Emily Binet Royall
NoMM Spring 2014

Viterbi NYC

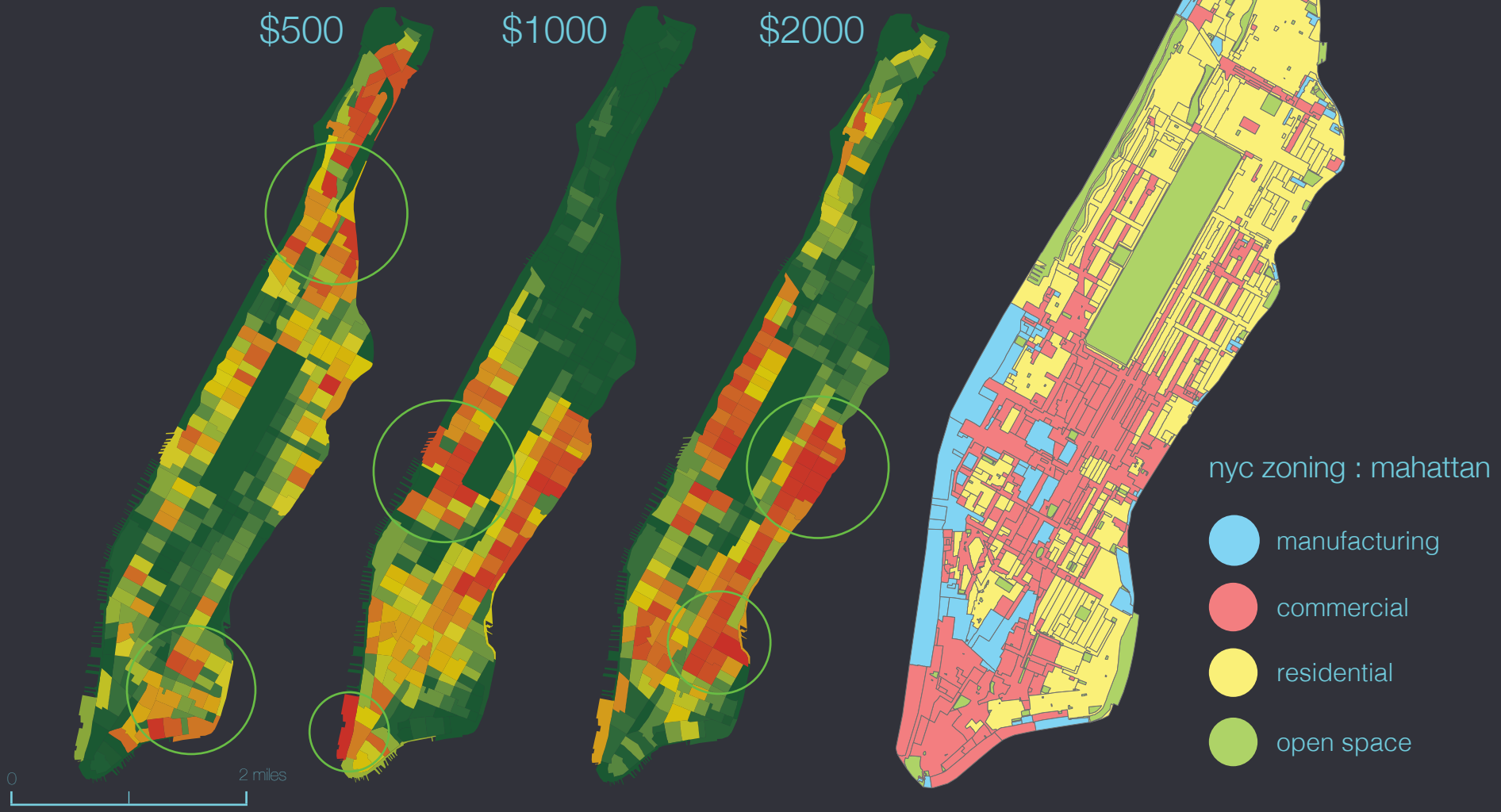
identifying hidden spatial states of rental markets in Manhattan

problem + precedent

Can we use dynamic programming to uncover hidden attributes of cities that give rise to observable patterns? What kind of implications might this have for urban design and policy?

This project uses the Viterbi Algorithm in an Hidden Markov Model (HMM) framework to uncover hidden “spatial states” or boundaries giving rise to observed rental patterns in Manhattan. Do the hidden states uncovered by the model agree with existing zoning policy?

