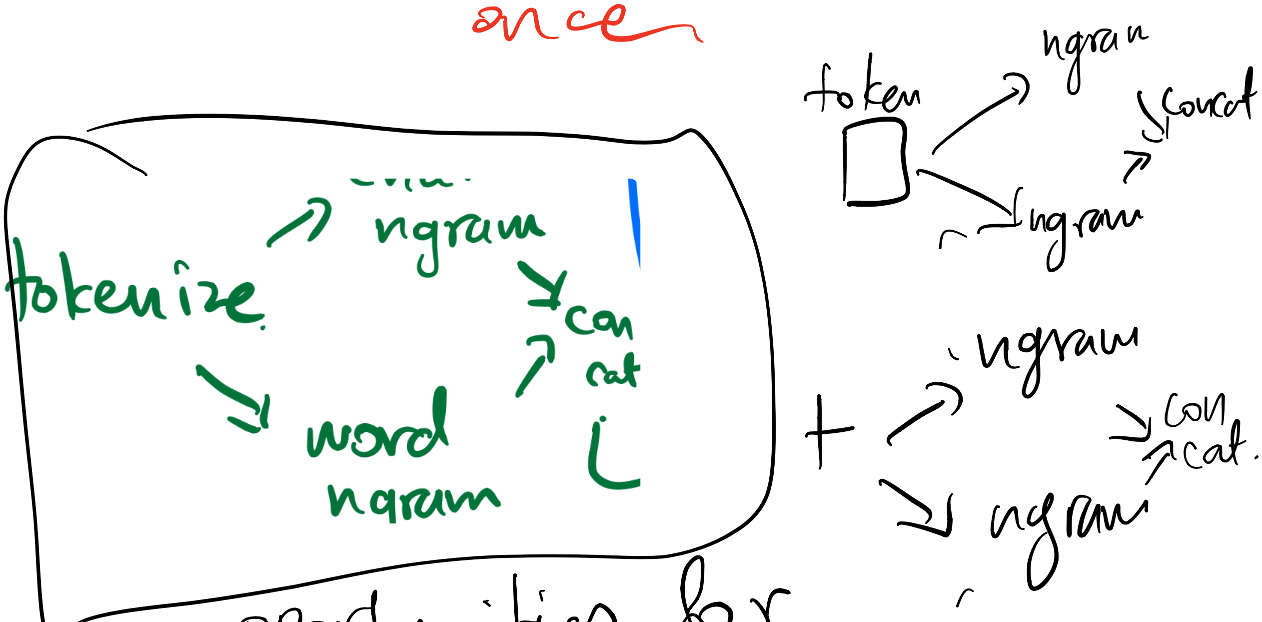
Featurization: → split tokens
→ extract char n-grams
word n-grams
→ find tfidf values
→ concat into a vector

Q: if we consider these ops as a black box?

You have two sentences that share 80% of the same words

Saving 1: remove redundant lookups (tfidf)

