Texture
Motivating applications

Hole-filling

(HW4) Image extrapolation
Approach: model image statistics

bad statistical model

good statistical model
Underlying assumption: statistical stationarity

Stationary image statistics

Non-stationary image statistics

“Texture” = image with stationary statistics
Image statistics

How do we capture the “right” level of structure?
Make use of probability models
Skin detection in images

Figure 3: Examples of skin detections. For each pair, the original image is shown above and the detected skin pixels are shown below.

We tested the performance of the skin classifier as the amount of training data was increased, using the histogram model. We divided the skin and non-skin images in the training set into chunks containing approximately 2.5 million skin pixels and 28 million non-skin pixels. On each iteration we added one such chunk of new skin and non-skin pixels to the evolving training set. A ROC curve was computed at each iteration showing the classifier performance on the partial training set as well as on the full test set. The results are shown in Figure 5. As more data is added, performance on the training set decreases because the overlap between skin and non-skin data increases. Performance on the test set improves because the test and training distributions become more similar as the amount of training data increases. Performance on both training and test sets converges relatively quickly. There is little change in either after about 8 iterations. This ROC curve convergence guided our data collection process. During this research, we added photos selected at 9
Given a RGB pixel ‘x’, what is the “probability” that it came from skin or background?
Color histogram of the “world”

(a) 2-D rendering of 3-D histogram model viewed along the green-magenta axis.

Bin of size 8 (32 discrete values per axis)
Size of bin proportional to count
Learned from 2 billion pixels from online images
(a) 2-D rendering of 3-D histogram model viewed along the green-magenta axis.

(b) Surface plot of the marginal density formed by integrating along the viewing direction in (a).

Bin of size 8 (32 discrete values per axis)
Size of bin proportional to count
Learned from 2 billion pixels from online images
(a) Contour plot for skin model, marginalized along the green-magenta axis.

(b) Contour plot for skin model, marginalized along the gray axis.

(c) Contour plot for non-skin model, marginalized along the green-magenta axis.

(d) Contour plot for non-skin model, marginalized along the gray axis.

Color histogram of skin pixels

Color histogram of nonskin pixels
Figure 3: Examples of skin detections. For each pair, the original image is shown above and the detected skin pixels are shown below. Would be 1.0 for a perfect detector.

We tested the performance of the skin classifier as the amount of training data was increased, using the histogram model. We divided the skin and non-skin images in the training set into chunks containing approximately 2.5 million skin pixels and 28 million non-skin pixels. On each iteration we added one such chunk of new skin and non-skin pixels to the evolving training set.

A ROC curve was computed at each iteration showing the classifier performance on the partial training set as well as on the full test set. The results are shown in Figure 5. As more data is added, performance on the training set decreases because the overlap between skin and non-skin data increases. Performance on the test set improves because the test and training distributions become more similar as the amount of training data increases. Performance on both training and test sets converges relatively quickly. There is little change in either after about 8 iterations. This ROC curvevergence guided our data collection process. During this research, we added photos selected at...
Bayesian image matting
observations  posterior probabilities
blended (composited) image
Figure 3: Examples of skin detections. For each pair, the original image is shown above and the detected skin pixels are shown below.

We tested the performance of the skin classifier as the amount of training data was increased, using the histogram model. We divided the skin and non-skin images in the training set into chunks containing approximately 2.5 million skin pixels and 28 million non-skin pixels. On each iteration we added one such chunk of new skin and non-skin pixels to the evolving training set. A ROC curve was computed at each iteration showing the classifier performance on the partial training set as well as on the full test set. The results are shown in Figure 5. As more data is added, performance on the training set decreases because the overlap between skin and non-skin data increases. Performance on the test set improves because the test and training distributions become more similar as the amount of training data increases. Performance on both training and test sets converges relatively quickly. There is little change in either after about 8 iterations. This ROC curve convergence guided our data collection process. During this research, we added photos selected at
**Probability review**

- \( P(R_1, R_2) = \)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \frac{2}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
</tr>
<tr>
<td>2</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{2}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
</tr>
<tr>
<td>3</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{2}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
</tr>
<tr>
<td>4</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{2}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
</tr>
<tr>
<td>5</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{2}{42} )</td>
<td>( \frac{1}{42} )</td>
</tr>
<tr>
<td>6</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{1}{42} )</td>
<td>( \frac{2}{42} )</td>
</tr>
</tbody>
</table>

What’s \( P(R_1 = i) \) for \( i = 1..6 \)?

What’s \( P(R_2 = j) \) for \( j = 1..6 \)?

The marginals \( P(R_1), P(R_2) \) look fair, but the joint \( P(R_1, R_2) \) is clearly biased.
Conditional Probability Tables (CPT)

- \( P(R1|R2) = \frac{P(R1,R2)}{P(R2)} \)

<table>
<thead>
<tr>
<th>R1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>2</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>3</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>4</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>5</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
</tr>
<tr>
<td>6</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
</tr>
</tbody>
</table>

Columns sum to 1

What’s \( P(R1=x|R2=y) \) for \( x = 1..6, y=1..6 \)?

Does the CPT contain the same information as the joint?

