CMPSCI 670: Computer Vision

Object detection *continued …*

University of Massachusetts, Amherst
November 10, 2014

Instructor: Subhransu Maji

Administrivia

- No class on Wednesday
  - Following Tuesday's schedule this Wednesday
- Office hours this week are at Thursday 3:45 - 4:45 pm

Today's lecture

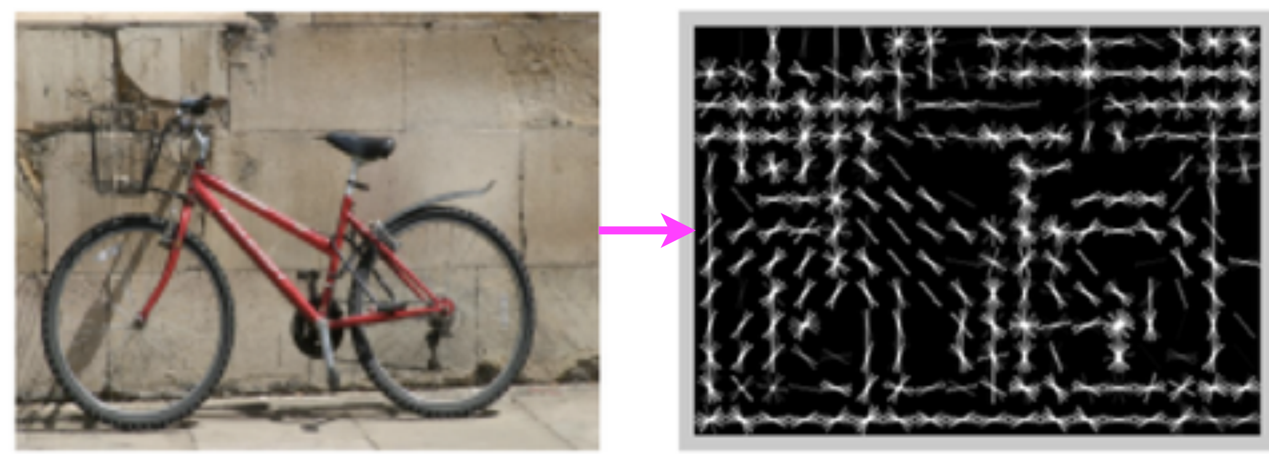
- Object detection
  - Speeding up it up
  - Making it more accurate
- Lecture overview
  - Recap last lecture (HOG, template matching, training)
  - Issues with the sliding window detector
  - Selective search using region proposals
  - Fast kernel SVM classifiers

Detection = repeated classification

face or not?

Histograms of oriented gradients (HOG)

- Introduce invariance
  - Bias / gain / nonlinear transformations
    - bias: gradients / gain: local normalization
    - nonlinearity: clamping magnitude, orientations
  - Small deformations
    - spatial subsampling
    - local “bag” models

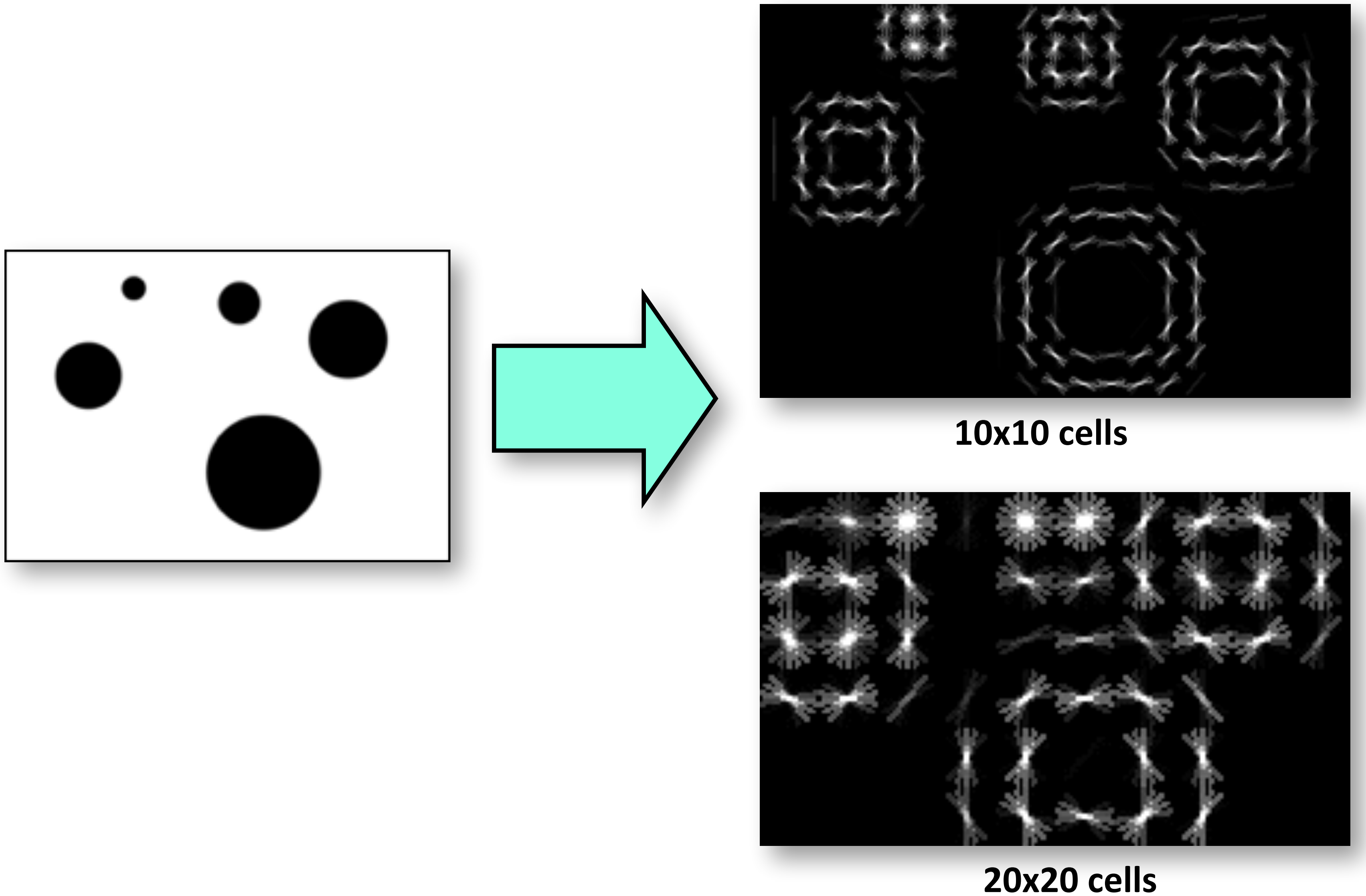


- References

  - “Histograms of oriented gradients for human detection.” N. Dalal and B. Triggs, CVPR 2005.
  - “Finding people in images and videos.” N. Dalal, Ph.D. Thesis, Institut National Polytechnique de Grenoble, 2006.

Histograms of oriented gradients (HOG)

- Partition image into blocks at multiple scales and compute histogram of gradient orientations in each block

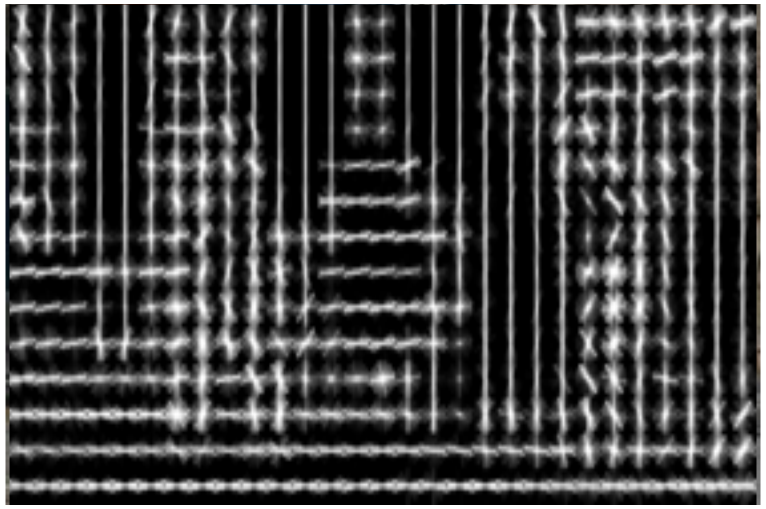


N. Dalal and B. Triggs, [Histograms of Oriented Gradients for Human Detection](), CVPR 2005

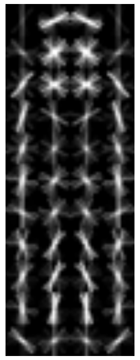
Image credit: N. Snavely

Template matching with HOG

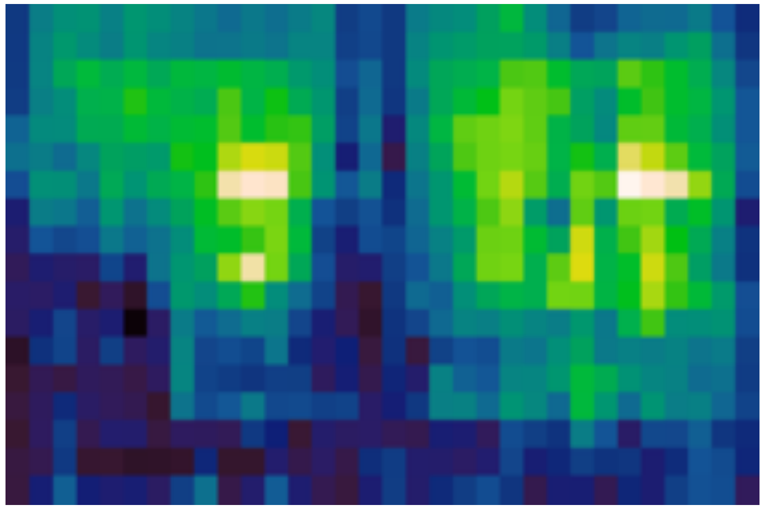
HOG feature map



Template

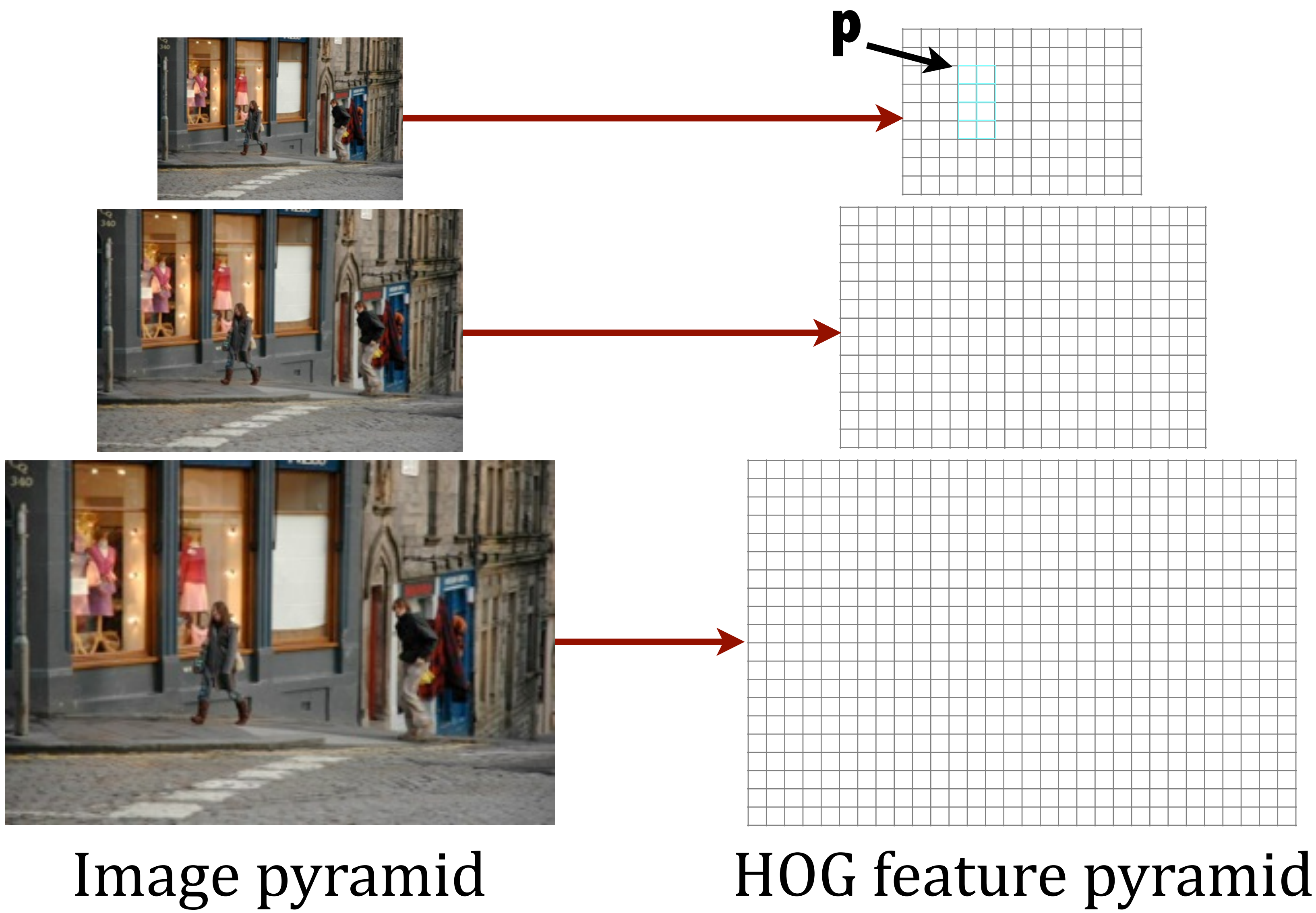


Detector response map



- Compute the HOG feature map for the image
- Convolve the template with the feature map to get score
- Find peaks of the response map (non-max suppression)
- What about multi-scale?

