Lecture 05: Principles of sliding window detectors

What we'd like to do?

- Visual scene understanding
- What is in the image and where
- Object categories, identities, properties, activities, relations, ...

Things vs Stuff

- Thing: object with specific shape
- Stuff: material defined by homogeneous or repetitive pattern. Has no specific spatial extent or shape
  ↳ Segmentation method

Recognition tasks

- Image classification: does the image contain an aeroplane?
- Object class detection/localization: where are the aeroplanes if any?
- Object class segmentation: which pixels are part of an aeroplane?
- Panoptic segmentation: Besides object segmentation, also background segmentation

Challenges

- Background clutter: lot of stuff in the background
- Occlusions and truncation: partially seen objects from the camera
- Intraclass variation: a class can have ~~a~~ lot of different versions

↳ object instant recognition: recognizes a specific model of an object not its generic class

≠

category detection: recognizes generic classes
it is harder to perform than instant detection

So why bother?

- Spatial relationships for image understanding and retrieval ("a cat riding a skateboard")
- Visual question and answering: object grasping/tracking
  Collision prevention, face recognition

Sliding window detectors

Problem of background clutter : Solution

- Use sub window :
  - At correct position, no clutter is present
  - Slide window to detect objects
  - Change size of the window to search over scales

Detection by classification

- Basic component : binary classifier
  ↳ sliding window over window using CNNs is too slow
- Detect objects in clutter by searching
  - sliding window : exhaustive search over position and scale
    ↳ in practice, it is possible to use same window size over spatial pyramid
      ↳ more efficient

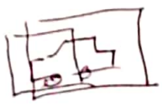
Window (image) classification

- Features usually engineered
- Classifier learned from data

$\boxed{\text{image}} \rightarrow \boxed{\text{feature extraction}} \rightarrow \underbrace{\begin{bmatrix} \vdots \end{bmatrix}}_{X} \rightarrow \boxed{\begin{matrix}\text{Classifier} \\ F(x)\end{matrix}} \rightarrow$ Car / Non Car, $P(C|X) \propto F(x)$

Problems with sliding windows

- Aspect ratio
- Granularity (finite grid)
- Partial occlusions
- Multiple responses →



CVPR 2005, pedestrian detection
(Conference on CV and PR)

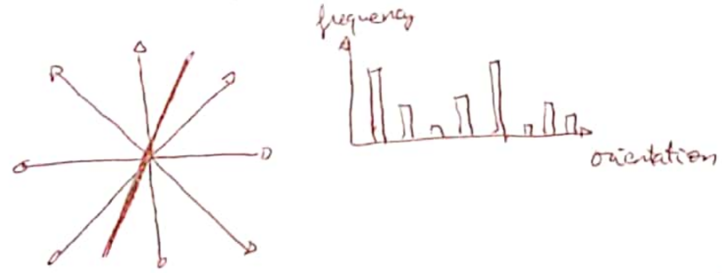
- Objective: detect (localize) standing humans in an image
- Sliding window classifier
- Train a Linear SVM classifier on whether a window contains a standing person or not
- Histogram of Oriented Gradients (HOG) feature
- Although introduced for pedestrian detection, it has been successfully used with many other categories

Window (image) classifier

image → Feature extraction HOG → [ ] x → Classifier SVM → Pedestrian / Non-pedestrian

Feature: HOG

- Tile 64 × 128 pixel window into 8 × 8 pixel cells
- Calculate gradient image (edge detection)
- Each cell represented by by histogram over 8 orientation bins (or sector)



- Adds a second level of overlapping spatial bins re-normalizing orientation histogram over larger spatial area
- Feature dimensions (approx) = 16 × 8 (for tiling) × 8 (orientations) × 4 (blocks) = 4096
- Similarity to CNN
  - CNN learns the filters automatically
  - Sum pooling
  - normalization
- OK job

