Million Dollar Machine
Training SVMs on very large data sets

Ignas Kukenys
University of Otago
Department of Computer Science
Graphics & Vision Research Group
2009
PhD student, 2nd year
Thesis title: Support Vector Machines for Human Face Detection

Roadmap in theory:
- Learn how others have used SVMs to detect faces
- Come up with a better, improved approach
- Implement a kick-ass face detector
- Get a degree!
Roadmap in practice:
- Start playing with existing methods
- Find gaps, try to fill them with research
- Bootstrapping, SVM approximation, classifier cascades, ...

Latest problem – big data sets
- How do you deal with large amounts of training data?
Face detection 101

» Is there a face in this photo?
» “But even my point-and-shoot camera does that!” - “Yes, but:”
  » Not accurate enough
  » Not fast enough
  » Nowhere near the performance of human vision

» Probably not the most efficient, but working approach:
  » Take every sensible patch of the target image
  » Decide whether it represents a face or not

» Classification problem in the field of Machine Learning
  » Support Vector Machines can classify well!
Support Vector Machines

- Binary classifier: \( f(x) = \pm 1 \)
- Uses training samples to express the decision function \( f(x) \)
Support Vector Machines

- Binary classifier: $f(x) = \pm 1$
- Uses training samples to express the decision function $f(x)$
Support Vector Machines

- Binary classifier: $f(x) = \pm 1$
- Uses training samples to express the decision function $f(x)$
Support Vector Machines

Formally, we have:

\[ f(x) = \sum_i y_i \alpha_i K(x_i, x) + b \]

Training samples \( x \) with labels \( y = \pm 1 \)

Kernel function \( K(x, z) \) that maps into a high-dimensional space

Weights \( \alpha \) and offset \( b \) are the results of training:

Optimization, Quadratic Programming (QP) problem
The curse of dimensionality

- All would be good, except it is too slow
- Samples $x$ are high-dimensional: $20 \times 20$ pixels = 400 dimensions
- Many dimensions also means a lot of samples of the object
- How many different $20 \times 20$ images represent an eye?
The curse of dimensionality

➔ All would be good, except it is too slow
➔ Samples \( x \) are high-dimensional: \( 20 \times 20 \) pixels = \( 400 \) dimensions
➔ Many dimensions also means a lot of samples of the object

➔ How many different \( 20 \times 20 \) images represent an eye?

➔ “Small linear transformations shouldn't change the class”
  ➔ Shifts, rotation, mirroring, brightness..
  ➔ Easy to go from 1000 samples to 1 million

➔ How do you train on 1M samples if 10K takes days?
SVM as a black box building block

- Summarizes a given set of data
- Chooses a subset of 'most representative' samples
- In practice has a limited capacity

**Instant idea:**
- Split training set into manageable subsets
- Apply SVM to each chunk
- Keep the representative support vectors
- Combine them into the final classifier
Divide & Conquer
Divide & Conquer
Divide & Conquer
Divide & Conquer
Divide & Conquer
Divide & Conquer

Diagram showing the process of dividing a region into smaller parts and conquering them.
Divide & Conquer
Divide & Conquer
Divide & Conquer
Divide & Conquer
Why do we expect this to work?

- Assumptions
  - SVM(A) better than SVM(B), if $A > B$
  - SVM(SVM(A)) worse than SVM(A)
  - SVM(A) better than a random subset of A (of the same size)

- Propositions
  - Incremental: SVM(SVM(A) + B) worse than SVM(A + B)
  - Division: SVM(SVM(A) + SVM(B)) worse than SVM(A + B)
  - Division worse than incremental:
    SVM(SVM(A) + SVM(B)) worse than SVM(SVM(A) + B)
Incremental approach

- Adding next chunk to the existing support vectors

- Suggested improvements
  - Use a fixed SVM size, acts like a buffer for support vectors
  - Only add samples that are misclassified by the latest SVM
Reduction approach

- Divide, train, merge support vectors, repeat
Approximate training tools

- Say you have N samples, only M fit into your SVM

- Incremental approach - works bottom-up:
  - Builds up a set of M samples that are support vectors

- Reduction approach – works top-down:
  - Keeps discarding samples until sets of M can't be reduced further
Million Dollar Machine experiment 1/3

- 2500 original eye patches
- Generate 500K using small transformations (positive samples)
- Select 500K random background patches (negative samples)
- Baseline function: original 2500 + bootstrap over 500K negative

- Incremental approach (SVM size of 5000):
  - Ran for 18 hours
  - Processed 350K of the 1M samples before filling the buffer
  - Resulting in 5000 support vectors
Million Dollar Machine experiment 2/3

- 2500 original eye patches
- Generate 500K using small transformations (positive samples)
- Select 500K random background patches (negative samples)
- Baseline function: original 2500 + bootstrap over 500K negative

- Reduction approach (SVM size of 500):
  - Ran for 10 hours
  - Discarded 957K of the 1M samples
  - SVM on random 5000 of the remaining samples kept 3900 as sv's
Million Dollar Machine experiment 3/3

Receiver Operating Characteristics

![ROC Curve](image-url)
Conclusions and further questions

- Augmenting the data can be worth it:
  - 1M-incremental function 98% vs. original 93% in accuracy

- Suggested approximate approaches can be useful tools
  - We can now study which transformations are 'better'

- If we used a clustering method instead of random subsets?
  - Parallel SVM?