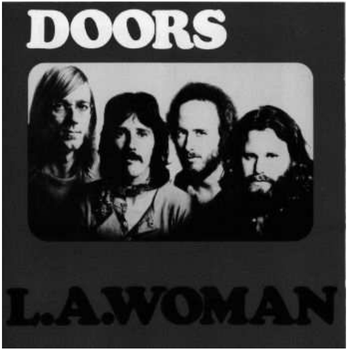
Bayesian Multiple Target Localization

Purnima Rajan (Johns Hopkins)
Weidong Han (Princeton)
Raphael Sznitman (Bern)
Peter Frazier (Cornell, Uber)
Bruno Jedynak (Johns Hopkins, Portland State)

Thursday July 9, 2015
International Conference on Machine Learning (ICML)
Lille, France

Here's the program

1. Study a Bayesian active learning problem.
2. Use our results to find faces in images like these:



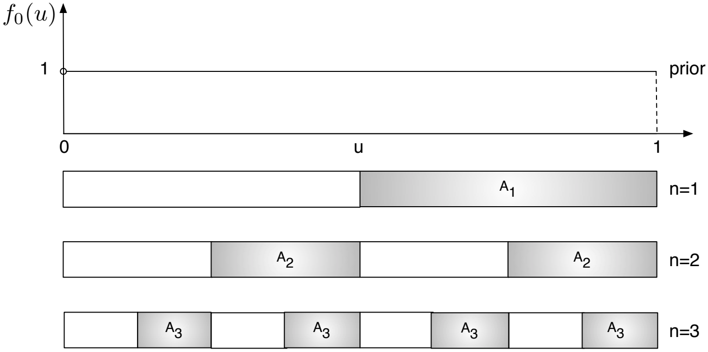
This is our Bayesian active learning problem: Localize targets by counting

- There are k targets in some set $\Omega = [0, 1]$.
- Their (unknown) locations are $\theta_1, \ldots, \theta_k \in \Omega$, drawn iid from a prior density.
- For $n = 1, \ldots, N$
  - We choose $A_n \subseteq \Omega$
  - We observe X_n.
    X_n's distribution depends on the number of targets in A_n.
- **Goal**: choose the questions to reduce uncertainty in the target locations, as measured by the expected entropy of the posterior.

The dyadic policy

The **dyadic policy** constructs A_n recursively:

- We partition Ω into 2^n regions.
- A_n is the union of half of these regions.
- To obtain the partition for A_{n+1}, we take the partition for A_n and split each region into two sub-regions with equal mass under the prior (but not necessarily equal size, if the prior is not uniform).



The dyadic policy can be easily parallelized

- The dyadic policy's questions A_n do not depend on the answers X_n.
- It is **non-adaptive**.
- This makes it easy to parallelize.
- We compare it with OPT, the value of an optimal **adaptive** policy.

The dyadic policy is a constant factor approximation to the optimal adaptive policy

Theorem

$$\frac{\text{DYADIC}}{\text{OPT}} \geq \frac{D_k}{C_k}.$$

- $\text{DYADIC} = H(p_0) - E^{\text{DYADIC}}[H(p_N)]$
  is the expected entropy reduction under the dyadic policy.
- $\text{OPT} = H(p_0) - \inf_\pi E^\pi [H(p_N)]$
  is the expected entropy reduction under an optimal adaptive policy.

The dyadic policy is a constant factor approximation to the optimal adaptive policy

Theorem

$$\frac{\text{DYADIC}}{\text{OPT}} \geq \frac{D_k}{C_k}.$$

- $\text{DYADIC} = H(p_0) - E^{\text{DYADIC}}[H(p_N)]$
  is the expected entropy reduction under the dyadic policy.
- $\text{OPT} = H(p_0) - \inf_\pi E^\pi[H(p_N)]$
  is the expected entropy reduction under an optimal adaptive policy.
- C_k is the channel capacity,

$$C_k = \sup_q H\left(\sum_{z=0}^{k} q(z) f(\cdot|z)\right) - \sum_{z=0}^{k} q(z) H(f(\cdot|z))$$

- D_k is this explicit expression,

$$D_k = H\left(\frac{1}{2^k}\sum_{z=0}^{k}\binom{k}{z} f(\cdot|z)\right) - \frac{1}{2^k}\sum_{z=0}^{k}\binom{k}{z} H(f(\cdot|z))$$