Linear classifier

- $f(x) = w^T x + b$
- It learns such weights w and bias b that

$$f(x_i) = \begin{cases} \geq 0 & y_i = 1 \\ < 0 & y_i = -1 \end{cases}$$

2D discriminant is a line
3D " is a plane

Linear separability

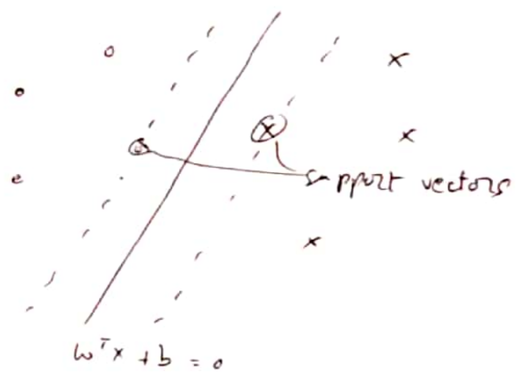
- The points might be linearly separable but with very narrow margin
- The large margin solution might be better, even constraints are violated
- In general there is a trade off between the margin and the number of mistakes on the training data

Support Vector Machine

. It is a way to optimize this trade off
- Find a good trade off between the classification margin and the missclassified training points

$f(x) = w^T x + b$

$$\min_{w \in \mathbb{R}^d} \|w\|^2 + C \sum_i^N \max(0, 1 - y_i f(x_i))$$



Learned Model using HOG detector

- Positive weights → evidence it is a person
- Negative weights → evidence it is not a person

} Average over positive training data

$f(x) = w^T x + b$

→ comes from 8x8

What do negative weights mean?

$$w^T x > 0$$
$$(w_+ - w_-) x > 0$$
$$w_+ x > w_- x$$

- Complete system compete pedestrian / pillar / doorway models
- Discriminative models come with own background model
- Avoid detections on doorways by penalizing vertical edges

Pedestrian model > Pedestrian background model

Problems when training a sliding window detector

- Inherently asymmetric problem: many more "non-object" than "objects"
- Classifier needs to have very low false positive rate
- "Non-object" category is very complex and needs a lot of data

Optimizing approach: Bootstrapping

- 1. Pick a negative training set at random
- 2. Train ~~the~~ classifier
- 3. Run on training data
- 4. Add false positives to training set
- 5. Repeat from step 2. (Retraining)

Data augmentation: With your available data you

- Flip
- Rotate
- Scale
- Crop
- Translate
- Apply Gaussian noise

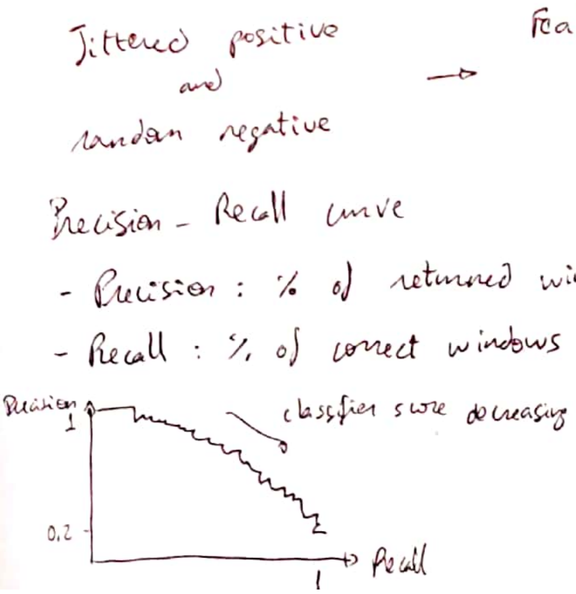
to the images ⇒ Jittered positive ⇐ ~~Jittered~~

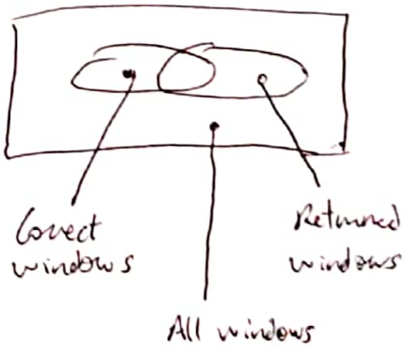
Window (image) first stage classification

Jittered positive and random negative → Feature extraction HOG → $[\vdots]$ x → linear SVM classifier $f(x) = w^T x + b$