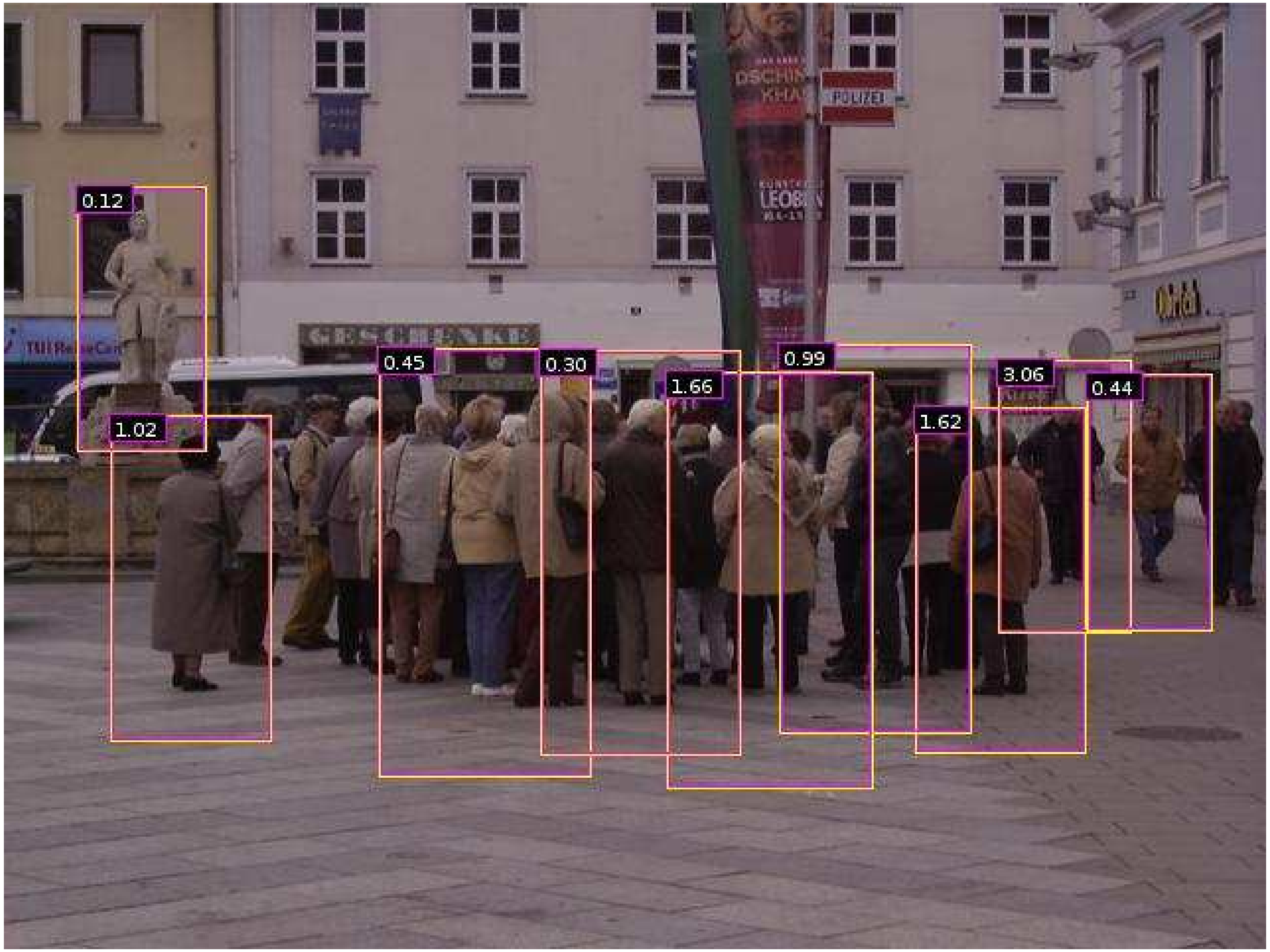
Multi-scale template matching



$$score(I, p) = \mathbf{w} \cdot \phi(I, p)$$

- Compute HOG of the whole image at multiple resolutions
- Score each sub-windows of the feature pyramid

Example pedestrian detections

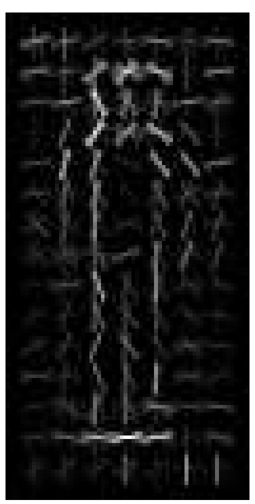


[Dalal06]

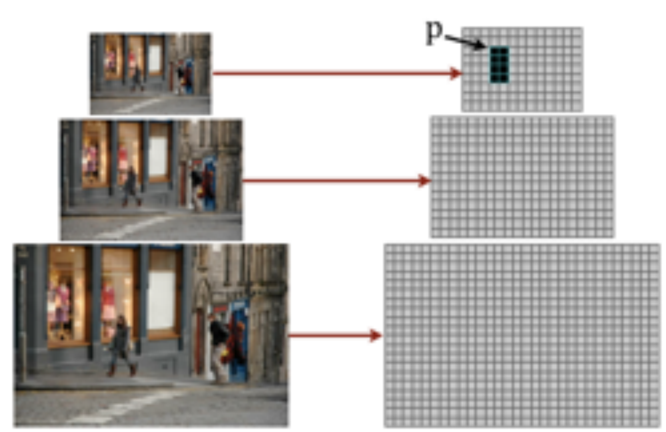
Mining hard negatives

Pos = {...  ...}

Neg_{rand} = {... random background patches ...}



SVM

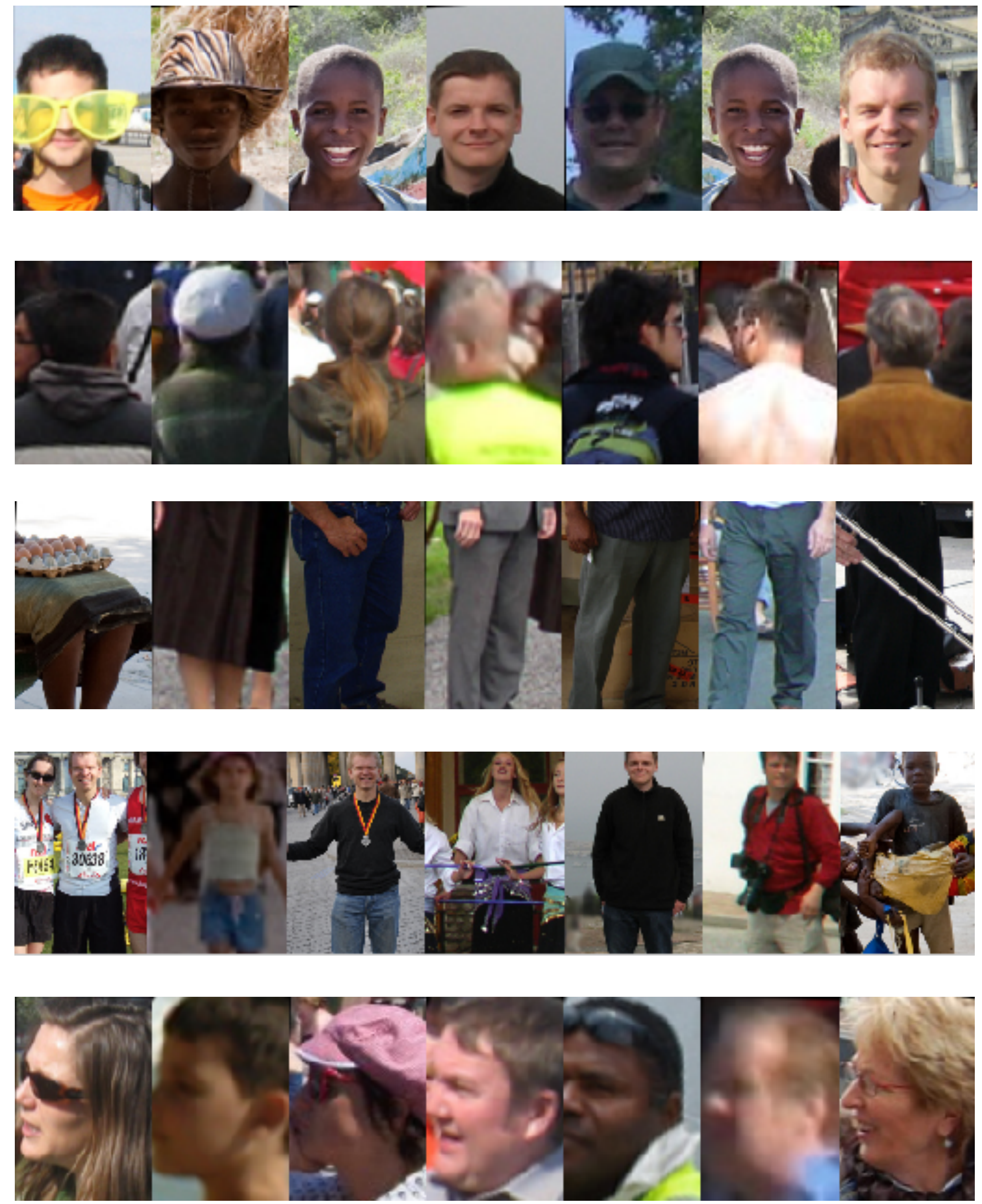


"Hard" negatives

+ Neg_{hard} = {... windows with score >= -1 ...}

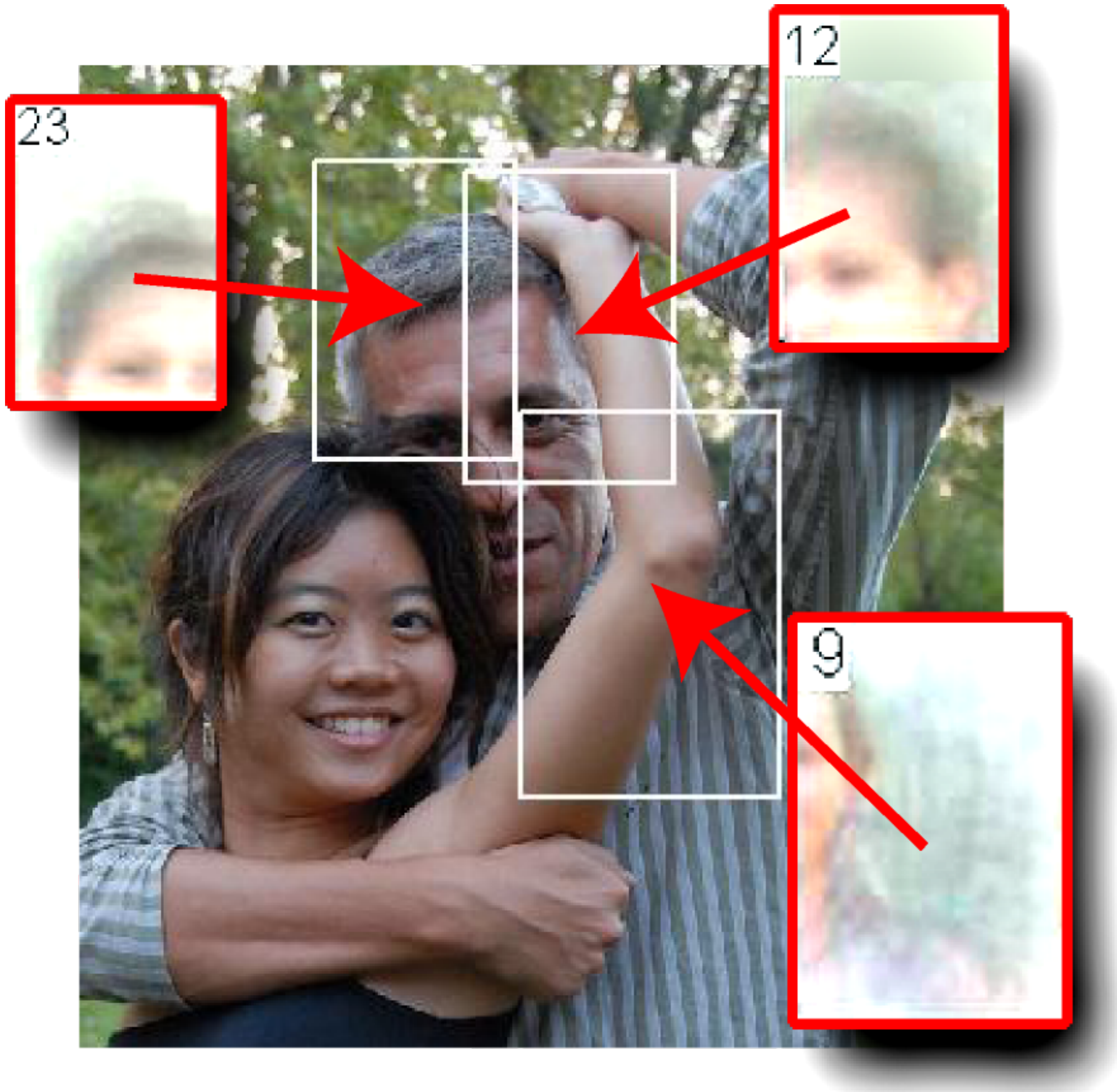
SVM

Poselets for person



Bourdev et al. 10

Person detection using poselets



- Detect each poselet in an image
- Vote for the person bounding box
- Find non-overlapping clusters
- Score each cluster using a weighted combination of poselet detection scores

$$s_i = \sum_{p \in C_i} w_p a_p$$

person detection score

weight of each poselet

poselet detection score

Bourdev & Malik 09, Bourdev et al.10, Maji & Malik 10

Issues with the “sliding window” approach

- Computationally expensive — there are too many windows
  - multiply by scales
  - multiply by aspect ratio