Prior work: L.E. Baum and T. Petrie (1966), Rabiner,
“A tutorial on hidden Markov models and selected applications in speech recognition” (1989), Huang & Kennedy,
“Uncovering Hidden Patterns by HMM.”



concept

HMM model elements:

$\lambda = \{Aij, B, \pi\}$

Given λ, adjust model parameters to maximize P(O | λ)

observed signal (rental prices)

p (emission)

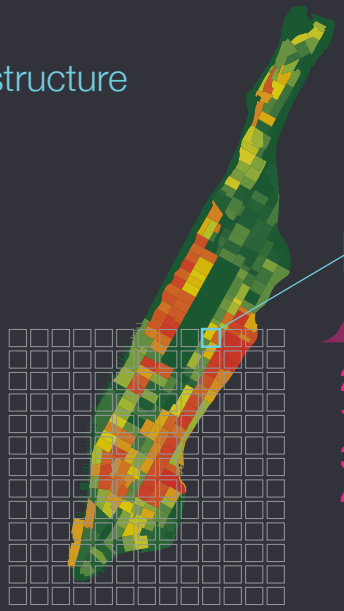
p (transition)

STATE 1 → STATE 2 → STATE 3

assumption: spatial states underly rental prices

hidden "grammar" (spatial states)

data structure

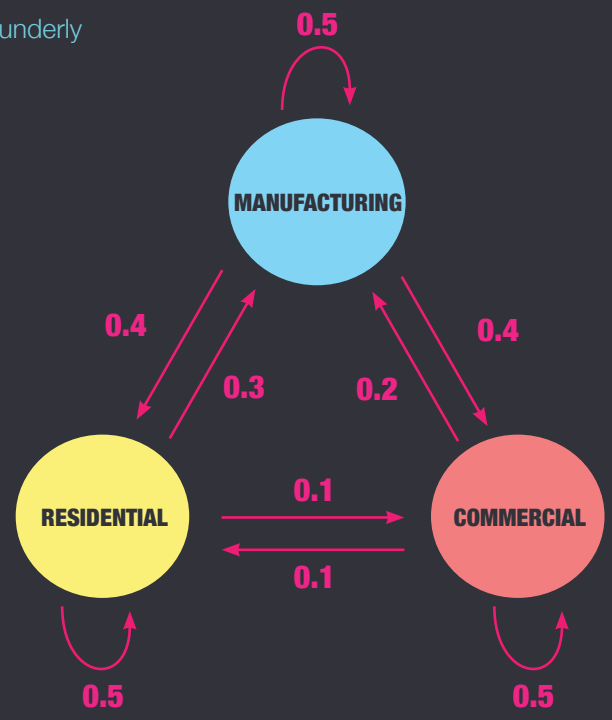


Gross Rent Categories
$ 100-200
$ 240-400
$ 450-600
$ 650-800
$ 900-2000

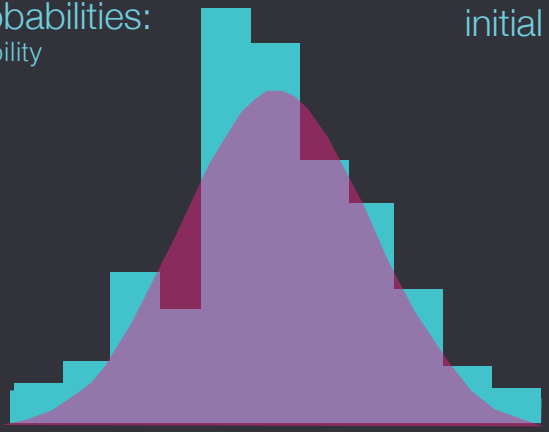
25%
10%
30%
45%

vector: frequency of # units within each rent category.