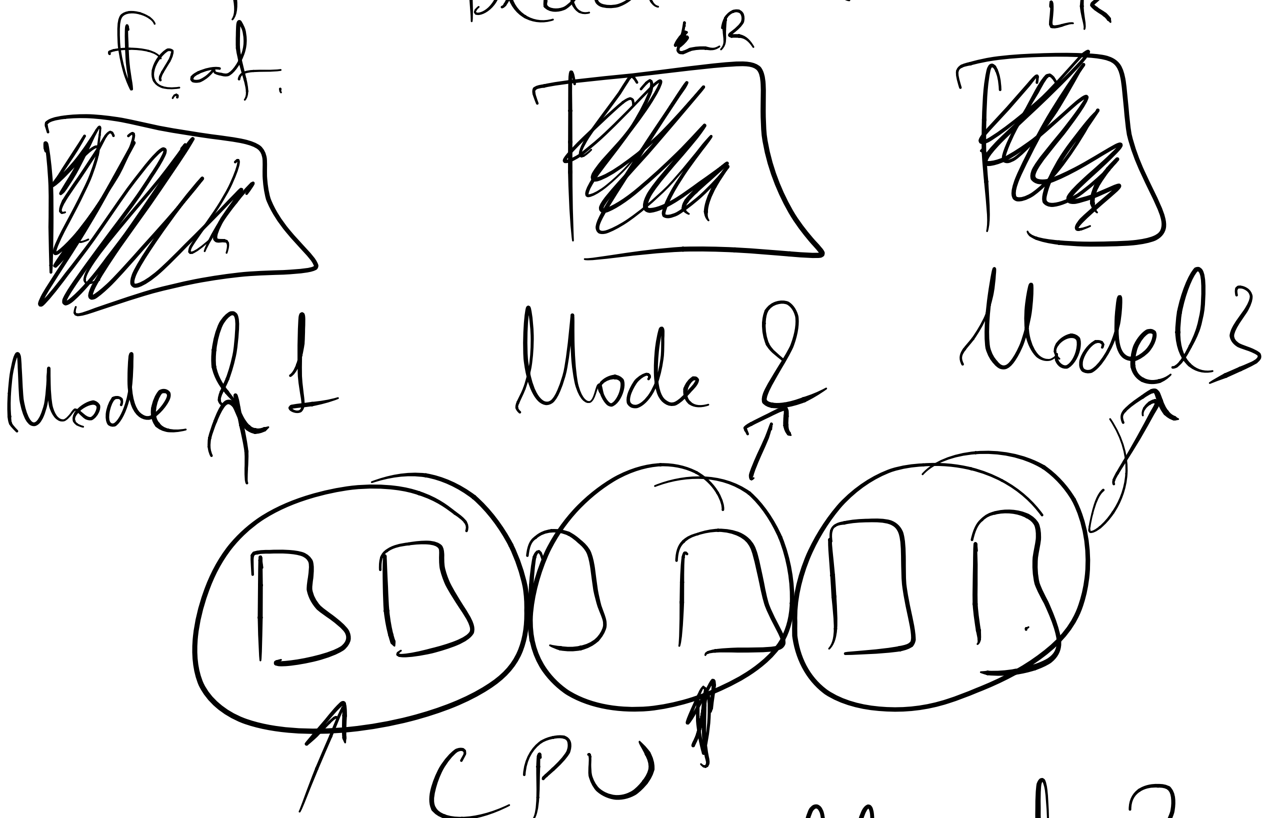
Batch → cache my ngrams and compute tfidf once



opportunities for sharing resources if I open the blackbox ?

Instead of having this setup

black boxes



How do you allocate CPUs to the Models?

Thursday

We need to open the model black box and look into individual components and processing?

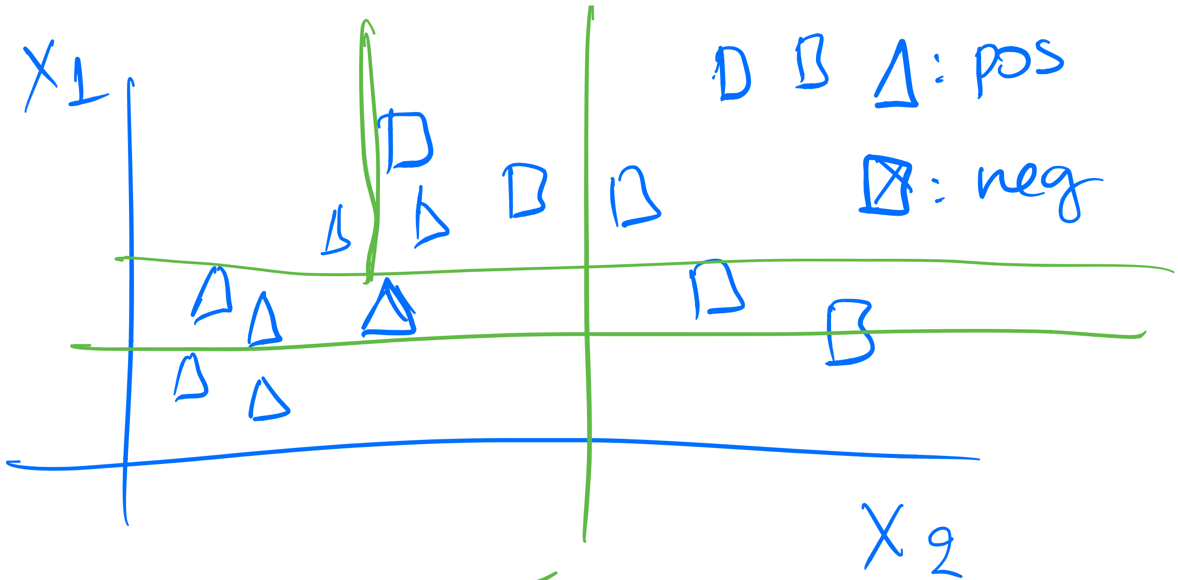
→ No all input are equally hard.