- Need very fast classifiers
  - Typically limited to linear SVMs and boosting
  - But these are not the most accurate (kernel SVMs, etc)

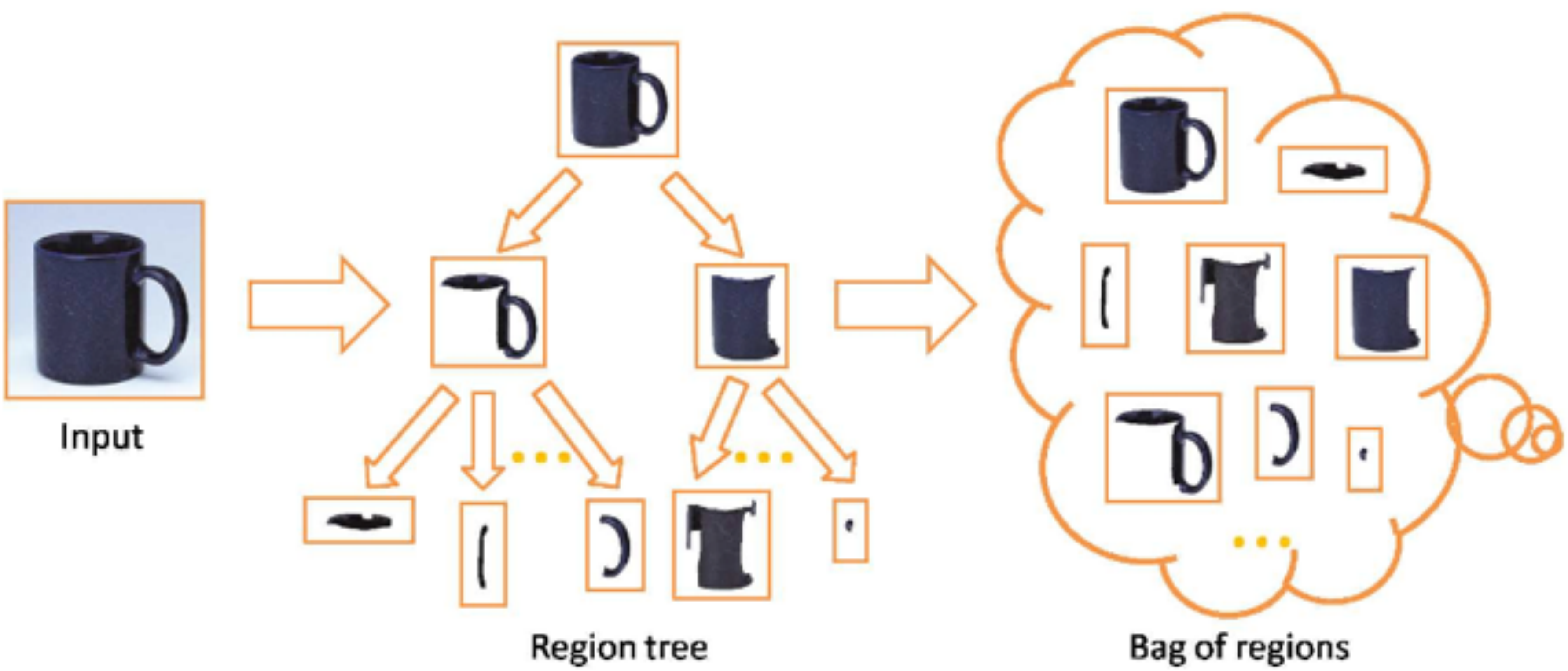
Intelligent sliding windows

- Instead of exhaustively searching over all possible windows, lets “intelligently” choose locations where the classifier is evaluated
- Some considerations:
  - We want a small number of such regions (~1000)
  - We want high recall — no objects should be missed
  - Category independent
    - that way we can share the cost of computing features
  - Fast — shouldn’t be slower than running the detector itself

How do we get such proposals?

Segmentations

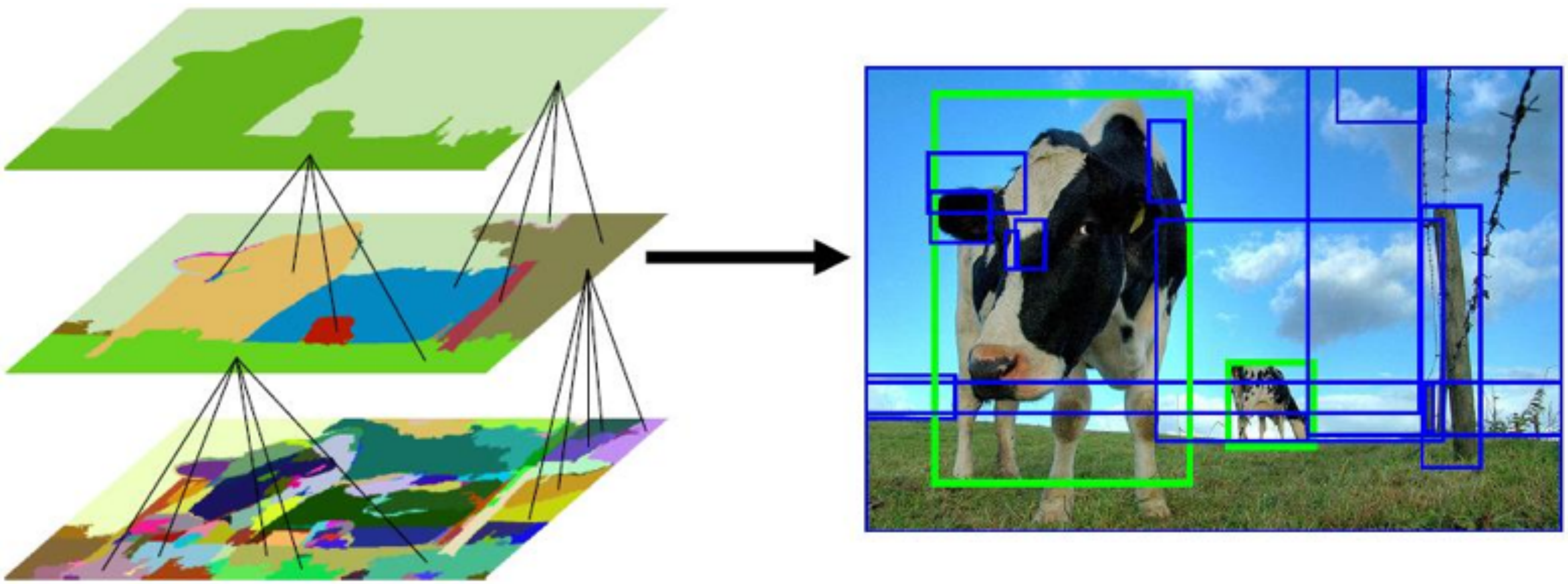
- Why might this be a good idea?
  - Can use low-level cues such as color and texture similarity which are category independent
  - Often fast to compute
  - Inherently span scale and aspect-ratio



Recognition using regions, Gu et al.

We will look at this approach

Segmentation as Selective Search for Object Recognition, K. Van de Sande, J. Uijlings, T. Gevers, and A. Smeulders, ICCV 2013



Winner of the PASCAL VOC challenge in recent years

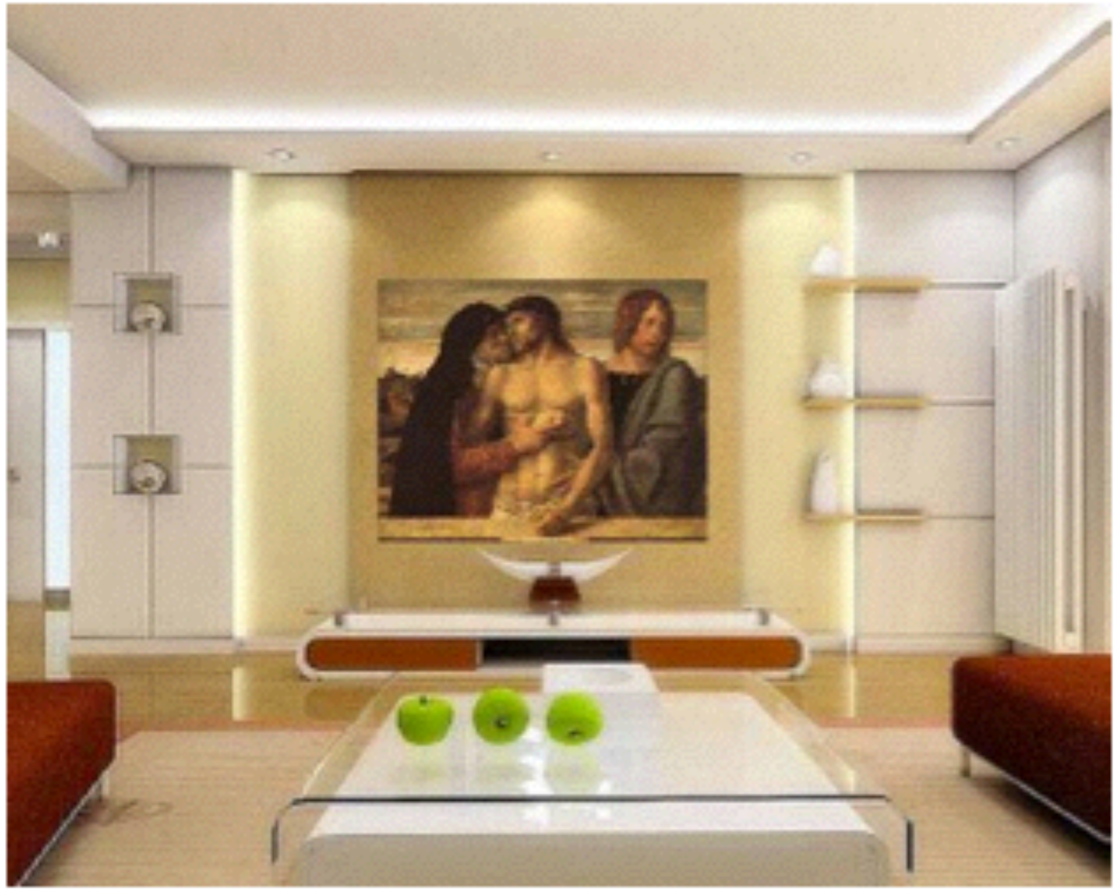
Lets start with segmentations

"Efficient graph-based image segmentation"
Felzenszwalb and Huttenlocher, IJCV 2004

- We typically get over-segmentation for big objects, i.e., objects are broken into multiple regions
- How can we fix this?

How to obtain high recall?