The dyadic policy is a 2-approximation to OPT in the noise-free case

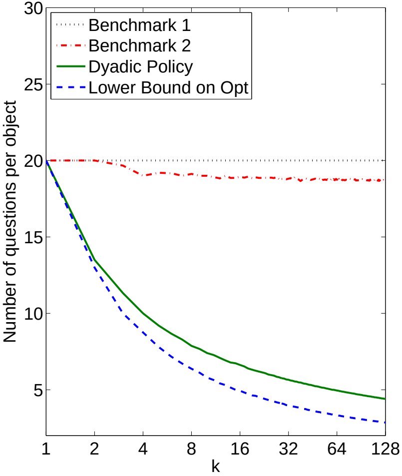
When observations are **noise-free**:

- $D_k = H(\mathrm{Bin}(k, \frac{1}{2})) \geq \frac{1}{2}$ and $C_k = \log(k+1)$, implying

$$\frac{\text{DYADIC}}{\text{OPT}} \geq \frac{H(\mathrm{Bin}(k, \frac{1}{2}))}{\log(k+1)} \geq \frac{1}{2}$$

- If $k = 1$ (there is a single target), then $\frac{\text{DYADIC}}{\text{OPT}} \geq 1$ and the dyadic policy is optimal among adaptive policies.
- Additional result not in the paper:
  In the noise-free case, for $k \geq 1$,
  the dyadic policy is optimal among non-adaptive policies.

Localizing k objects simultaneously is much faster than localizing them one at a time



- Plot shows: (# of questions to reduce total entropy by k $\times$ 20 bits)/k
- Benchmark 1 localizes targets one-at-a-time using bisection.
- Benchmark 2 is sequential bifurcation (Bettonvil and Kleijnen 1997).
- Green and blue lines come from our bounds.

Proof Sketch

Theorem

$$\frac{\text{DYADIC}}{\text{OPT}} \geq \frac{D_k}{C_k}.$$

- Expected entropy reduction of the optimal: $\text{OPT} \leq C_k N$.

Proof Sketch

Theorem

$$\frac{\text{DYADIC}}{\text{OPT}} \geq \frac{D_k}{C_k}.$$

- Expected entropy reduction of the optimal: $\text{OPT} \leq C_k N$.
- Expected entropy reduction of the dyadic: $\text{DYADIC} = D_k N$.

Proof Sketch

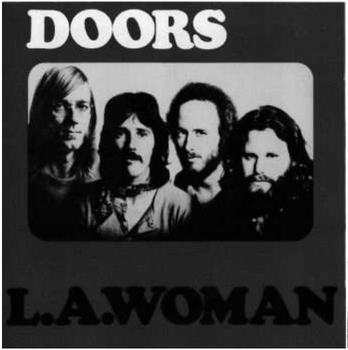
Theorem

$$\frac{\text{DYADIC}}{\text{OPT}} \geq \frac{D_k}{C_k}.$$

- Expected entropy reduction of the optimal: $\text{OPT} \leq C_k N$.
- Expected entropy reduction of the dyadic: $\text{DYADIC} = D_k N$.
  - The answers to the dyadic questions are **independent**.
  - The entropy reduction under the dyadic is the sum of the entropy reduction due to each question.
  - The number of targets in A_n is $\text{Binomial}(k, \frac{1}{2})$, allowing us to compute the entropy reduction from this question.

The dyadic policy can be used to localize multiple object instances in computer vision

Let's build a tool that can find faces in images like this:



The standard approach applies a high accuracy classifier at each pixel in the image

We have access to a high-accuracy classifier.

- input is the image, and a pixel within the image.
- output is whether or not there is a face centered at that pixel.
- we use a boosted collection of 500 stumps
  (features are windowed sums of oriented gradients)

Standard practice is to run the high-accuracy classifier at each pixel within the image to find all of the faces.

- But, this is slow.

We use the dyadic policy to localize multiple object instances

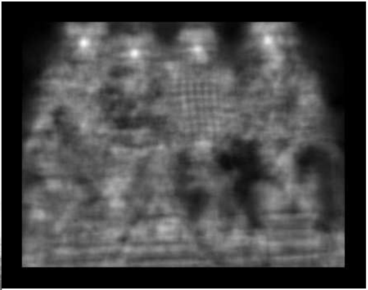
Instead, we do this:

- **(Screening)** Use the dyadic policy, together with a fast low-accuracy classifier, to compute a posterior distribution on object instance locations.
- **(Refinement)** Using the posterior to prioritize pixels, run a slow high-accuracy classifier on a (hopefully small) number of pixels to confirm the object instance locations.

