Duke UNIVERSITY

Improved remote sensing estimates by exploiting detector/classifier error patterns

Jordan Malof

2016.04.7

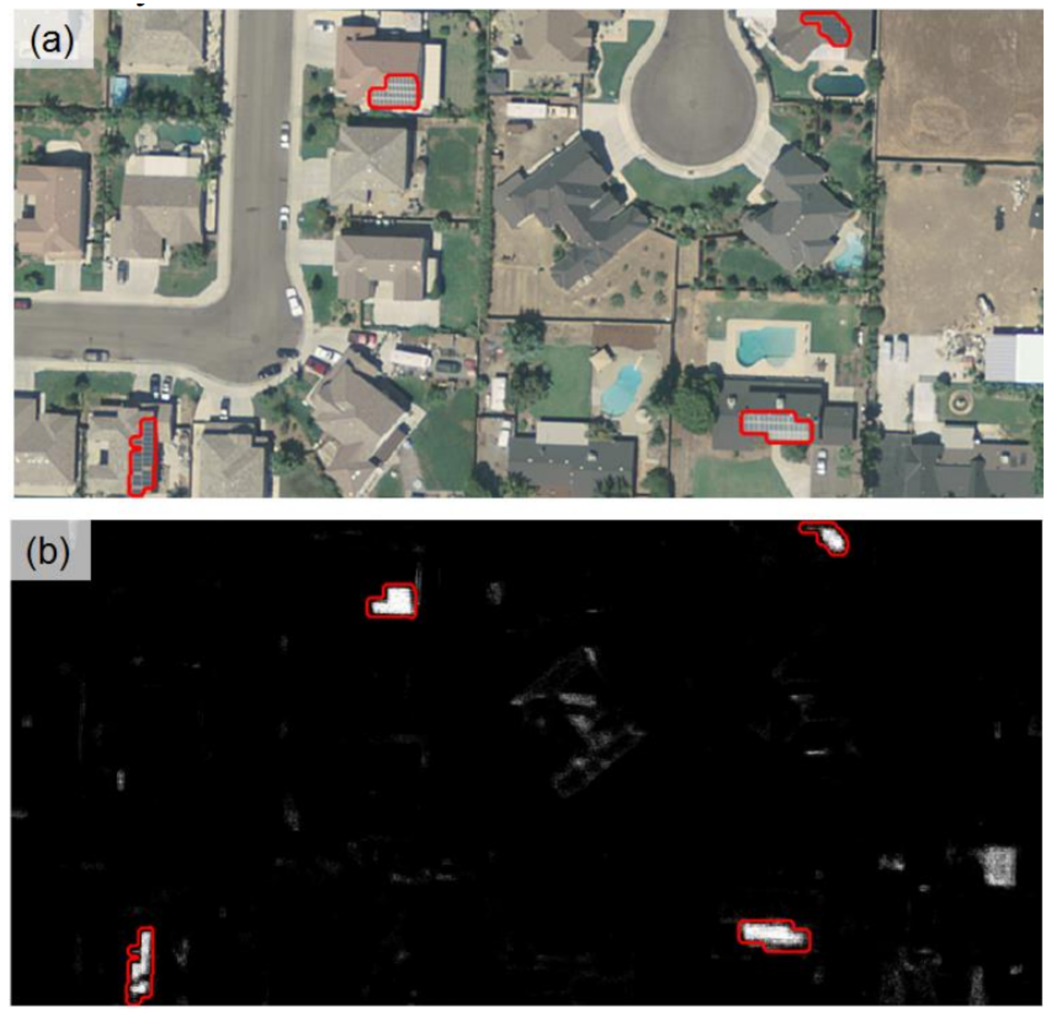
SSPACISS

Introduction

- We want use high resolution color aerial imagery to estimate how many solar Photovoltaic (PV) arrays are installed on rooftops in a city
- We have built a computer vision algorithm that assigns a "confidence value" to each pixel in the aerial imagery, indicating how likely that pixel is to correspond to a PV array
  - Journal paper was just accepted to "Applied Energy", but is currently available on arXiv
  - http://arxiv.org/abs/1607.06029

