The Cheetah problem

Nuno Vasconcelos
ECE 271A
Cheetah

• statistical learning only makes sense when you try it on data
• we will test what we learn on a image processing problem
  – given the cheetah image, can we teach a computer to segment it into object and foreground?
  – the question will be answered with different techniques, typically one problem per week
• first problem this week
  – brief introduction to image representation (features) and other pre-processing steps
Image representation

- we will use the **discrete cosine transform (DCT)**
  - think of it as a Fourier Transform, but *real*
  - maps an array of pixels (image block) into an **array of frequency coefficients**
  - for block \(x(i,j)\)

\[
T(k_1, k_2) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} 4x(i, j) \cos \left( \frac{\pi k_1}{2N} (2i + 1) \right) \cos \left( \frac{\pi k_2}{2N} (2j + 1) \right)
\]

- each coefficient is a **projection** onto a basis function
- basis functions are **2D sinusoids** of different frequencies
- \(T(k_1, k_2)\) captures image information on the **frequency band**

\[
\left[ \frac{\pi k_1}{2N}, \frac{\pi k_1}{2N} + 1 \right] \times \left[ \frac{\pi k_2}{2N}, \frac{\pi k_2}{2N} + 1 \right]
\]
In a picture

- we will use blocks of 8 x 8 pixels
- the DCT basis functions are
- 1st function is constant, 1st coefficient is the block mean, not very interesting (depends on illumination etc.)
- there is a MATLAB function – dct2(.) – that computes the DCT coefficients
In a picture

- coefficients have a natural order by frequency
- it is called the zig-zag pattern
- allows us to transform the 2D array of coefficients into a vector
- this vector has 64 features, i.e. is a point on a 64D space
- we will make available a file with this zig-zag pattern
Image representation

- **Image**
- **8x8 blocks**
- **8x8 DCT**
- **Discrete Cosine Transform**
- **Bag of DCT vectors**
- **$\mathbb{R}^{64}$**
- **64**
- **Number of blocks**
Features

• 64D is a lot, we will see later in the course how to pick good features
• for now we will use a single feature

\[ X = \text{location of the coefficient of 2}\text{nd largest magnitude} \]

• e.g. for vector \((100, 12, -32, -53, 14)\) we have \(X = 4\)
• **rationale:** 1\text{st} coefficient is always the largest, but not very informative, 2\text{nd} largest gives the dominant frequency band
• note that \(X\) is now a scalar feature, we can estimate all CCDs with histograms
Classifier

- **Training:**
  - break training images into 8x8 blocks
  - for each block
    - compute DCT,
    - order coefficients with zig-zag scan
    - pick position of 2nd largest magnitude as the feature value
  - note: we will give you this!
  - the collection of all such positions is the training set
  - from training set estimate $P_{X|Y}(x|\text{cheetah})$, $P_{X|Y}(x|\text{background})$, using histograms, and $P_Y(\text{cheetah})$, $P_Y(\text{background})$, using common-sense
Classifier

- **classification:**
  - break training images into 8x8 blocks
  - for each block
    - compute DCT,
    - order coefficients with zig-zag scan
    - pick position of 2\textsuperscript{nd} largest magnitude as the feature value
  - use BDR to find class Y for each block
  - create a binary mask with 1’s for foreground blocks and 0’s for background blocks

- note: you’ll have to implement all of this on your own
Remarks

• this is a realistic problem
• the solution WILL NOT BE PERFECT
• there is no unique right answer
• by looking at the resulting segmentation mask, you will know if the results are “decent”
  – holes, noisy, is OK
  – but it should look somewhat like this
Most common problems

• “my segmentation mask is very blocky”
  – during classification, use a sliding window that moves by one pixel at each step
  – this will give you a binary value per pixel (e.g. assign it to the central pixel in the block, or the top left corner) for the segmentation mask

• “I get complete garbage”
  – make sure to always work with doubles in the range [0-1] (this is how the training data was created)
  – after you read the image do
    • \textit{im2double(image)}
    • or \textit{double(image)/255}
Most common problems

• “my probability of error is too high”
  – make sure to use the same histogram binning in all histograms
  – MATLAB let’s you do this easily

• “how do I read an image on MATLAB?”
  – you should be able to figure out the answers to these type of questions on your own
  – MATLAB’s help, tutorials, etc.

• other questions, email Nikhil, but please be gentle on him