Fewer calls to the high-accuracy classifier vs. the standard approach makes it faster.

Screening Step 1:
Run a fast low-accuracy classifier at all pixels

Step 1: Run a fast low-accuracy classifier at each pixel in the image.

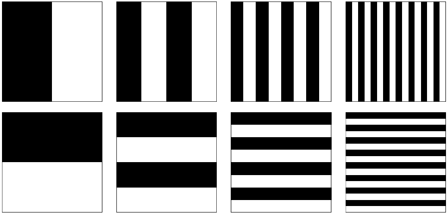


- Our low-accuracy classifier is a boosted collection of 50 stumps.
- 50 stumps is 10% of the high-accuracy classifier.

Screening Step 2: Answer the dyadic questions

Step 2: Compute the answers to the dyadic questions

- X_n is the sum of the low-accuracy response over the region A_n.
- The regions A_n, $n = 1, \ldots, 8$, are



- X_n can be computed quickly using an integral image.

Screening Step 3: Compute the posterior

Under other policies, computing the posterior is slow (needs MCMC).

Under the dyadic policy, **computing the posterior is fast**:

- Let $C = \cap_{n=1}^{N} B_n$, where each B_n is either A_n or $\Omega \setminus A_n$.
- The expected number of targets in C under the posterior is

$$E[\text{number of targets in } C|X_{1:N}] = k \prod_{n=1}^{N} \left(\frac{e_n}{k}\right)^{s_n} \left(1 - \frac{e_n}{k}\right)^{1-s_n},$$

  where

  - $s_n = 1\{B_n = A_n\}$.
  - $e_n = E[\text{number of targets in } A_n|X_n] = \sum_{j=0}^{k} \frac{jP(X_n = x_n|Z_n = j)P(Z_n = j)}{P(X_n = x_n)}$

Refinement

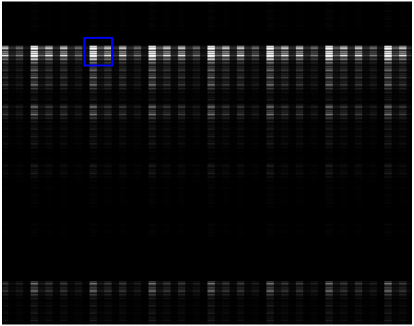
Following screening, we do refinement. We use one of these algorithms.

- **Posterior Rank (PR):** Rank pixels by the marginal probability of containing a target, and apply the high-accuracy classifier in this order.
- **Iterated Posterior Rank (IPR)**: Like Iterated Rank, but when a target is found, we "mask" the target and recompute the posterior and pixel order. Masking is done without recomputing the answers to the dyadic questions:

$$E[\text{number of } \textbf{new} \text{ targets in } C|X_{1:N}] = k \prod_{n=1}^{N} \left(\frac{e_n'}{k}\right)^{s_n} \left(1 - \frac{e_n'}{k}\right)^{1-s_n},$$

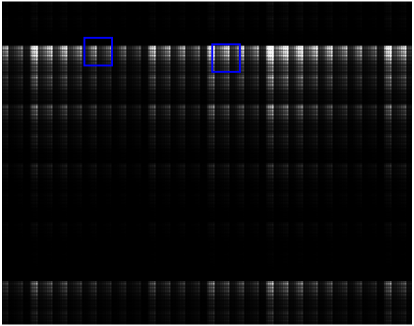
  where e_n has been replaced by e_n', the expected number of new targets in A_n, given X_n and the locations of discovered targets.