What if we knew both the \( P(R1|R2) \) and \( P(R2) \) tables?
Independence

R1 and R2 are independent iff \( P(R1,R2) = P(R1)P(R2) \)

Equivalent “intuitive” definition: R1 and R2 are independent iff \( P(R1|R2) = P(R1) \)

We can check by computing marginals or conditionals from tables
A thought experiment: modeling words

- \( P(\text{word}|\text{word\_prev}) = \)

<table>
<thead>
<tr>
<th>word_prev</th>
<th>run</th>
<th>walk</th>
<th>l</th>
<th>salt</th>
<th>to</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>run</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>walk</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>l</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>salt</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>to</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
</tr>
<tr>
<td>...</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
</tr>
</tbody>
</table>

What would be reasonable CPT values?
Conditional independence

\[
P(word1, word2, word3)
\]

Intuitively, knowing word1 doesn't affect predictions of word3 if we know word2

\[
P(word3|word1, word2) = P(word3|word2)
\]
How much should we condition on?

is ?

my car is ?

I walked to campus because my car is ?

What are good predictions of ?
How do we model long-range structure?
Nth order markov models

$N = \text{number of words that we condition on, to make a prediction}$

$P(\text{word}|\text{history}) = P(\text{word}|\text{word}_{\text{prev1}},\text{word}_{\text{prev2}})$

<table>
<thead>
<tr>
<th>word</th>
<th>l, run</th>
<th>you, walk</th>
<th>l,l</th>
<th>you,yo u</th>
<th>to, you</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>l</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>you</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>walk</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>run</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>...</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
</tr>
<tr>
<td>...</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
</tr>
</tbody>
</table>
Shannon’s N-gram models

Shannon (1948) proposed a way to generate English-looking text using $N$-grams

Assume a Markov model
Use large text to compute CPT of each word given $N-1$ previous words
Starting from a seed repeatedly sample the conditional probabilities to generate new words
One can use letters instead of whole words
Synthesis

I am tired of walking.
Nth order markov chains

- **Markov property**:  $p(\text{word} \mid \text{history}) = p(\text{word} \mid N \text{ past words})$
- **Stationarity**: the stochastic model is the same regardless of position

These statistical assumptions allow us to capture sentence statistics with a single CPT

<table>
<thead>
<tr>
<th>word</th>
<th>I, run</th>
<th>you, walk</th>
<th>I, l</th>
<th>you, you</th>
<th>to, you</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>you</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>walk</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>run</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>...</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
</tr>
<tr>
<td>...</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
</tr>
</tbody>
</table>
Results (using CPTs from alt.singles corpus):

“As I've commented before, really relating to someone involves standing next to impossible.”

“One morning I shot an elephant in my arms and kissed him.”

“I spent an interesting evening recently with a grain of salt.”
“People, having a much larger number of varieties, and are very different from what one can find in Chinatowns across the country (things like pork buns, steamed dumplings, etc.) They can be cheap, being sold for around 30 to 75 cents apiece (depending on size), are generally not greasy, can be adequately explained by stupidity. Singles have felt insecure since we came down from the Conservative world at large. But Chuqui is the way it happened and the prices are VERY reasonable.”
Pixel synthesis
Statistical modeling of texture

- Assume stochastic model of texture (*Markov Random Field* instead of *Markov Chain*)
- *Markov property*: \( p(\text{pixel} \mid \text{rest of image}) = p(\text{pixel} \mid \text{N} \times \text{N} \text{ neighborhood}) \)
- *Stationarity*: the stochastic model is the same regardless of position
Nth order markov models

- $P(\text{pixel} | \text{image}) = P(\text{pixel} | \text{neighborhood})$

<table>
<thead>
<tr>
<th>Pixel</th>
<th>1,1...1</th>
<th>1,1...2</th>
<th>...</th>
<th>....</th>
<th>...</th>
<th>256,256,...256</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1,...1</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>1,1,...2</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>...</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>...</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
<td>1/7</td>
</tr>
<tr>
<td>...</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
<td>1/7</td>
</tr>
<tr>
<td>256..256</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>1/7</td>
<td>2/7</td>
</tr>
</tbody>
</table>

This table is hopeless to explicitly create and store
Implicit CPTs

Find matches for current neighborhood in input image
Build column of CPT by counting the fraction of times we see a pixel value
Sample from this column

But this is equivalent to simply picking a match at random and outputting its center pixel
Finding matches

- Sum of squared differences (SSD)
Finding matches

• Sum of squared differences (SSD)
  – *Gaussian-weighted* to make sure closer neighbors are in better agreement

\[
\| \ast \left( \begin{array}{cc}
\begin{array}{cc}
\text{gray} & \text{gray} \\
\text{gray} & \text{gray}
\end{array}
\end{array} \right) - \left( \begin{array}{cc}
\begin{array}{cc}
\text{blue} & \text{purple} \\
\text{purple} & \text{blue}
\end{array}
\end{array} \right) \|_2^2
\]
Varying Window Size
Varying Window Size

Increasing window size
Homage to Shannon
Hole filling

• Hole filling: growing is in “onion skin” order
  – Within each “layer”, pixels with most neighbors are synthesized first
  – Normalize error by the number of known pixels
  – If no close match can be found, the pixel is not synthesized until the end
Summary

• The Efros & Leung algorithm
  – Very simple
  – Surprisingly good results
  – …but very slow