Model Cascades ⟷ Boosting

↪ weak learners ($p > 0.5$)

Combine weak learners into an ensemble

the ensemble is a strong learner ($p > 1 - \varepsilon$)



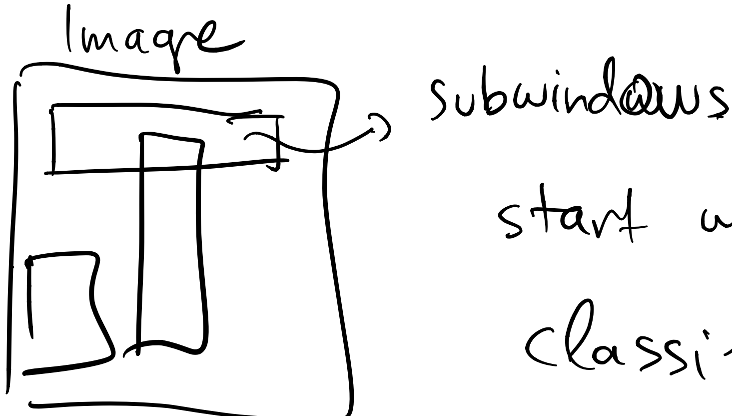
→ vertical line is a model that uses only 1 out of two features

→ weak learners are much cheaper to evaluate because they rely on fewer features

Cascading classifiers

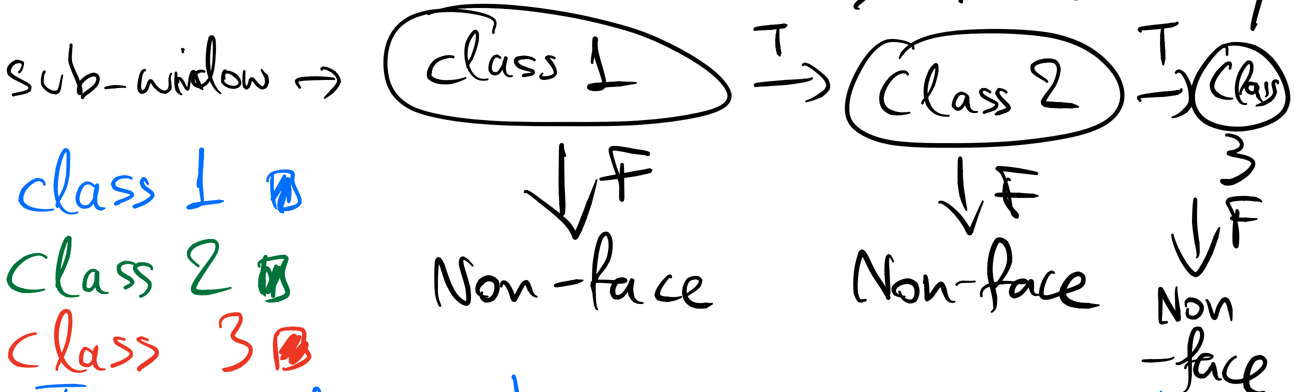
Viola/Jones Face Detector
2001

Goal: real time face detection



start with simple classifiers that reject many on the negative sub-windows while they detect almost all positive sub-windows