- Images are intrinsically hierarchical



- Segmentation at a single scale is not enough
  - Lets merge regions to produce a hierarchy

Hierarchical clustering

- Compute similarity measure between all adjacent region pairs *a* and *b* as:

$$S(a,b) = S_{size}(a,b) + S_{texture}(a,b)$$

Proportion of the image area that a and b jointly occupy

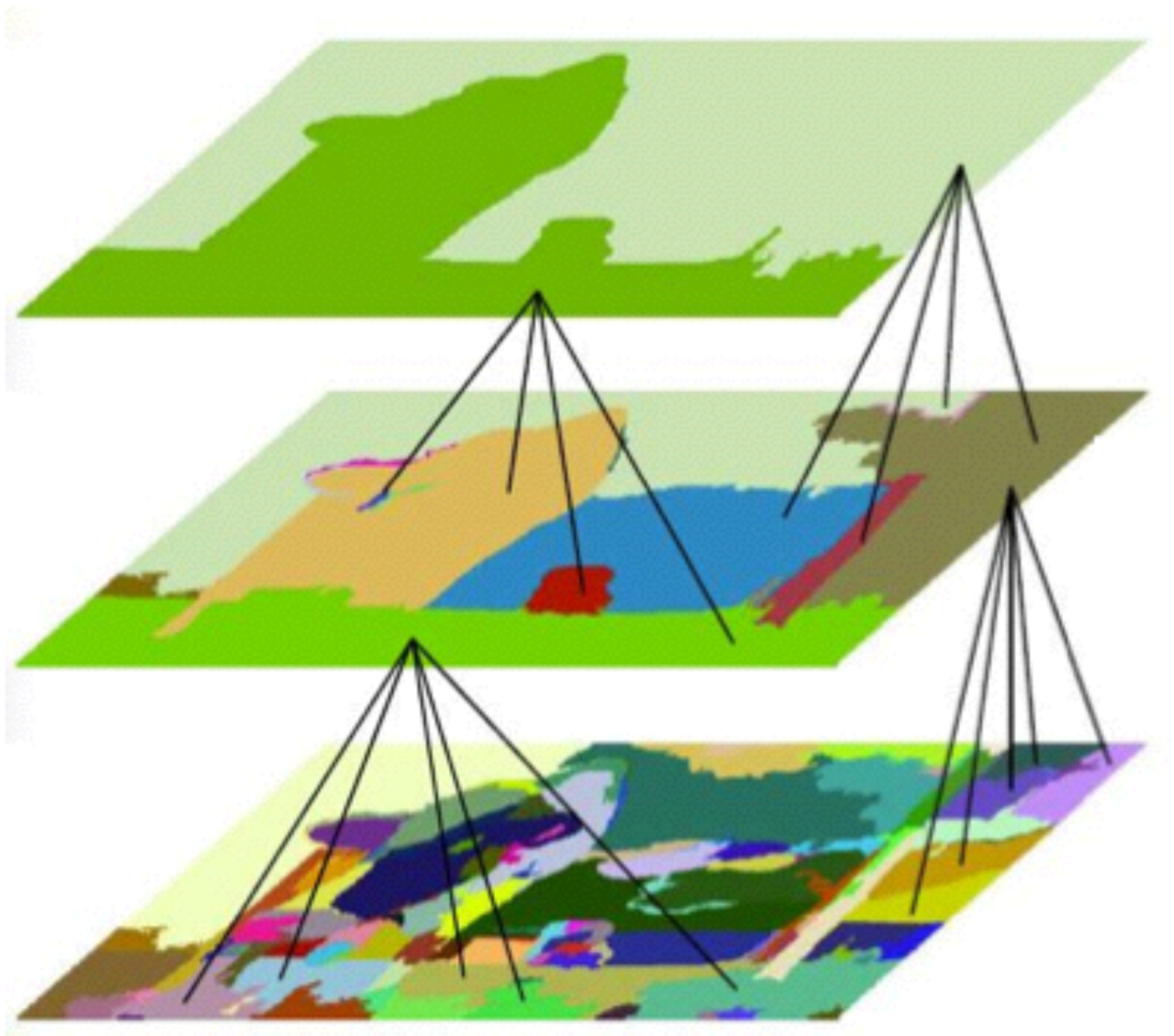
Histogram intersection of 8-bin gradient direction histogram computed in each color channel

$S_{size}(a,b)$ → Encourages small regions to merge early and prevents single region from gobbling up all others one by one.

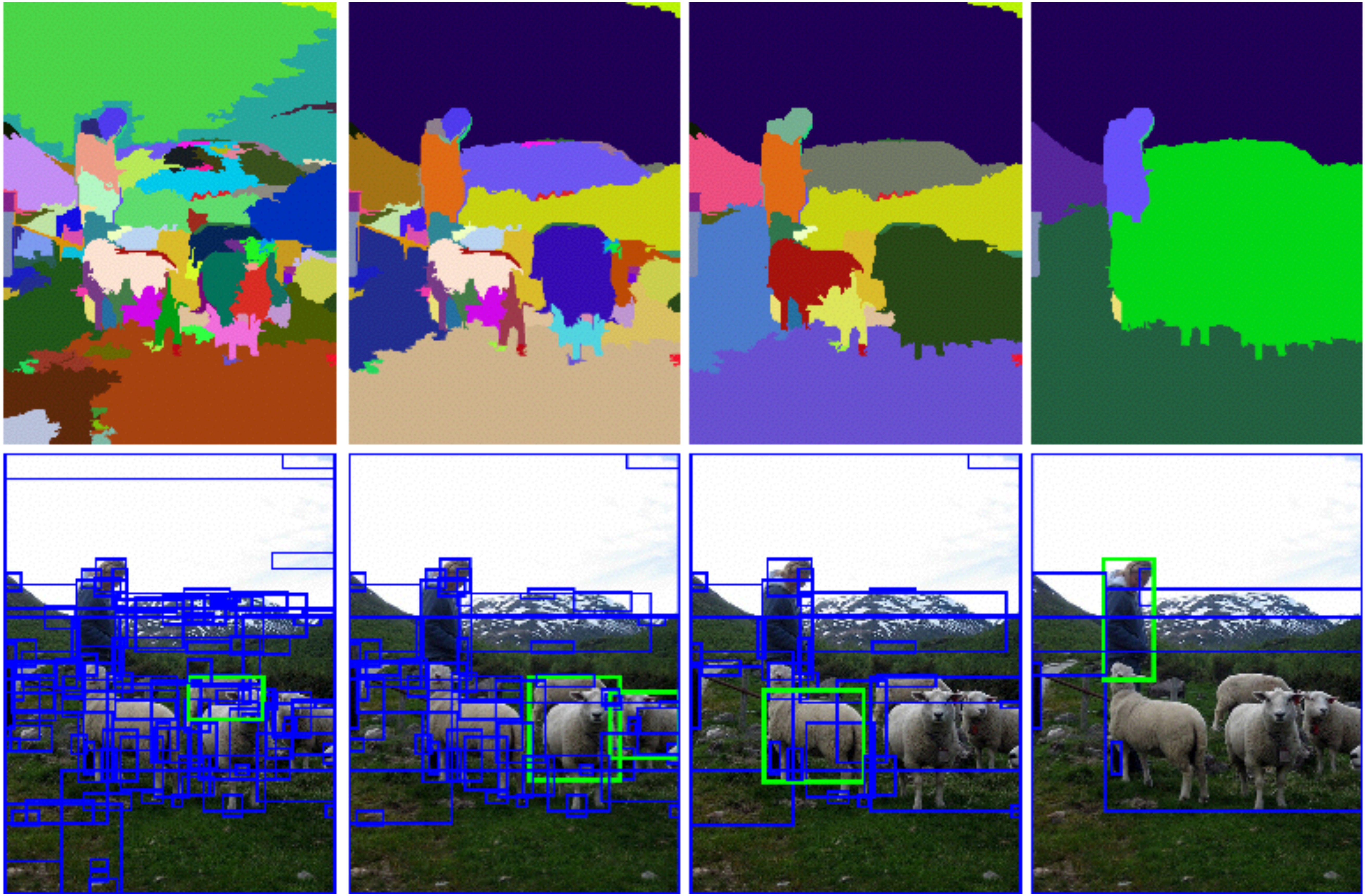
$S_{texture}(a,b)$ → Encourages regions with similar texture (and color) to be grouped early.

Hierarchical clustering

1. Merge two most similar regions based on S
2. Update similarities between the new region and its neighbors
3. Go back to step 1 until the whole image is a single regions



Example proposals



Example proposals

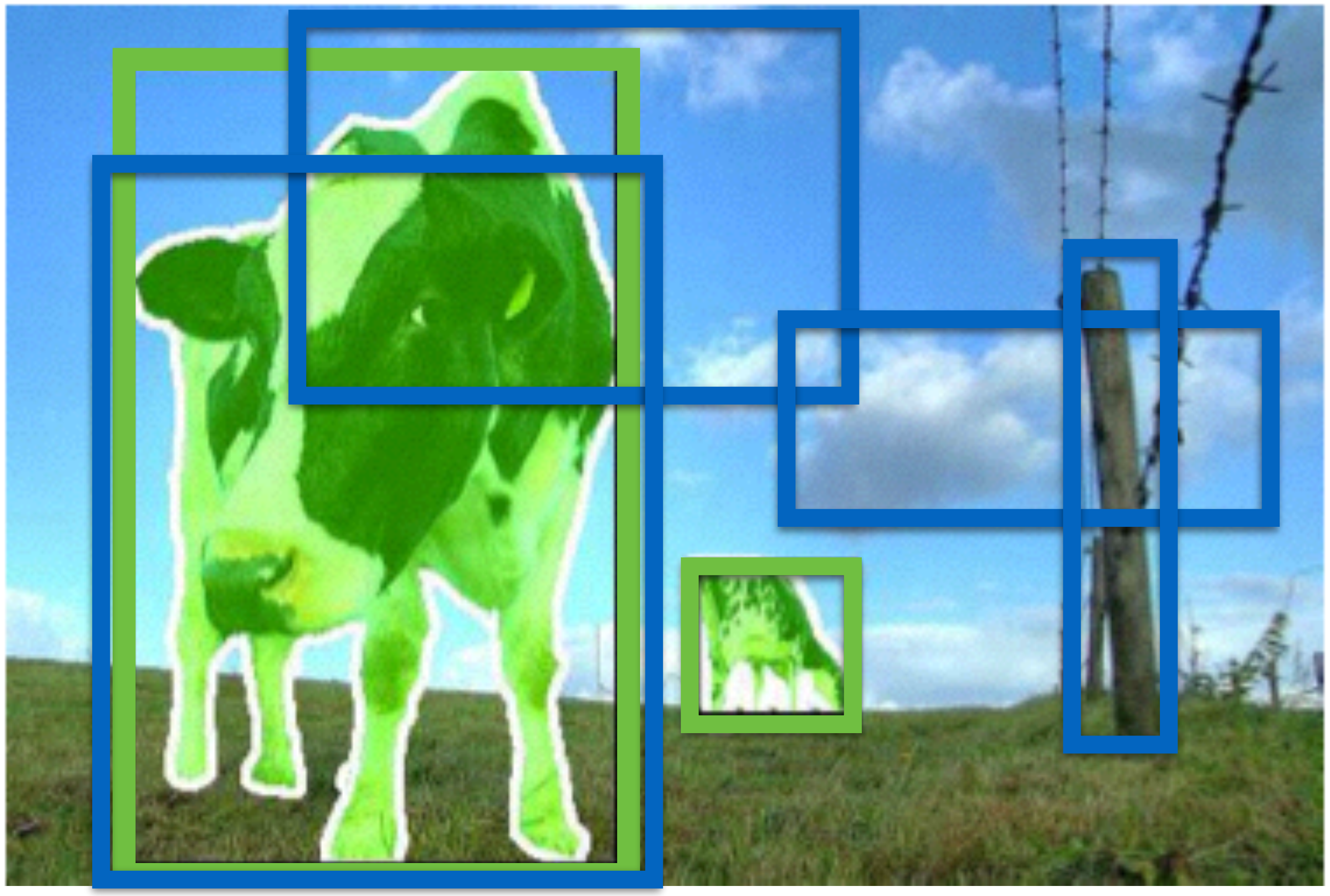
Adding diversity to the proposals

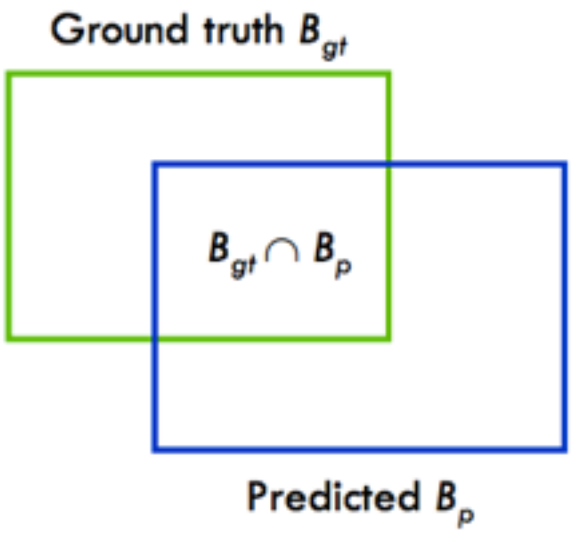
Color cues work best

Texture cues work best

- No single segmentation works for all images
- Use different color spaces
  - RGB, Opponent color (e.g., LAB), normalized rgb, hue
- Vary parameters in the Felzenszwalb segmentation method
  - k = [100, 150, 200, 250] (k= threshold parameter)