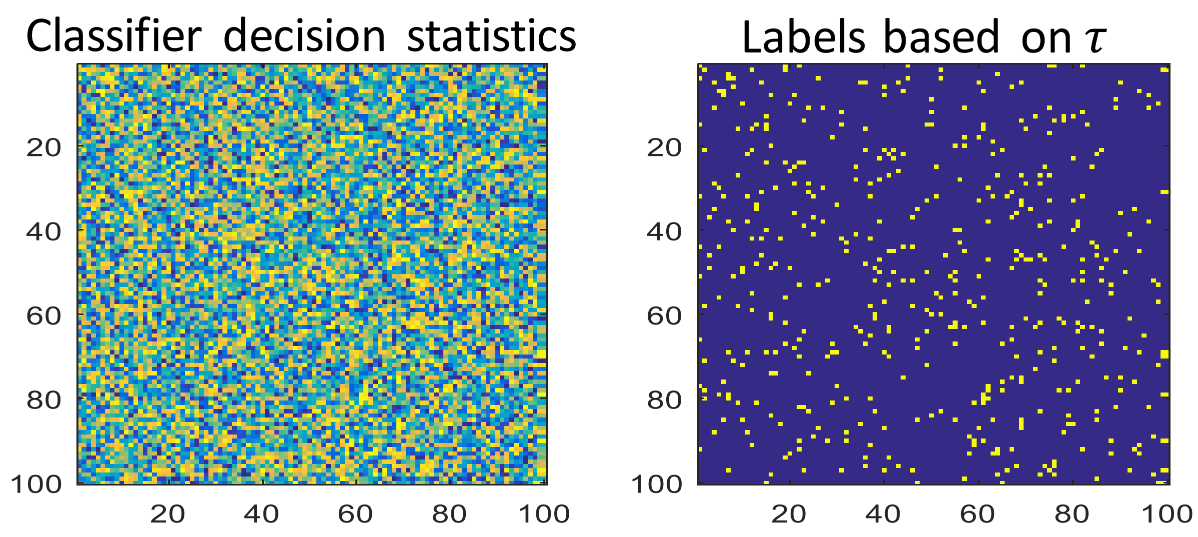
Example algorithm output

- (a) Original imagery
- (b) “confidence map” output
- Each pixel is assigned a confidence value



Recall standard detection theory

- A classifier/detector assigns a decision statistic, d_i, to each pixel in the imagery
- Then, *each individual pixel* is labeled as panel or non-panel, based on applying a threshold, τ, to the d_i values
- To estimate the number of PV array pixels, N_{H1}, we sum all the pixels above the threshold



How do we pick τ?

- Classically, we choose the "minimum error rate" threshold. Assume it is $\tau = 0.6$.
  - This minimizes both types of error (below)

Truth	0	1	1	1	1	1
predictions	1	0	1	1	1	1
d_i	0.8	0.1	0.7	0.8	0.9	0.75

Two types of error

- This will yield the highest *pixel-wise* accuracy i.e., we have maximum accuracy when assigning each individual pixel its label
- BUT, what if we just wanted the total number of labels=1? In this case we don't care about correctly labelling individual pixels. Perhaps there are better ways to guess the total number of panels if we don't care about label correspondence.

Maybe there is a better way?

- The detector, and decision theory for picking τ, are based on an assumption that (i) we need to make a decision (yes/no) for every pixel, and that (ii) we need to identify **where** the PV pixels are
- Here we don't need to know exactly where PV pixels are located, or make a decision for individual pixels. We simply need to estimate N_{H1} for some area
- Perhaps we can estimate N_{H1} a little better by sacrificing knowledge about where the PV array pixels are
  - This is a little bit like Heisenbergs principle, but for detection; you can either know where the panel pixels are, or how many panel pixels there are, but not both!

Overall research goals

- Initial results suggest there are indeed several better ways to estimate N_{H1}
  - So far I have identified several potential improvements
- This semester we will focus on two goals:
  - Apply these approaches to synthetic data to study their properties
  - Apply them to real aerial imagery data, and demonstrate they outperform traditional decision theory methods

Three methods we will study

- We will investigate three methods for improving estimates of N_{H1}
  - “Prior method”, “Posterior method”, and “Error correction method”
- I briefly describe each of these methods next

Method #1 – "Prior Method"

- This is the simplest method
- Estimate the prior distribution over panels, $P[H1]$ and non-panels $P[H0]$ based on training data.
- Let the total number of decision statistics (i.e., scanned pixels) be N
- Then $N_{H1} \cong P[H1] * N$