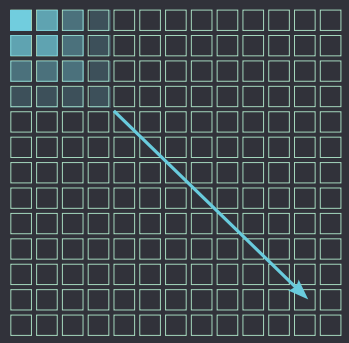
transition probabilities:



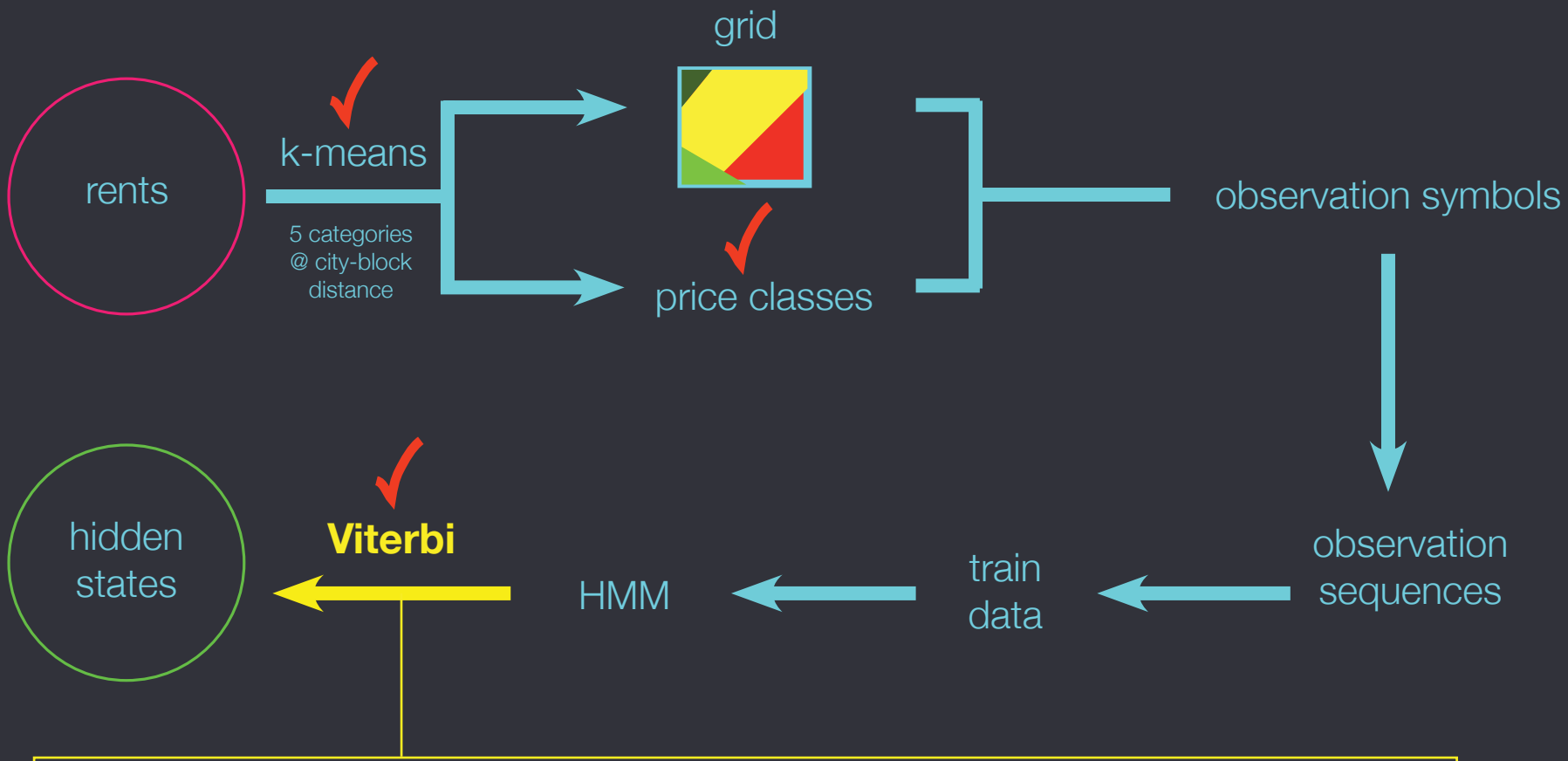
emission probabilities:
gaussian probability density function



initial probabilities:



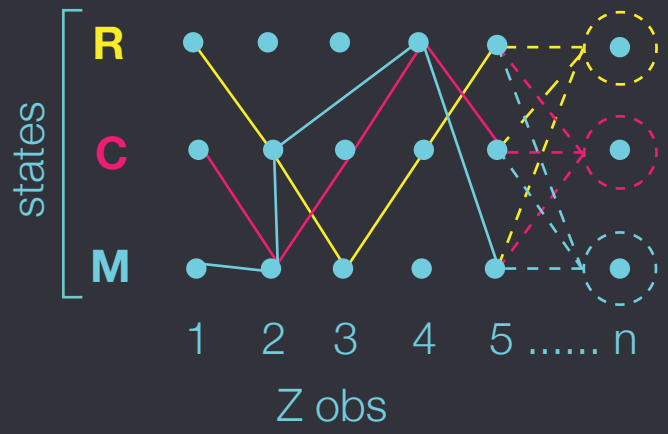
process



Viterbi algorithm: finds the most likely sequence of hidden states (viterbi path) that produces a sequence of observed events in an HMM.

pseudocode

1. Assume emission & transition probabilities
2. Goal:
   find most likely sequence of states $Z^* = \text{argmax } P$ (hidden states z | given obs x)
3. Knowing:
   if $f(a) \geq 0$ and $g(a,b) \geq 0$,
   then $\max_{a,b} f(a)\, g(a,b) = \max_a [\, f(a) \max_b g(a,b) \,]$
4. Begin:
   $\text{argmax}_{z1\text{-}n}\, p(z \mid x) = \text{argmax}_{z1\text{-}n}\, p(z,x)$ (joint distribution)
5. U = max. Find recursion for:
   $\max U_k(z_k) = \max p\,(z_{1:k}, x_{1:k})$ }expand
   $\max z_{1:k-1} = \max p\,(x_k \mid z_k) * p(z_k \mid z_{k-1}) * (z_{1:k-1} \mid x_{1:k-1})$
   emission transition previous
6. Distribute max & found recursion:
   $\max z_{1:k-1} = \max p\,(x_k \mid z_k) * p(z_k \mid z_{k-1}) * \max p(z_{1:k-1} \mid x_{1:k-1})$
   $U_k(z_k) = \max p\,(x_k \mid z_k) * p(z_k \mid z_{k-1}) * U_{k-1}(z_{1:k-1} \mid x_{1:k-1})$ for k = 2....n



7. Keep track of max sequence in each step & compute for the next
8. Keep track of maximizing path that terminates at state.
9. Max for path Z can be found by appending path to one of previous paths.

output + evaluation

$$\max z_{1:k-1} = \max p(x_k \mid z_k) * p(z_k \mid z_{k-1}) * (z_{1:k-1} \mid x_{1:k-1})$$

```
for z in states:
    #Recursion, gives max stored in "prob".
    (prob, state) = max((emisprob[z][obs[t]] * transprob[z0][z] * U[t-1][z0], z0) for z0 in states)
    #update U as prob
    U[t][z] = prob
    #update newpath. Keep track of max seq path at each step
    newpath[z] = path[state] + [z]
#throw away old path
path = newpath
```

```
Last login: Sun May 18 21:23:21 on ttys000
/var/folders/gx/08svkf3j1sq8ms1p0rh12x7r0000gn/T/Cleanup\ At\ Startup/EBR_Viterb
i_2-422155418.293.py.command ; exit;
Emilys-MacBook-Pro:~ eroyall$ /var/folders/gx/08svkf3j1sq8ms1p0rh12x7r0000gn/T/C
leanup\ At\ Startup/EBR_Viterbi_2-422155418.293.py.command ; exit;
        0     1     2     3     4
Resid: 0.111 0.020 0.006 0.002 0.000
Manuf: 0.060 0.033 0.014 0.006 0.001
Comme: 0.174 0.026 0.011 0.002 0.001

(0.001363759278, ['Commercial', 'Manufacturing', 'Manufacturing', 'Manufacturing
', 'Commercial'])
logout

[Process completed]
```

Output:

-table of max values for each state & observation category

-sequence of corresponding hidden states

Evaluation:

-Algorithm works, but real data needs to be handled.

future directions

1. Get the algorithm to handle real data sets:

    a. symbolize tract data using grid method
        -"alternate advancing technique" (Zhang, C.: Generalized Markov Chain Approach for conditional simulation of categorical variables form grid samples (2006).
    b. import the observation sequence where each obs represents a tract with a vector = price frequency distribution

2. Objective approach to calculating transmission & emission probabilities:

    a. calculate emission probabilities for each tract (run rand.py for the size of tract dataset)
    b. create a program that would move from cell to cell and record state transition.

3. GIS or Grasshopper? Might be possible to write a python script for GIS.
    a. alternatively, a grasshopper plug-in.

4. Repeat for time-sequence data: Social Explorer & ACS