Evaluating object proposals





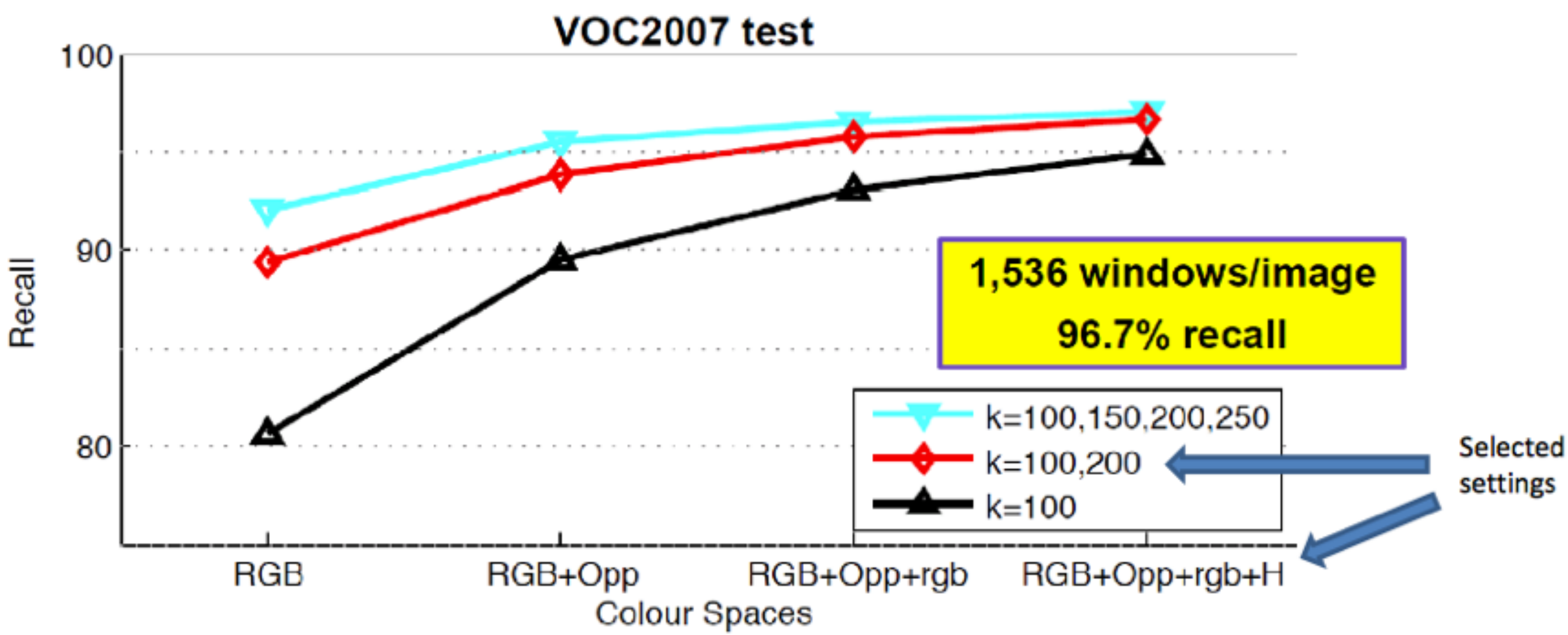
$$\text{overlap}(B_{gt}, B_p) = \frac{|B_{gt} \cap B_p|}{|B_{gt} \cup B_p|}$$

We want:

1. Every ground truth box be covered by at least one proposal
2. We want as few proposals as possible

Evaluating object proposals

- Recall is the proportion of objects that are covered by some box with overlap > 0.5

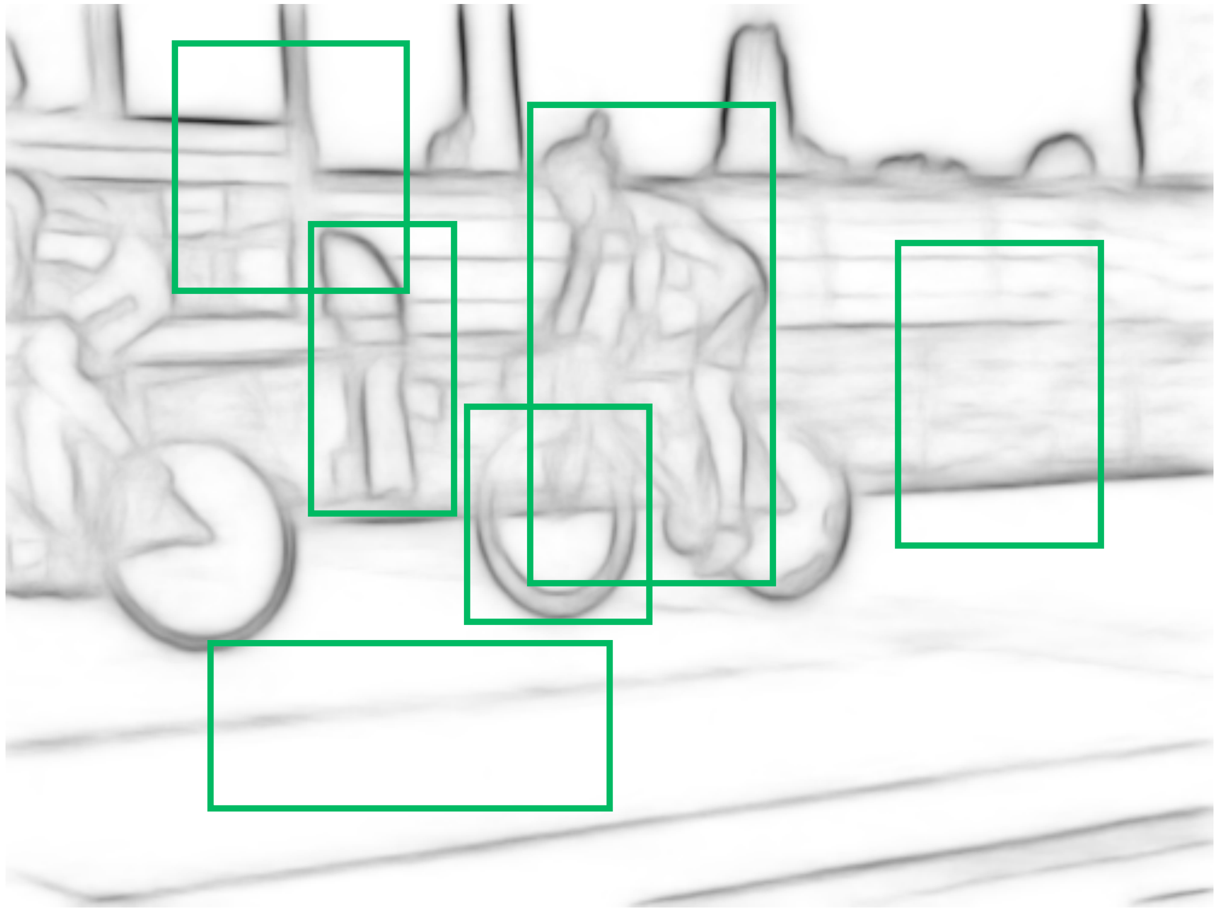


Compare this to ~100,000 regions for sliding windows

Another approach: “Objectness"

- What is an object? Alexe et al., CVPR 2010
- Learns to detect objects from background using
  - color, texture, edge cues
  - generic object detector
- One of the early methods for object proposals

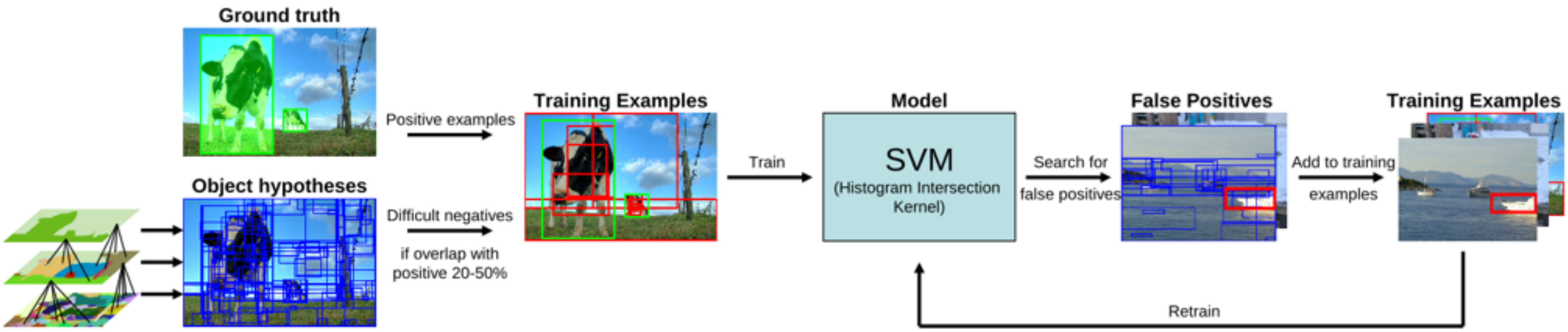
Another approach: "Edge boxes"



- Edge Boxes: Locating Object Proposals from Edges, Zitnick and Dollar, ECCV 2014
- Number of contours that are wholly contained inside the box is an indicative of the likelihood that the box contains an object.
- Very fast (0.25s per image)

Detection using region proposals

- Once again, detection = repeated classification
- But we only classify object proposals
- Training a classifier



Details of the features

- HOG was used in the Dalal & Triggs model for efficiency
- But we can use complex features and better classifiers
    - In particular SIFT bag of words features