Precision - Recall curve

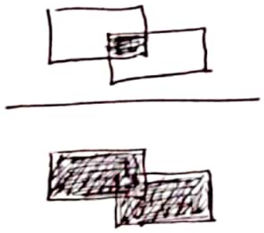
- Precision: % of returned window that are correct
- Recall: % of correct windows that are returned



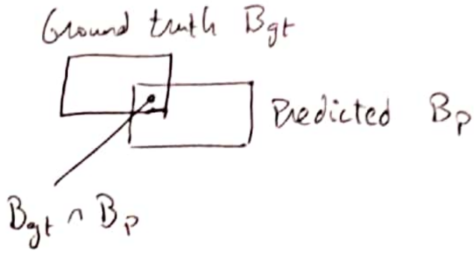


Evaluating the detected bounding boxes

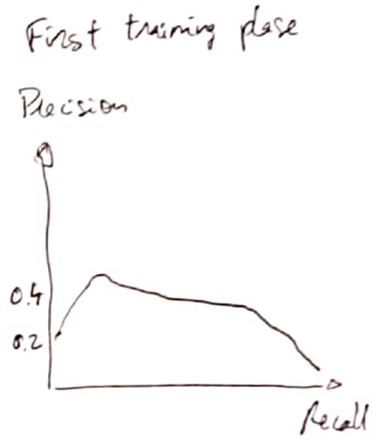
- Area of overlap (AO) measure
- Correct detection if intersection over union larger than threshold



> threshold (50%)



$$AO(B_{gt}, B_p) = \frac{|B_{gt} \cap B_p|}{|B_{gt} \cup B_p|}$$



Second training phase → Retrain using better data

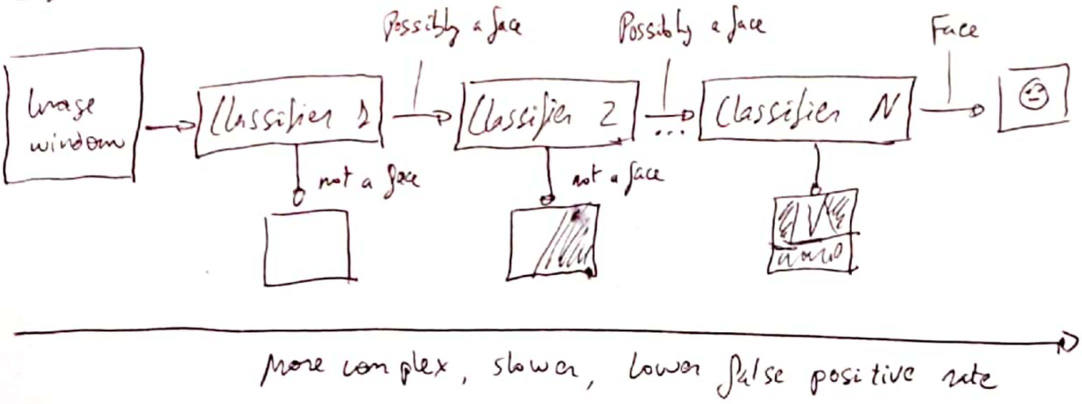
- Find high scoring false positive detections
- Use them as hard negatives for next training round
- Cost = # training image · inference time per image

~~First training phase~~

Accelerating sliding window search

- Sliding window search is slow since some many windows are needed
- m×n × scale = 100 000 windows for 320×240 image
- Most windows are negative
- It is possible to speed up the search

Cascade classification



In the end, you spend much less computational cost

- Slow and expensive classifiers only applied to a few windows
- Controlling complexity vs speed: numbers of features, numbers of parts, ...

Detection proposal: Hierarchically clustering superpixels

- Larger homogeneous area is considered a unit in the image → superpixel
- Hierarchical segmentation: start with small and merge using cues

Produces roughly 2000 regions per image with 95% of hitting relevant objects

Things to remember

- Detection by sliding window classification → Concept and components
- Multiple scales (and aspect ratios) to detect objects of different size
- Importance of hard negative mining (due to class imbalance)
- Cascade detectors → speed up inference
- Speed up training and inference by selecting sub-set of windows only