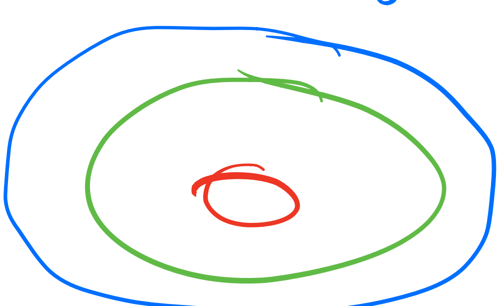
Result from the first classifier → trigger arrival from the second classifier.



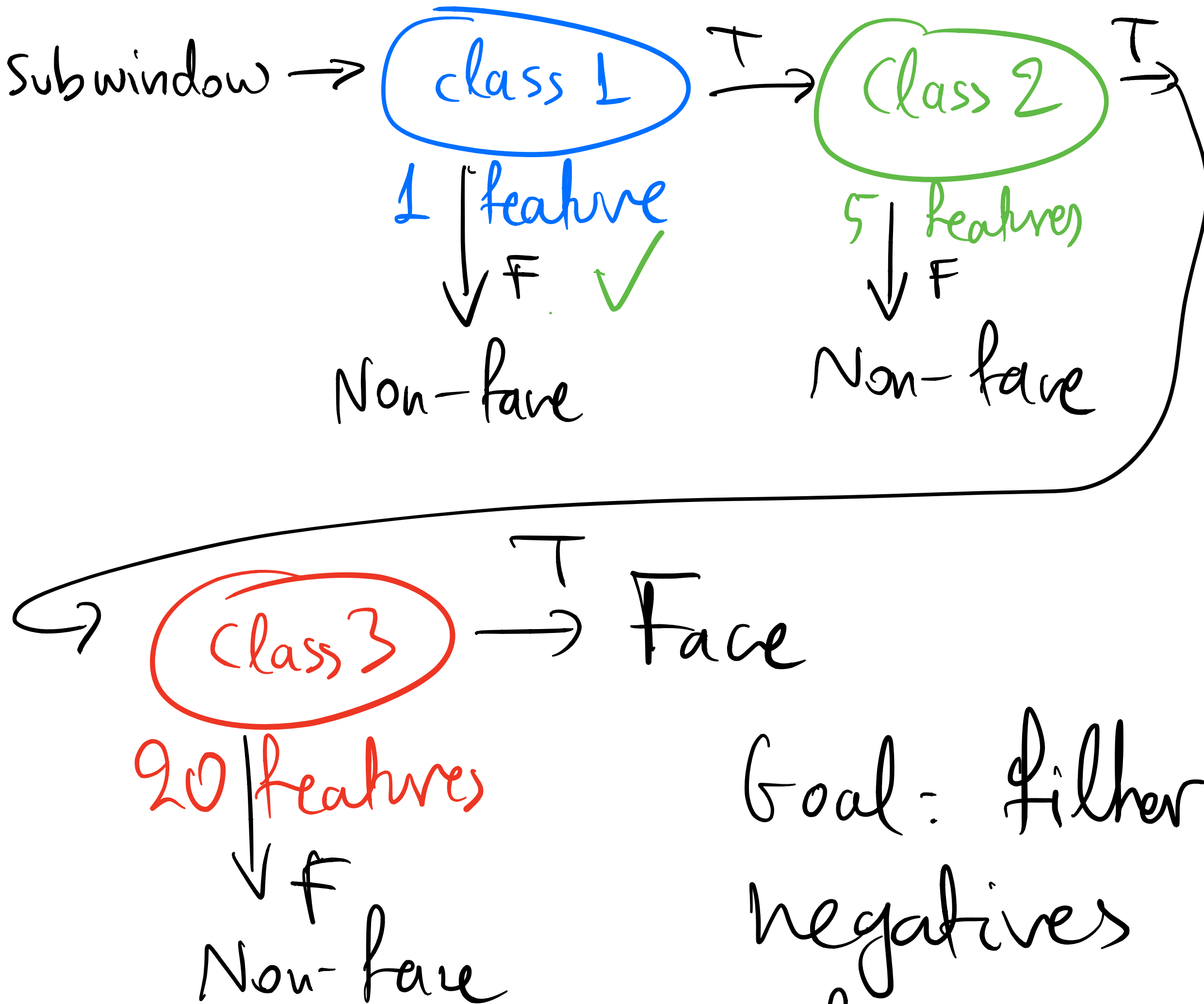
The negative outcome can come out at any point