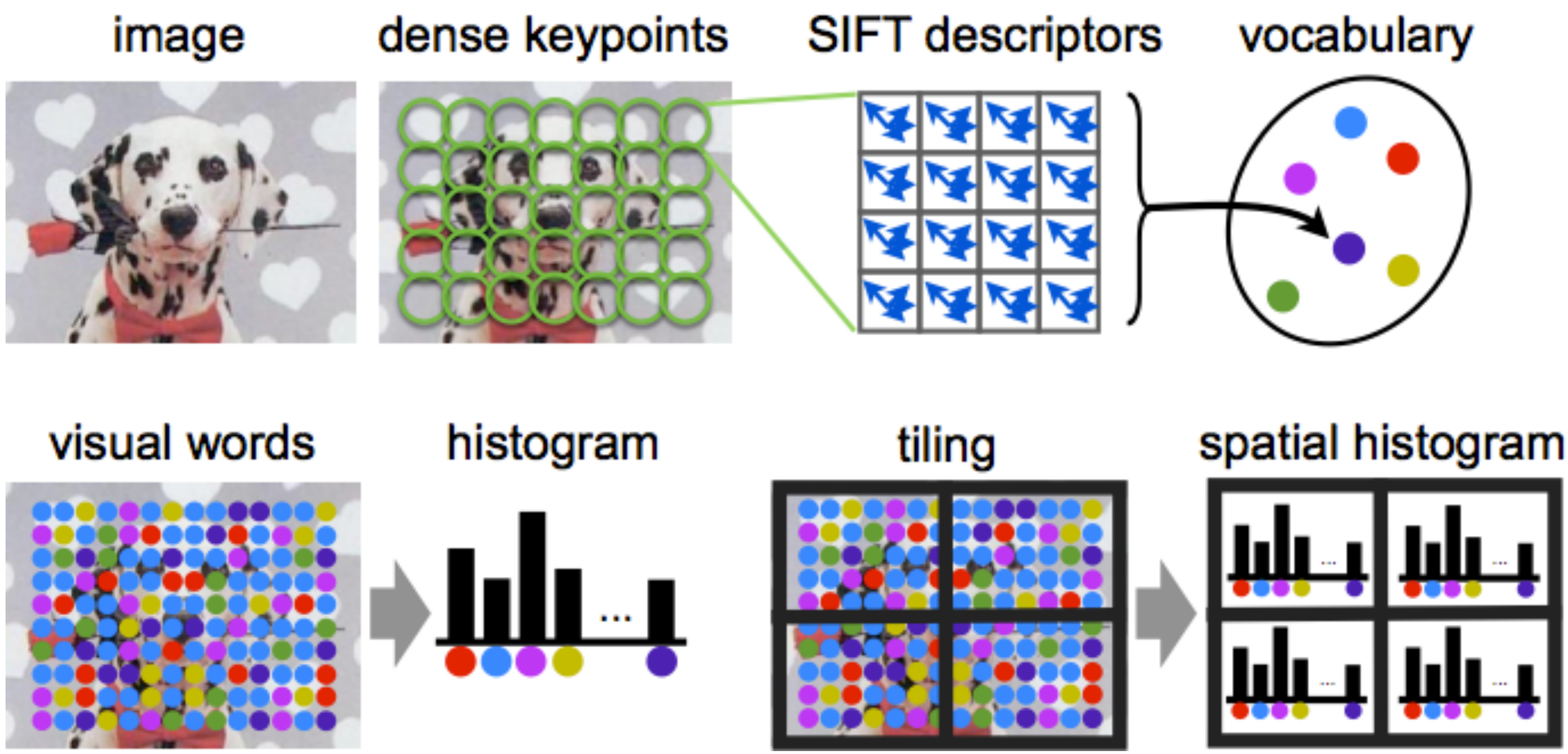


Image by Andrea Vedaldi

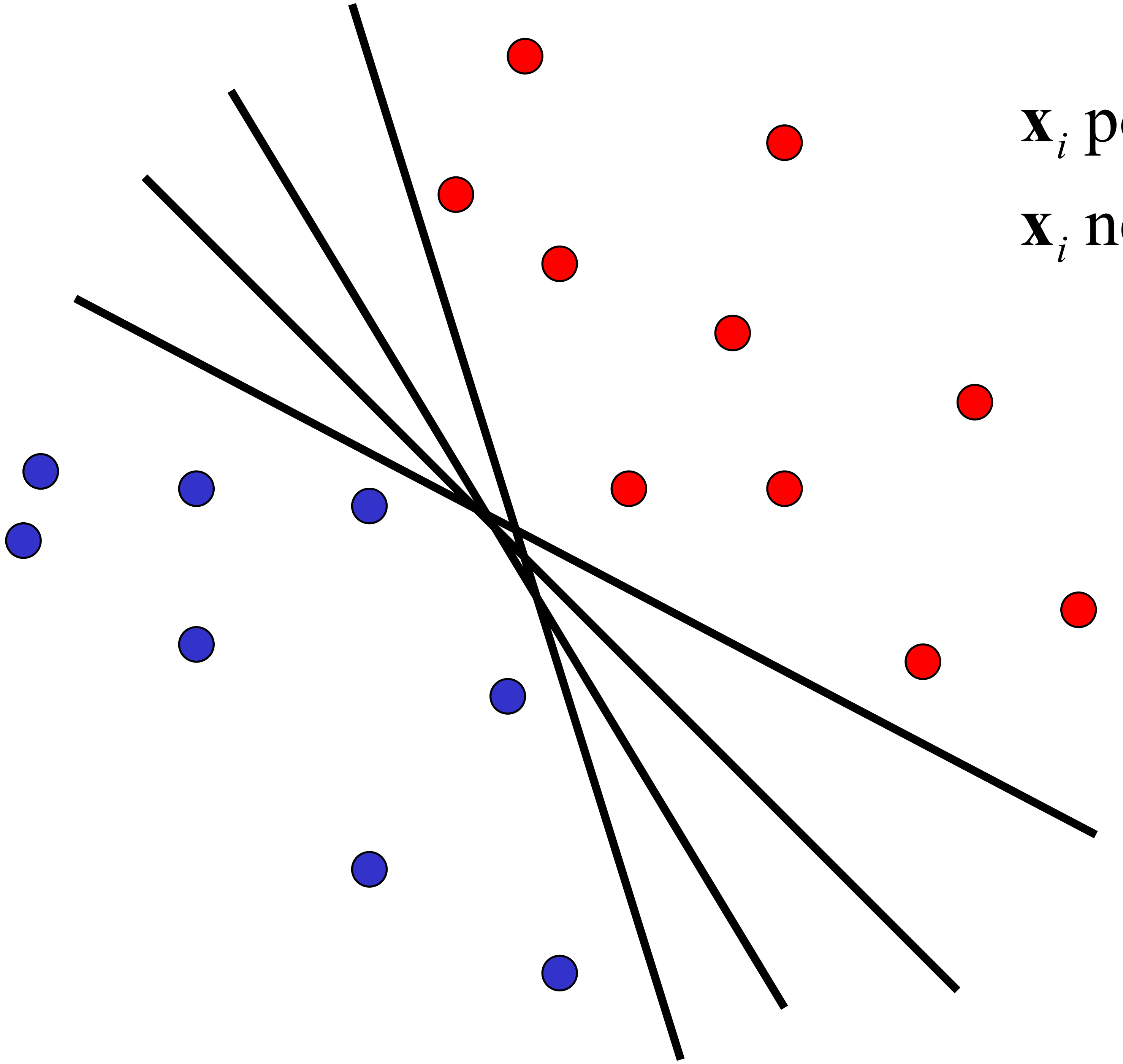
Details of the classifier

- SVM classifier with a histogram intersection kernel
- Recap of SVMs

Linear classifiers

Linear classifiers

- Find linear function (*hyperplane*) to separate positive and negative examples



$$\mathbf{x}_i \text{ positive}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$

$$\mathbf{x}_i \text{ negative}: \quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$$

Which hyperplane is best?

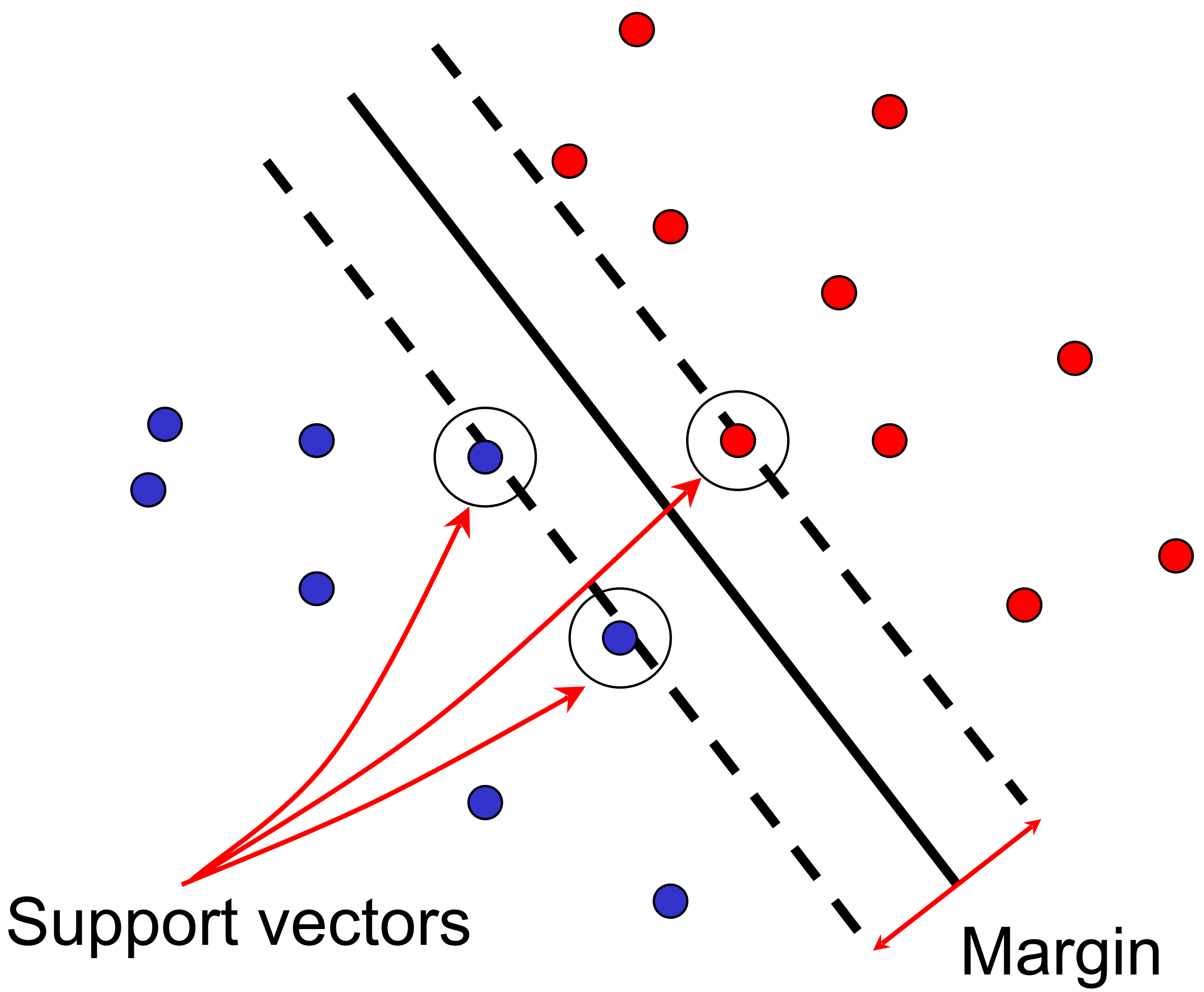
Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples

C. Burges, **A Tutorial on Support Vector Machines for Pattern Recognition**, Data Mining and Knowledge Discovery, 1998

Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples



$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and hyperplane: $\quad \dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

Therefore, the margin is $2 / \|\mathbf{w}\|$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Finding the maximum margin hyperplane

1. Maximize margin $2 / \|\mathbf{w}\|$
2. Correctly classify all training data:

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

Quadratic optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$$

C. Burges, **A Tutorial on Support Vector Machines for Pattern Recognition**, Data Mining and Knowledge Discovery, 1998

Finding the maximum margin hyperplane

- Solution:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

Learned weight
(nonzero only for support vectors)

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Finding the maximum margin hyperplane

- Solution: $$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

  $\mathbf{w} \cdot \mathbf{x}_i + b = y_i$, for any support vector

- Classification function (decision boundary):

$$\mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point $\mathbf{x}$ and the support vectors $\mathbf{x}_i$

- Solving the optimization problem also involves computing the inner products $\mathbf{x}_i \cdot \mathbf{x}_j$ between all pairs of training points

C. Burges, [A Tutorial on Support Vector Machines for Pattern Recognition](), Data Mining and Knowledge Discovery, 1998

What if the data is not linearly separable?

- Separable: $$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$$

- Non-separable: $$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$ $$\text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i \geq 0$$

- *C*: tradeoff constant, ξ_i : *slack variable* (positive)
- Whenever margin is ≥ 1, $\xi_i = 0$ $\xi_i = 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)$
- Whenever margin is < 1,

What if the data is not linearly separable?

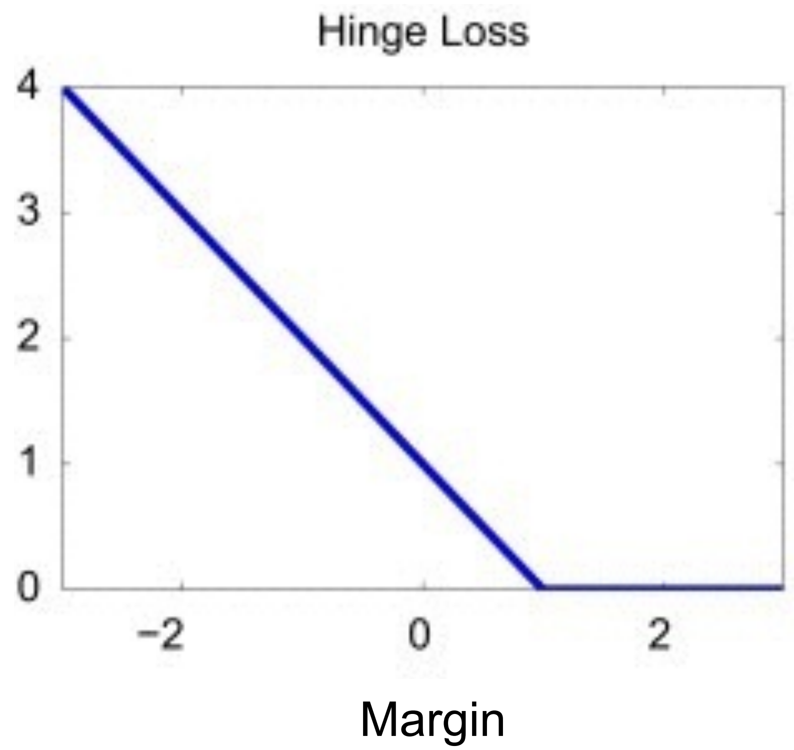
$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \max\left(0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)\right)$$

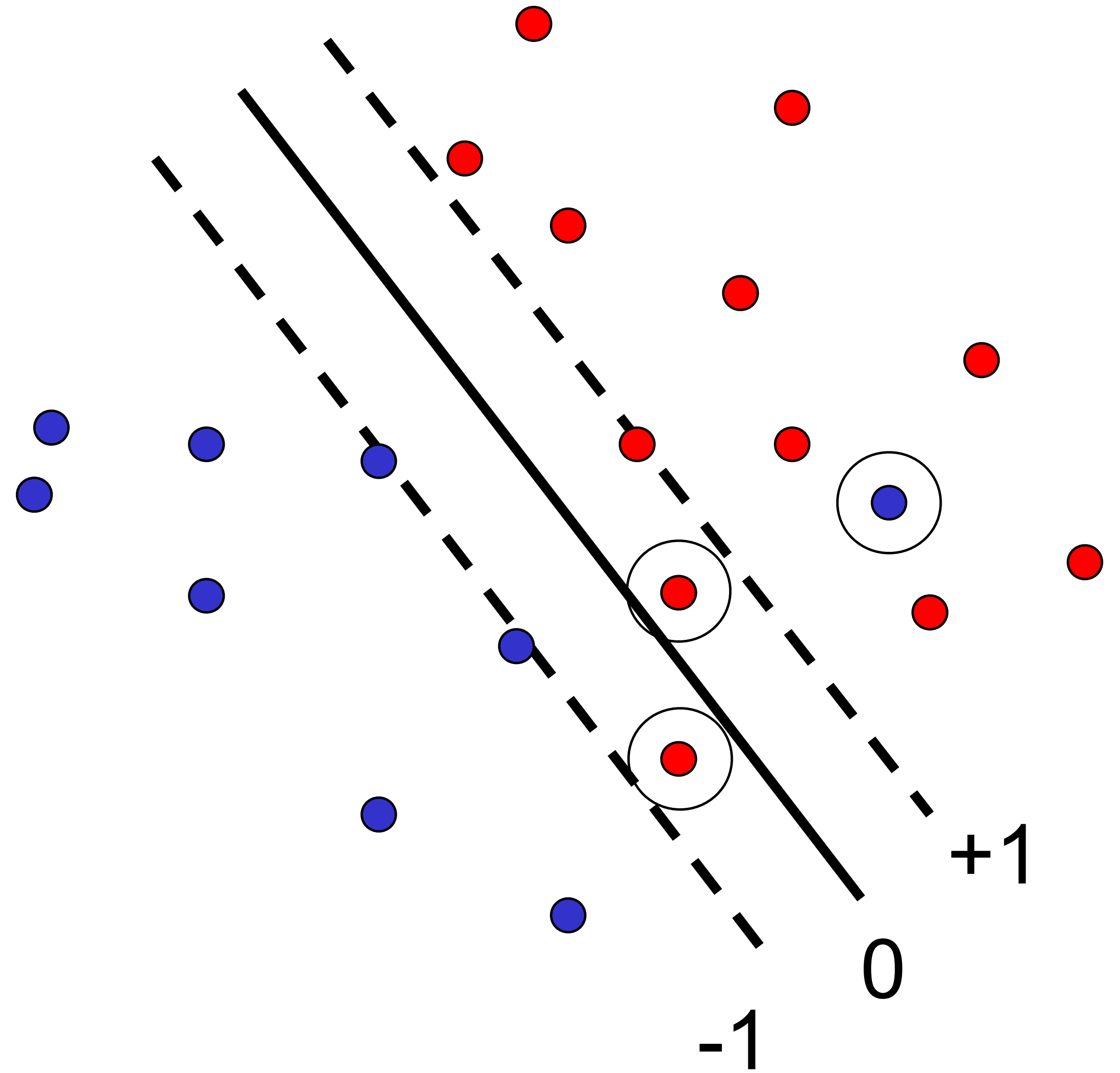
Maximize margin

Minimize classification mistakes

What if the data is not linearly separable?