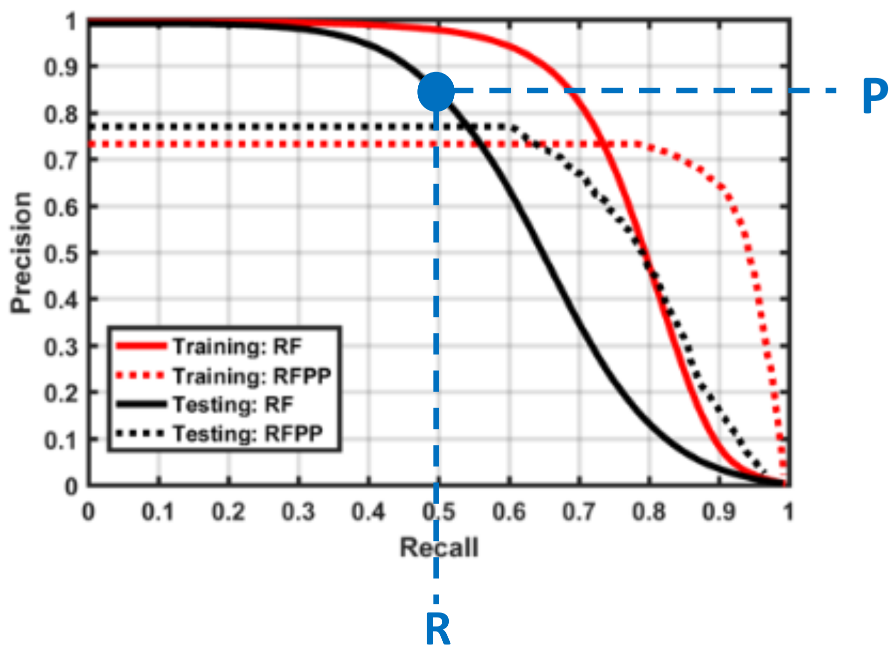
- If the total area that is being scanned is very large, we might expect that this is a pretty good estimator.
- If the total area being scanned is very small, then this estimator might be pretty bad

Method #2 – "Posterior method"

- Let $p_i = \Pr(H1 | d_i) = \frac{\Pr(d_i|H1)}{\Pr(d_i)} = \frac{\Pr(d_i|H1)}{\Pr(d_i|H1)\Pr(H1)+\Pr(d_i|H0)\Pr(H)}$
- Then the probability of a given pixel to have label $l_i \in \{0,1\}$ is a Bernoulli trial
- $\Pr(l_i|d_i) = Bernoulli(p_i)$
- $E[l_i] = p_i$
- Now we want to do this for all pixels, to get expected number of H1 observations, N_{H1}
- $N_{H1} = E[\sum_i l_i] = \sum_i E[l_i] = \sum_i p_i$
- If you are sure you know the statistics of your detector (e.g., $\mathrm{P(d_i|H1), P(H1)}$, etc) then this is the best approach. The problem is that, frequently, the statistics of the detector on the training data and the testing data are not quite the same

Method #3

- PR curves are analogous to ROC curves (look it up)
- Find the minimum error operating point on the PR curve, based on classifier training data.
  - This will provide a confidence threshold, τ
- Use τ to threshold pixel confidence values, and then sum to obtain N_{H1}
- Based on the PR curve we know how many errors the classifier makes (on average). Now we can correct for those errors:
- $\widehat{N}_{H1} = \frac{(P * N_{H1})}{R}$



The precision and recall corresponding to the minimum error operating point

Timeline

- October 7th – Finish synthetic data experiments (see next slide)
- November 1st - Real data experiments
- November 14th – Poster presentation
  - Prepare all figures for poster presentation
- December 1st – First draft of 4-page conference paper