Example: Seinfeld, Iterated Posterior Rank

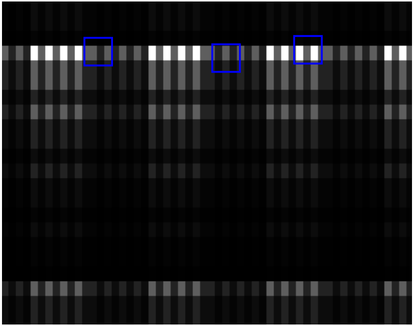
Iteration 1

Example: Seinfeld, Iterated Posterior Rank



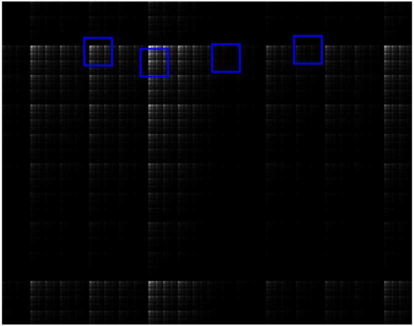
Iteration 2

Example: Seinfeld, Iterated Posterior Rank



Iteration 3

Example: Seinfeld, Iterated Posterior Rank



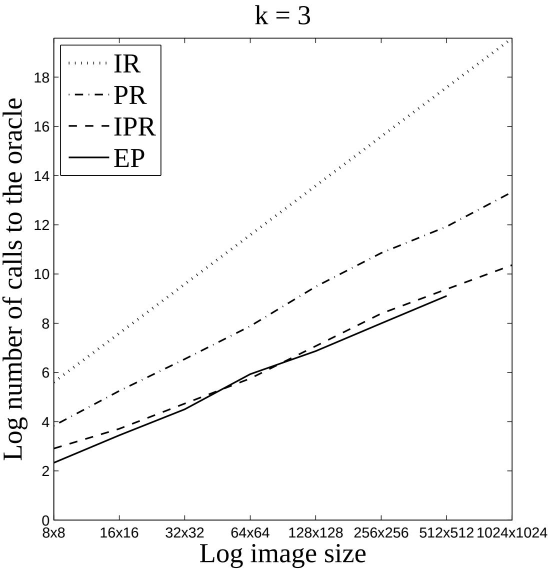
Iteration 4

The dyadic policy reduces the number of times we call the slow high-accuracy classifier on real images

Results for 35 images from the MIT+CMU corpus.

- Dataset contains 130 faces (3.7 faces per image on average)
- Dataset contains 1.2×10^7 pixels
- We found all the faces by running the high-accuracy classifier on only **3% of the pixels** using posterior rank.
- A naive approach would run it on more than 50% of the pixels.

The dyadic policy reduces the number of times we call the slow high-accuracy classifier on simulated images



- IR = iterated rank (baseline that goes in arbitrary order)
- (I)PR = (iterated) posterior rank (our algorithms)
- EP = entropy pursuit

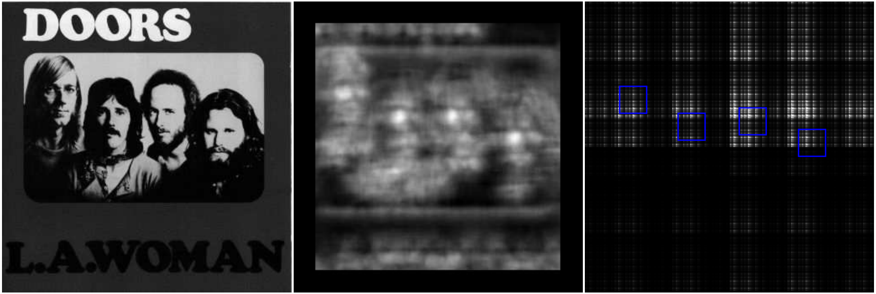
Future Work

- Practical improvements for computer vision
  - Unknown k (easy)
  - Integration of better weak / strong classifiers (easy)
  - Likelihoods that depend on the size of the queried region (hard)
  - Separate multiple scales (hard)
- Other applications
  - Screening for important input factors to computational codes.
  - Group testing [our model generalizes the classical model]
  - Combinatorial chemistry
  - Heavy hitter detection in network analysis

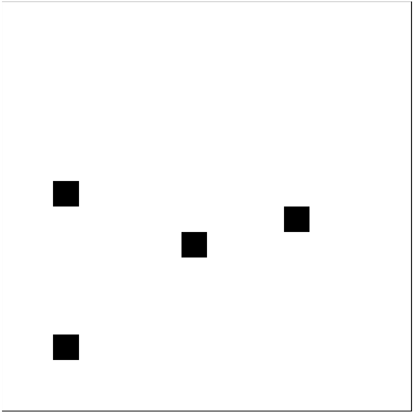
Thank You!