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\max\big(0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)\big)$$

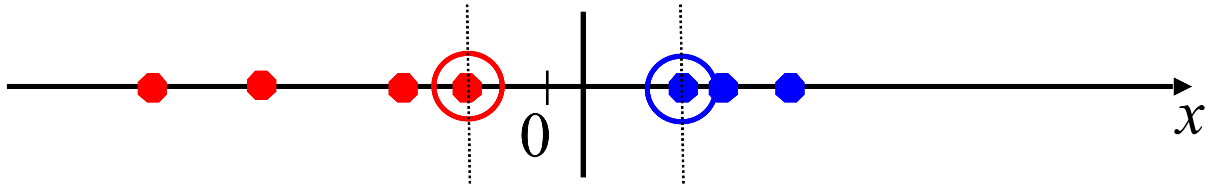




Demo: http://cs.stanford.edu/people/karpathy/svmjs/demo

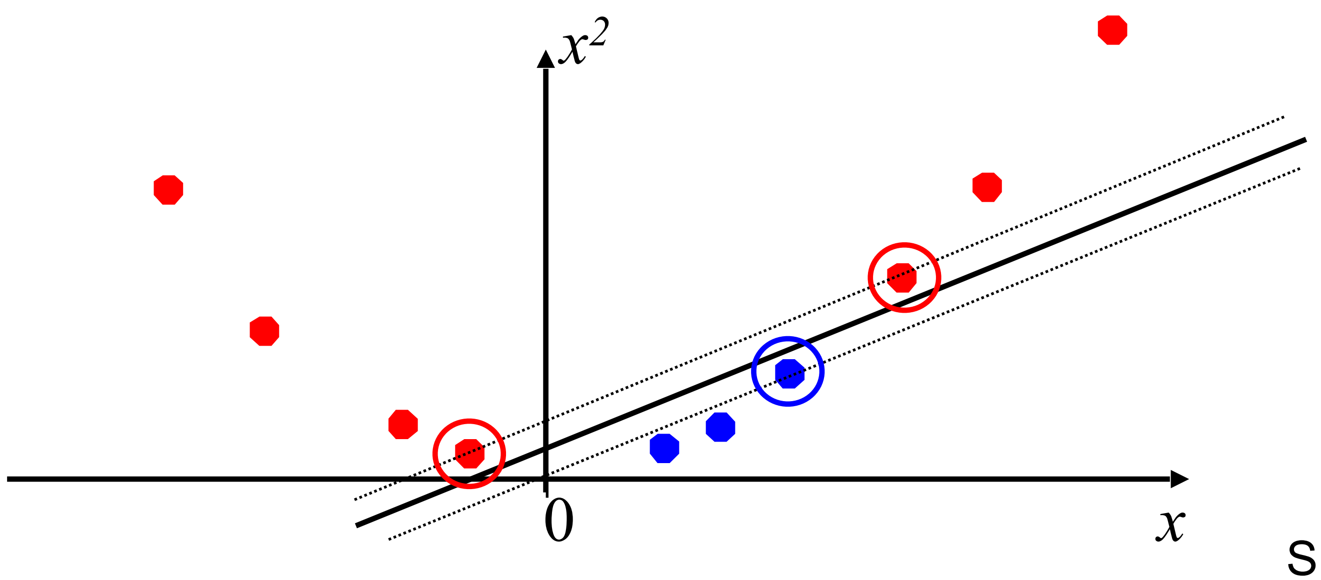
Nonlinear SVMs

- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?

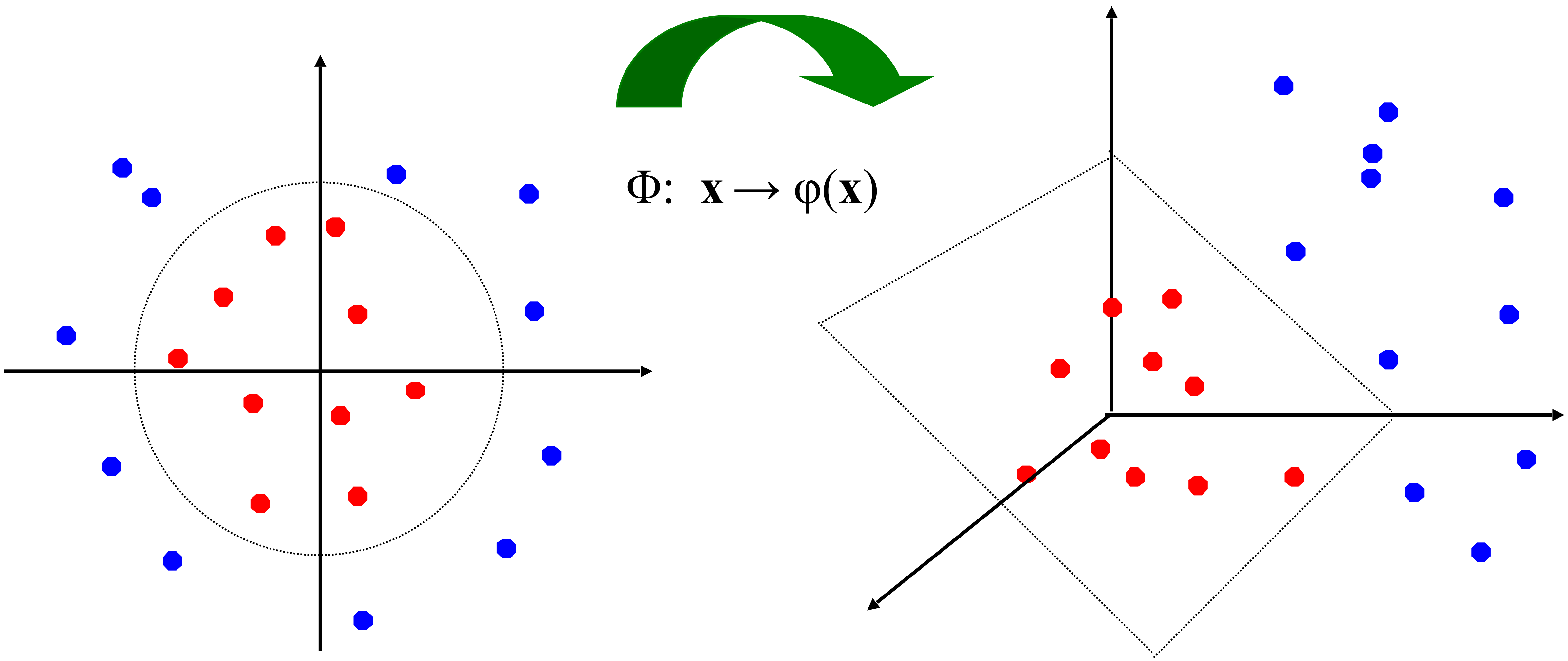
- We can map it to a higher-dimensional space:



Slide credit: Andrew Moore

Nonlinear SVMs

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Slide credit: Andrew Moore

Nonlinear SVMs

- *The kernel trick*: instead of explicitly computing the lifting transformation $\boldsymbol{\varphi}(\mathbf{x})$, define a kernel function K such that

$$K(\mathbf{x}, \mathbf{y}) = \boldsymbol{\varphi}(\mathbf{x}) \cdot \boldsymbol{\varphi}(\mathbf{y})$$

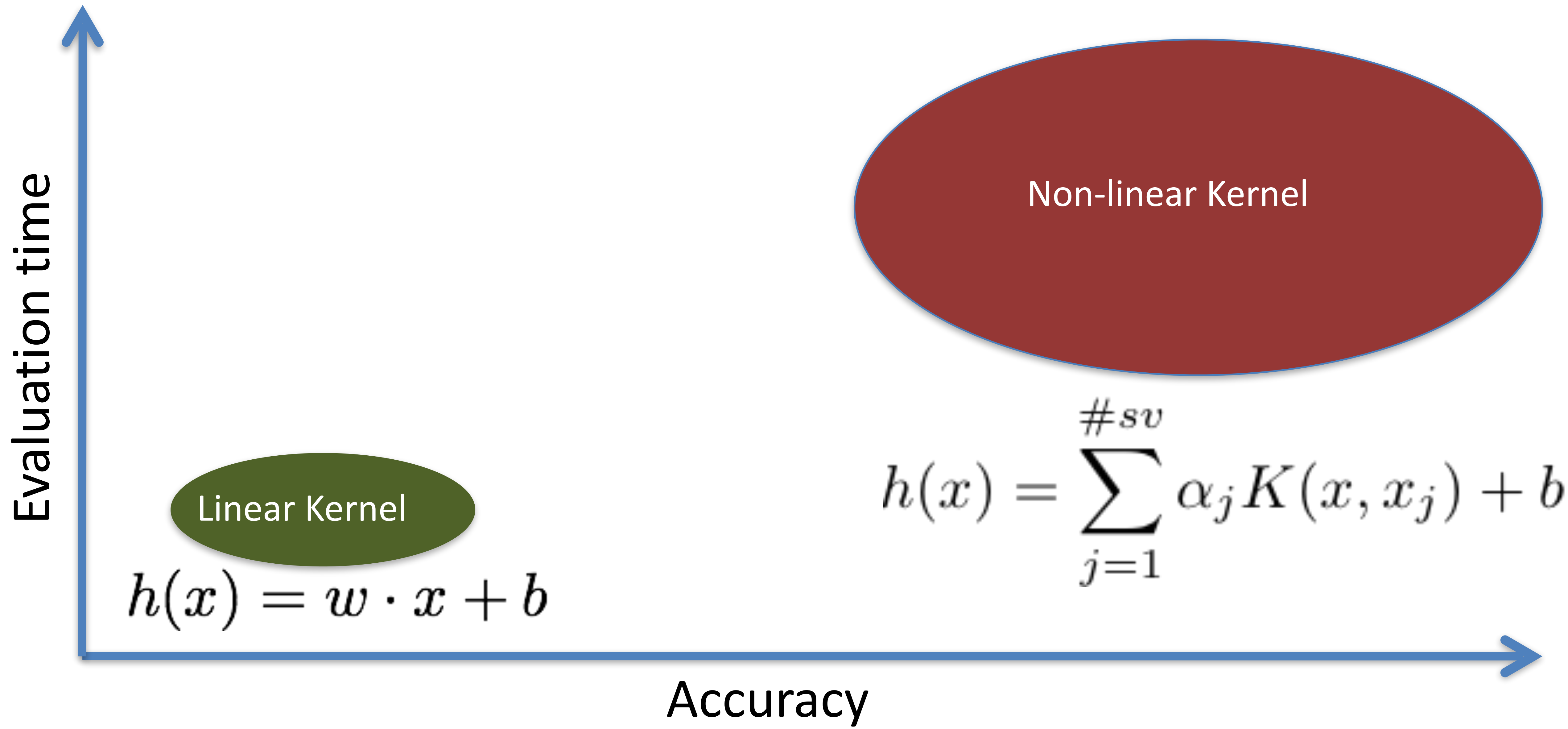
(the kernel function must satisfy *Mercer's condition*)

- This gives a nonlinear decision boundary in the original feature space:

$$\sum_i \alpha_i y_i \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}) + b = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

C. Burges, [A Tutorial on Support Vector Machines for Pattern Recognition](), Data Mining and Knowledge Discovery, 1998

Accuracy vs. Evaluation Time



Linear SVM: O (feature dimension)

Non Linear SVM: O (**# support vectors** X feature dimension)

What is the Intersection Kernel?

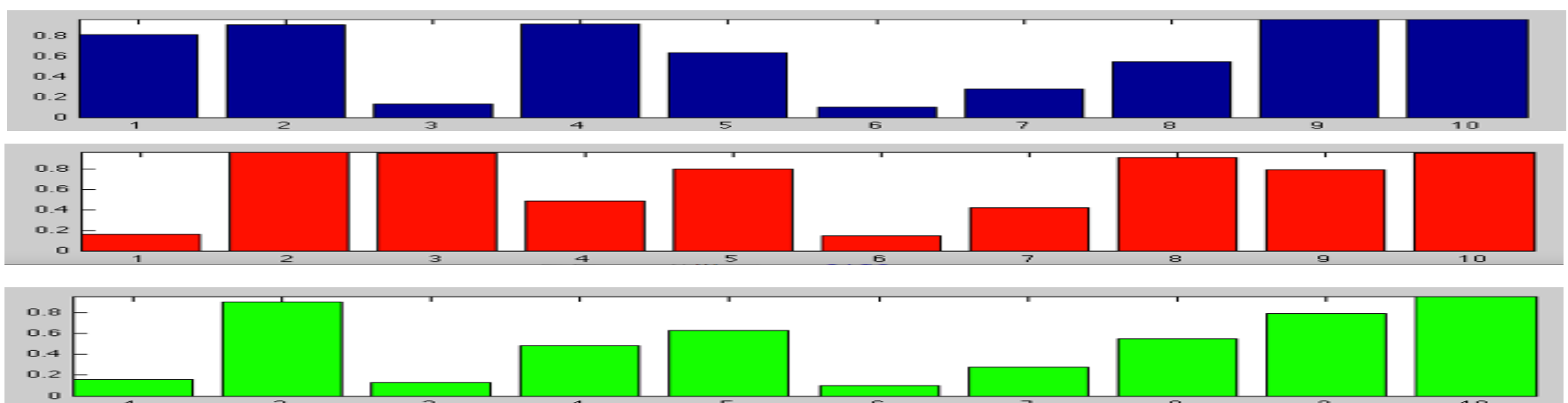
"Histogram Intersection" kernel between histograms a, b:

$$K_{\min}(a,b) = \sum_{i=1}^{n} \min(a_i, b_i) \qquad \begin{matrix} a_i \geq 0 \\ b_i \geq 0 \end{matrix}$$

a

b

min(a,b)



Introduced by Swain and Ballard 1991 to compare color histograms.

