Cascades of Support Vector Machines
for Face Detection

Ignas Kukenys
Supervised by
Brendan McCane
Geoff Wyvill

University of Otago, Department of Computer Science, 2008
Contents

- Introduction to face detection
- Apples vs. Oranges: SVM classifiers
- Need for speed 1: approximation
- Need for speed 2: cascades
- Research, results and future work
Face detection? My camera already does that!

- Just a few months ago FUJIFILM released their FinePix Z200fd:

  
  “Couple Timer. Just the two of you? Couple Timer waits until two faces are close together in the frame. Select “Near”, “Close-up” or “Super-Close” depending on the level of intimacy you’re aiming for!”

- Fine print:
  - Depending on the scene conditions, Face Detection performance may vary.
  - Face Detection frames in photos of LCD screens in this page have been simulated.
Researching face detection

- We're nowhere near human vision
  (even more so when it comes to faces)
- We need face detection for recognition and security, search, rich interactions, consumer electronics, tagging photos on Facebook
- And we need it ever faster and more accurate!
Finding faces in a photo

- Unfortunately, no magic :(
  - Take a sub-image at every position, every scale
  - Decide if it's a face or not
- In practice: ~100K different sub-images for a 500x300 photo!!

- To decide about each sub-image, we need a binary classifier..
Enter: Support Vector Machines!

- Universal binary classifier: separates Apples from Oranges
- Put objects into a “good” space and separate them linearly
- Use objects to define the boundary (*support vectors*)
Support Vector Machines

- The decision function is expressed in terms of training data:

  \[ F(z) = \text{sign} \left[ \sum_{i=1}^{N} y_i \alpha_i K(x_i, z) + b \right] \]

  - \( x \) – training data objects (support vectors)
  - \( y \) – labels (±1)
  - \( K(x, z) \) – kernel function for the “good” space
  - \( \alpha \) – weights (product of training)
  - \( b \) – offset (bias)

- Training a Support Vector Machine
  - Finding weights \( \alpha \) for the training objects..
  - So that the separation margin is maximized

- More support vectors = Higher accuracy = Slower decision!
SVM on 2D points

http://miracle.otago.ac.nz/postgrads/ignas/TinySVM/
Problems with “raw” SVMs

- More support vectors = Higher accuracy = Slower decision!
- In practice:
  - Your image training data is 20x20 = 400 dim. vectors (long!)
  - Your $K(x, z) = \exp(||x-z||^2/\sigma)$ (expensive!)
  - You have ~2000 support vectors (a lot!)
- Sure you can evaluate $F(z)$ in a millisecond..
- ..but there's 100K of z's = minutes per image
Need for speed 1: approximation

- It's nice to use training data as support vectors.
- But we could calculate optimal vectors and use fewer of them!
- “It's like finding bananas to mark the boundary!”

- Not a trivial task! Non-linear mapping, etc.
Approximation techniques

- Luckily there are ways to find the approximating vectors
- **Tens of vectors instead of thousands** – but still need a few

- Greedy is faster, optimal is more accurate
Need for speed 2: classifier cascades

- Fact: **99.99%** of queries during detection are **non-objects**!
- Desired strategy:
  - Have an incremental decision function
  - As soon as sure it's a non-object, discard it!
- Set up a series of approx. functions with 1, 2, 3, ... **M** vectors
- Make each function very confident about discarding (**biasing**)

\[ F(z) = \begin{cases} 
\text{False} & \text{if True?}, \\
\text{True?} & \text{if False.}
\end{cases} \]

1 → True? → False → 2 → True? → False → 3 → True? → False → 4 → ... → True!
Classifier cascades

- “Greedy” cascade:
  - Shares the vectors, one new vector per step - FAST!
- Globally optimized cascade:
  - New optimal set of vectors for every step – MORE ACCURATE!

- Our own “hybrid” cascade!
  - Shares the vectors, but optimizes every new pair
  - As complex as greedy, but more accurate from early on
  - In real life, beats the greedy cascade in time taken by 9%!
Classifier cascades: globally optimized

--- Optimized together
Classifier cascades: “greedy”

- - - - - Optimized together
Classifier cascades: hybrid

Optimized together
Research project

- **Goal**: fast and accurate face detection service for online apps
  - Component-based detection (eyes + nose + lips = face)
  - Retraining based on user feedback
  - A step towards face recognition?

- Implementation so far:
  - Eye detector with **88% accuracy**
  - Takes ~**400ms** for an average 500x300 photo
Training data example
Results
Progress & future work

- Training a SVM detector for a chosen object
- Bootstrapping to increase accuracy of a SVM
- Cascades of approximations for speed
- Need more face components: nose, lips?
- Defining a component model of a face, matching it
- Obtaining useful feedback from usage, retraining
Questions? Go bananas!