but the <u>positive outcome</u> has to go through <u>all</u> classifiers

↳ classifiers with lower false positive rate.



↳ class 1 ⊃ class 2 ⊃ class 3



Goal: filter negatives fast

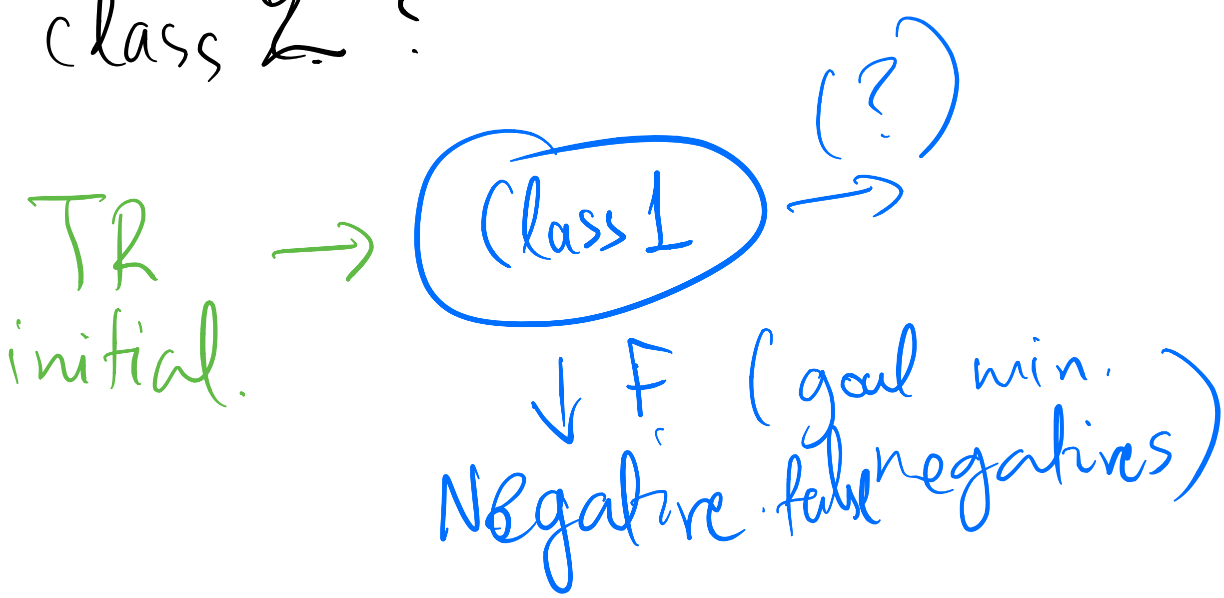
Q: How about training?

A classifier is "optimal" if it minimizes false negatives

* For each classifier our training goal is to minimize false neg instead of total prediction (classification error)

* Each classifier has a <u>different</u> training set (TR)

What is the TR of class 2?



TR initial. → TR for which initial Class 1 says "true"

(this dataset still contains false pos. non-face elements)

This training is expensive because I change the TR set while I propagate data through the cascade

Follow up: questions:

→ how do I learn the best cascade so that

I minimize → latency
→ resource usage

maximize accuracy.

→ Willump (MLsys 20)

Statical optimization to correlate features with different classes