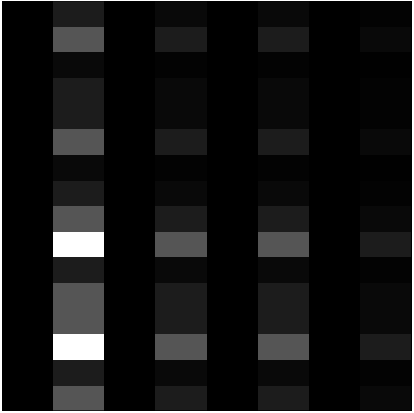
Backup

Example: The Doors, Posterior Rank

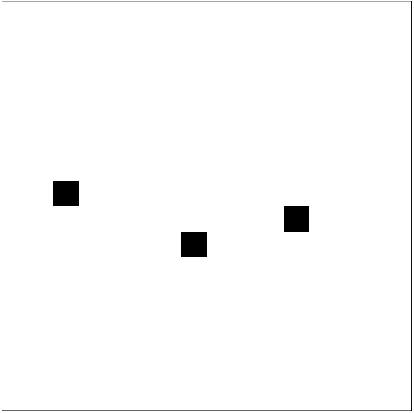


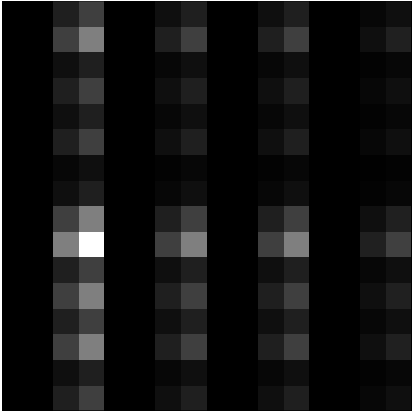
Simulated Image: Iterated Posterior Rank



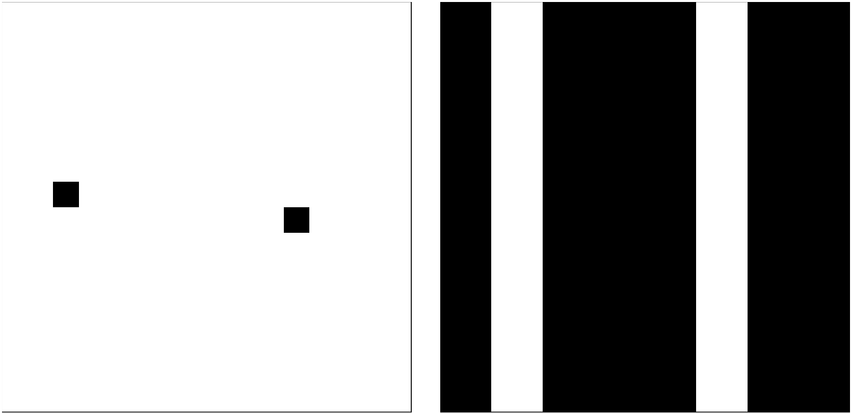


Simulated Image: Iterated Posterior Rank

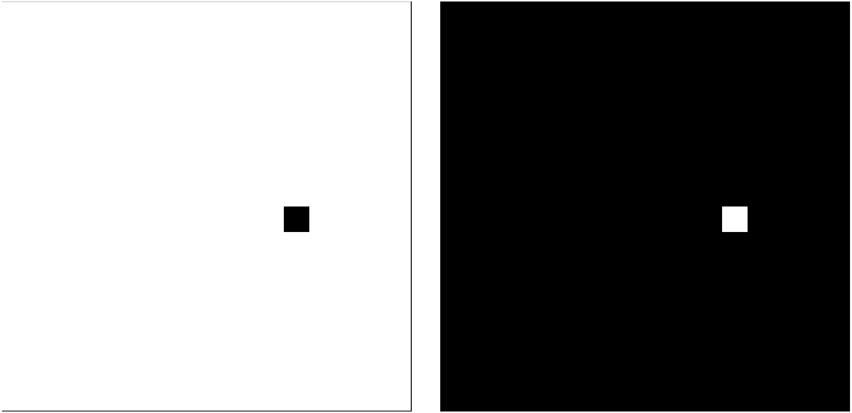




Simulated Image: Iterated Posterior Rank



Simulated Image: Iterated Posterior Rank



Integral Images

- $\text{Integral}(x, y)$ is the sum of the low-accuracy classifier's responses at all pixels below and to the left of y.
- $\text{Integral}(x, y)$ can be computed iteratively for all x and y by touching each pixel once.
- Sums over rectangles can be computed quickly from these values: sum within box = integral(upper left) - integral(upper right) - integral(lower left) + integral(lower right)

The dyadic policy reduces the number of times we call the slow high-accuracy classifier on simulated images