SVM classification function

$$h(x) = \sum_{j=1}^{\#sv} \alpha_j K_{\min}(x, s_j) + b = \sum_{j=1}^{\#sv} \alpha_j \left(\sum_{i=1}^{\#dim} \min(x_i, s_{ij}) \right) + b$$

sum over support vectors

#sv times slower than linear SVM

SVM classification function

$$h(x) = \sum_{j=1}^{\#sv} \alpha_j K_{\min}(x, s_j) + b = \sum_{j=1}^{\#sv} \alpha_j \left(\sum_{i=1}^{\#dim} \min(x_i, s_{ij}) \right) + b$$

Key Insight : Additive Property

$$\begin{aligned} h(x) &= \sum_{j=1}^{\#sv} \alpha_j \left(\sum_{i=1}^{\#dim} \min(x_i, s_{ij}) \right) + b \\ &= \sum_{i=1}^{\#dim} \left(\sum_{j=1}^{\#sv} \alpha_j \min(x_i, s_{ij}) \right) + b \\ &= \sum_{i=1}^{\#dim} h_i(x_i) + b \end{aligned}$$

$$h_i(x_i) = \sum_{j=1}^{\#sv} \alpha_j \min(x_i, s_{ij})$$

Maji, Berg and Malik, CVPR 08

SVM classification function

$$h(x) = \sum_{j=1}^{\#sv} \alpha_j K_{\min}(x, s_j) + b = \sum_{j=1}^{\#sv} \alpha_j \left(\sum_{i=1}^{\#dim} \min(x_i, s_{ij}) \right) + b$$

Algorithm 1

$$h_i(x_i) = \sum_{j=1}^{\#sv} \alpha_j \min(x_i, s_{ij}) \qquad O(\#sv)$$

Maji, Berg and Malik, CVPR 08

SVM classification function

$$h(x) = \sum_{j=1}^{\#sv} \alpha_j K_{\min}(x, s_j) + b = \sum_{j=1}^{\#sv} \alpha_j \left(\sum_{i=1}^{\#dim} \min(x_i, s_{ij}) \right) + b$$

Algorithm 1

$$h_i(x_i) = \sum_{j=1}^{\#sv} \alpha_j \min(x_i, s_{ij}) \qquad O(\#sv)$$

$$= \sum_{j:s_{ij}<x_i} \alpha_j s_{ij} + \left(\sum_{j:s_{ij}\geq x_i} \alpha_j \right) x_i$$

Maji, Berg and Malik, CVPR 08

SVM classification function

$$h(x) = \sum_{j=1}^{\#sv} \alpha_j K_{\min}(x, s_j) + b = \sum_{j=1}^{\#sv} \alpha_j \left(\sum_{i=1}^{\#dim} \min(x_i, s_{ij}) \right) + b$$

Algorithm 1

$$h_i(x_i) = \sum_{j=1}^{\#sv} \alpha_j \min(x_i, s_{ij}) \qquad O(\#sv)$$

$$= \sum_{j: s_{ij} < x_i} \alpha_j s_{ij} + \left(\sum_{j: s_{ij} \geq x_i} \alpha_j \right) x_i$$

sort the support vector values in each coordinate, and pre-compute these sums for each rank.

Maji, Berg and Malik, CVPR 08

SVM classification function

$$h(x) = \sum_{j=1}^{\#sv} \alpha_j K_{\min}(x, s_j) + b = \sum_{j=1}^{\#sv} \alpha_j \left(\sum_{i=1}^{\#dim} \min(x_i, s_{ij}) \right) + b$$

Algorithm 1

$$h_i(x_i) = \sum_{j=1}^{\#sv} \alpha_j \min(x_i, s_{ij})$$

~~$O(\#sv)$~~

$$= \sum_{j:s_{ij} < x_i} \alpha_j s_{ij} + \left(\sum_{j:s_{ij} \geq x_i} \alpha_j \right) x_i$$

$O(\log(\#sv))$

sort the support vector values in each coordinate, and pre-compute these sums for each rank.

To evaluate, find position of in the sorted support vector values s_{ij} (cost : log #sv) look up values, multiply & add

SVM classification function

$$h(x) = \sum_{j=1}^{\#sv} \alpha_j K_{\min}(x, s_j) + b = \sum_{j=1}^{\#sv} \alpha_j \left(\sum_{i=1}^{\#dim} \min(x_i, s_{ij}) \right) + b$$

Algorithm 2

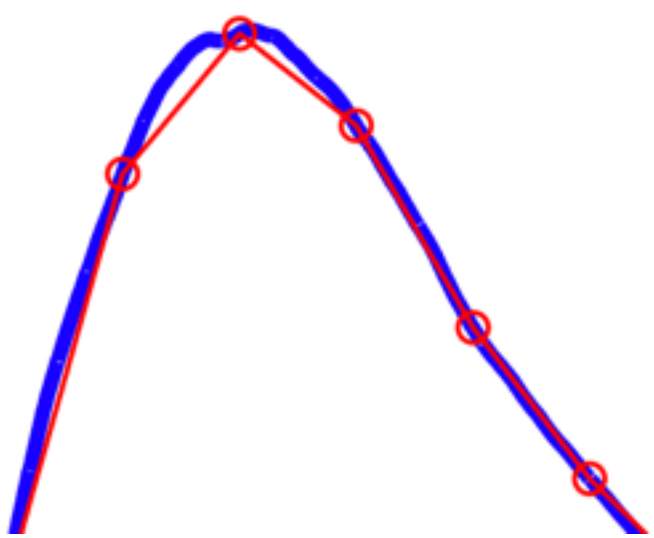
$$h_i(x_i) = \sum_{j=1}^{\#sv} \alpha_j \min(x_i, s_{ij})$$

$$= \sum_{j:s_{ij}<x_i} \alpha_j s_{ij} + \left(\sum_{j:s_{ij} \geq x_i} \alpha_j \right) x_i$$

~~$O(\#sv)$~~

~~$O(\log(\#sv))$~~

$O(1)$



For IK h_i is piecewise linear, and quite smooth, blue plot. We can *approximate* with fewer uniformly spaced segments, red plot. Saves time & space!

SVM classification function

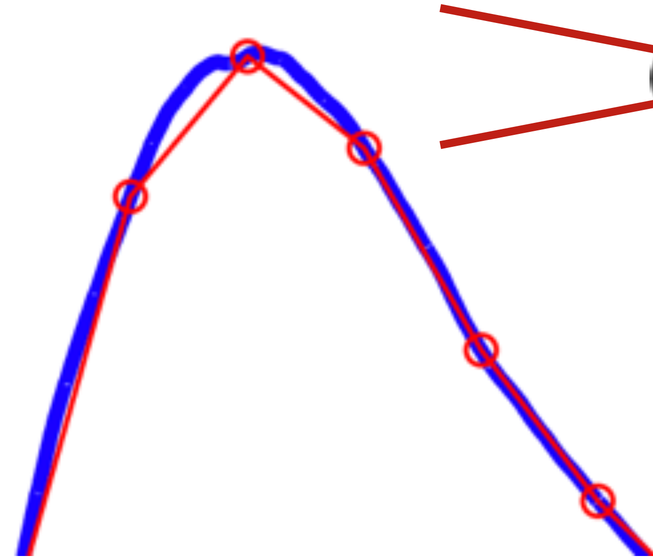
$$h(x) = \sum_{j=1}^{\#sv} \alpha_j K_{\min}(x, s_j) + b = \sum_{j=1}^{\#sv} \alpha_j \left(\sum_{i=1}^{\#dim} \min(x_i, s_{ij}) \right) + b$$

Algorithm 2

$$K(a, b) = \sum_{i=1}^{\#dim} K_i(a_i, b_i)$$

$$h_i(x_i) = \sum_{j=1}^{\#sv} \alpha_j K_i(x_i, s_{ij})$$

~~$O(\#sv)$~~



$O(1)$

Intersection $K(a, b) = \min(a, b)$

Chi-squared $K(a, b) = \frac{2ab}{a + b}$

Jensen-Shannon $K(a, b) = a \log\left(\frac{a+b}{a}\right) + b \log\left(\frac{a+b}{b}\right)$

Linear vs. Intersection Kernel SVM

Dataset	Measure	Linear SVM	IK SVM	Speedup
INRIA pedestrians	Recall@ 2 FPPI	78.9	86.6	2594 X
DC pedestrians	Accuracy	72.2	89.0	2253 X
Caltech101, 15 examples	Accuracy	38.8	50.1	37 X
Caltech101, 30 examples	Accuracy	44.3	56.6	62 X
MNIST digits	Error	1.44	0.77	2500 X
UIUC cars (Single Scale)	Precision@ EER	89.8	98.5	65 X

On average **5x** slower than linear SVM but **100-1000x** faster than standard kernel SVM classifier

Maji, Berg and Malik, CVPR 08

Results on PASCAL VOC detection

System	plane	bike	bird	boat	bottle	bus	car	cat	chair
NLPR	.533	**.553**	**.192**	**.210**	.300	.544	.467	.412	**.200**
MIT UCLA [29]	.542	.485	.157	.192	.292	**.555**	.435	.417	.169
NUS	.491	.524	.178	.120	.306	.535	.328	.373	.177
UoCTTI [9]	.524	.543	.130	.156	**.351**	.542	**.491**	.318	.155
This paper	**.582**	.419	**.192**	.140	.143	.448	.367	**.488**	.129

chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv
.200	**.315**	.207	.303	.486	.553	.465	.102	.344	.265	**.503**	.403
.169	.285	.267	.309	.483	.550	.417	.097	.358	.308	.472	.408
.177	.306	.277	.295	**.519**	**.563**	.442	.096	.148	.279	.495	.384
.155	.262	.135	.215	.454	.516	**.475**	.091	.351	.194	.466	.380
.129	.281	**.287**	**.394**	.441	.525	.258	**.141**	**.388**	**.342**	.431	**.426**

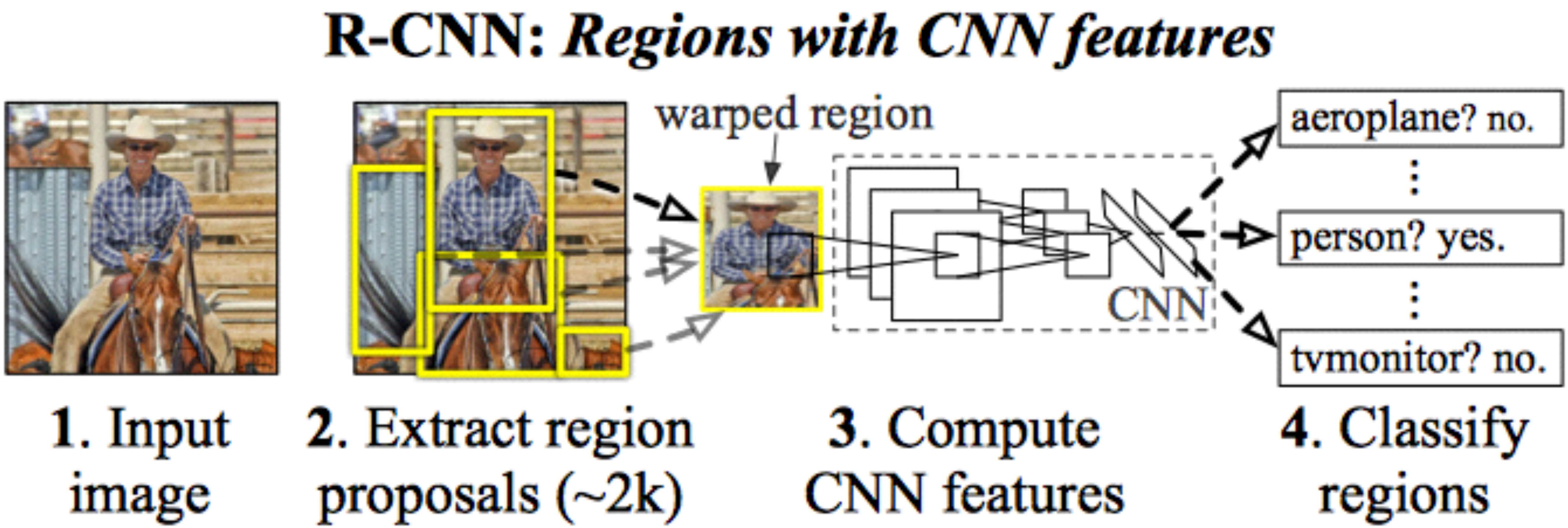
PASCAL VOC 2010 detection results

This paper = selective search

Does better on deformable objects such as animals

Current state of the art in detection

- R-CNNs (Girshick et al.)
    - Regions with CNN features



- We will look at **CNNs** in the next lecture