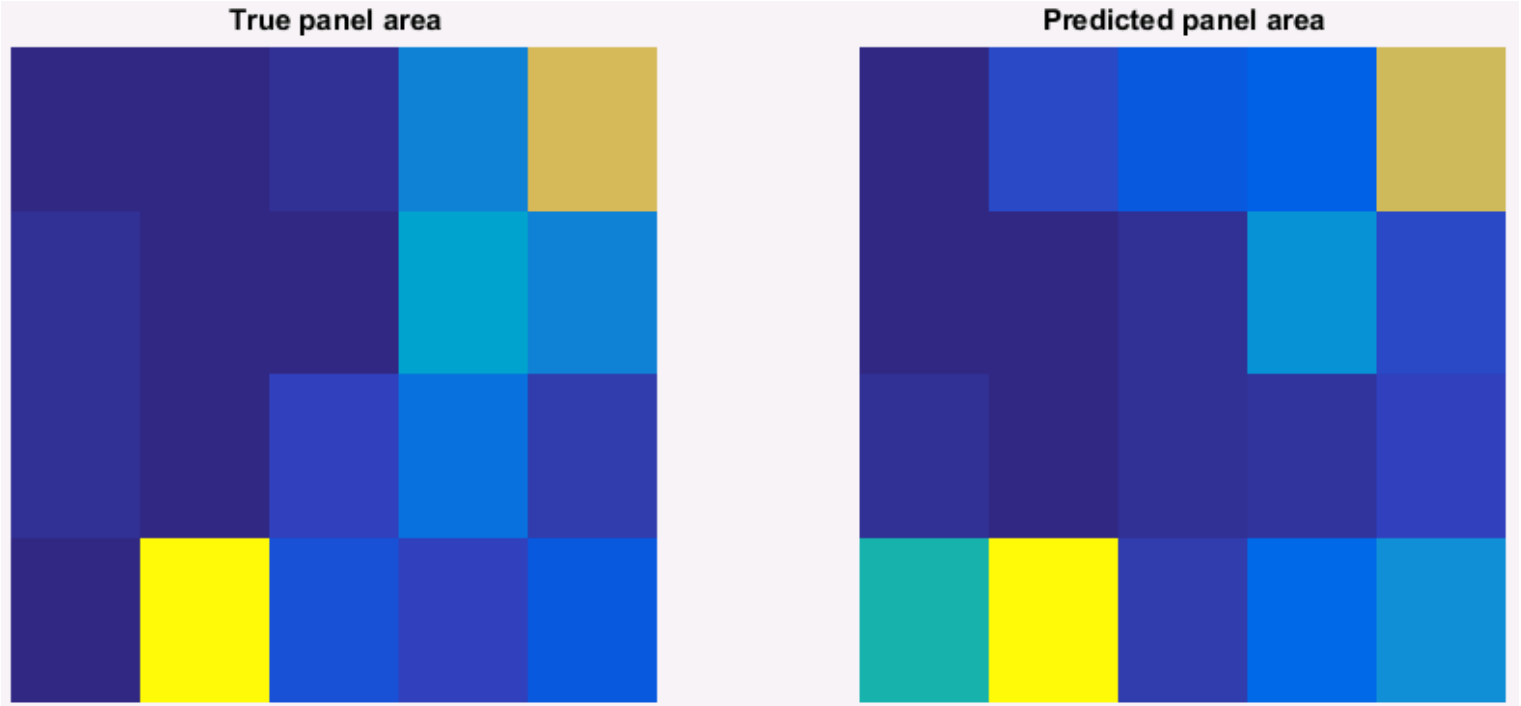
Synthetic data experiments

- Use known synthetic H1/H0 distributions
- Let N_{train} and N_{test} be the number of training and testing pixels.
- For each experiment below, create six testing datasets
  - $N_{test} \in \{500{,}1000{,}2000\ 4000{,}8000{,}16000\}$
- Experiment 1: Assume training/ testing distributions identical.
  - Compute error in N_{H1} for the three new methods, and the classical method. Measure $\mathrm{Error} = \ RMSE/N_{test}$
    - Repeat five times for each value of N_{test} and average the $Error$ values
- Experiment 2: Change priors on the testing data, and repeat experiment 1
  - Experiment 2a: See if you can estimate the new priors of the testing data automatically
- Experiment 3: Change means of $P(d|H1)$ and $P(d|H0)$ on the testing data, and repeat experiment 1
  - Experiment 3a: See if you can estimate the new mean values of the testing data automatically

End

Main result for next PV paper

- Show a “heat map” of panel area over a city, at varying levels of resolution
- Below I show Fresno test data (not in spatial order)
  - Pearson Correlation Coefficient: 0.88



Again, Fresno training data

– Pearson Correlation Coefficient: 0.88