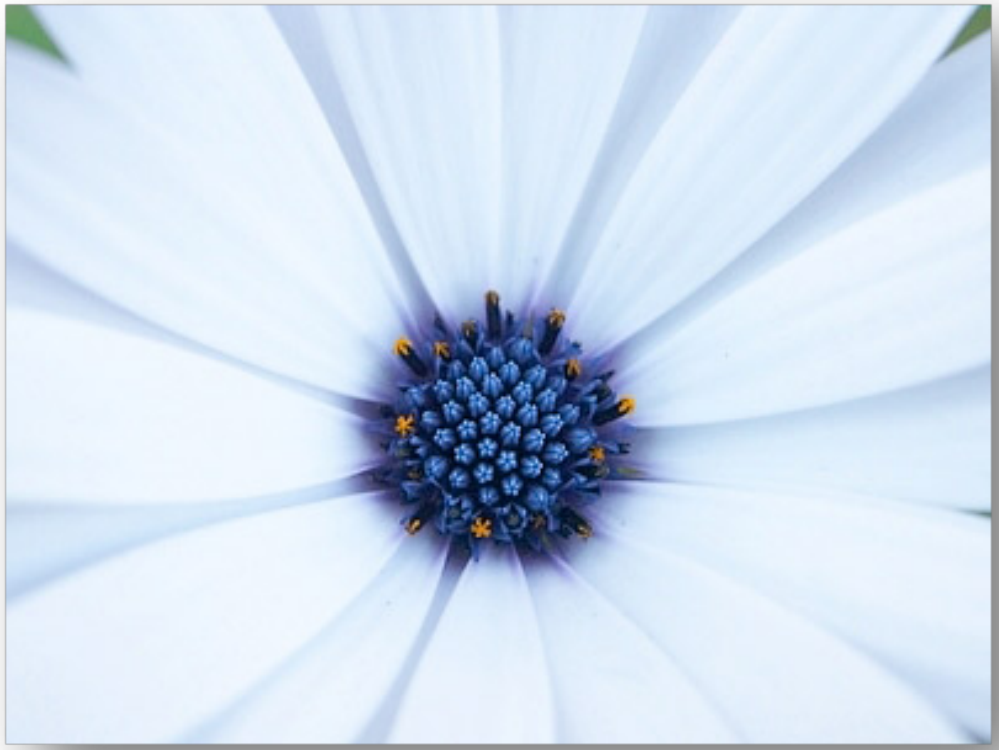
Max margin visual object processing

Jordi Vitrià

Universitat de Barcelona

November 19, 2013



Long term scientific objective

Long term objective

To understand the fundamental processes underlying the visual perception of objects (and its parts) to derive new computer vision algorithms that, one day, may enable machines to see.

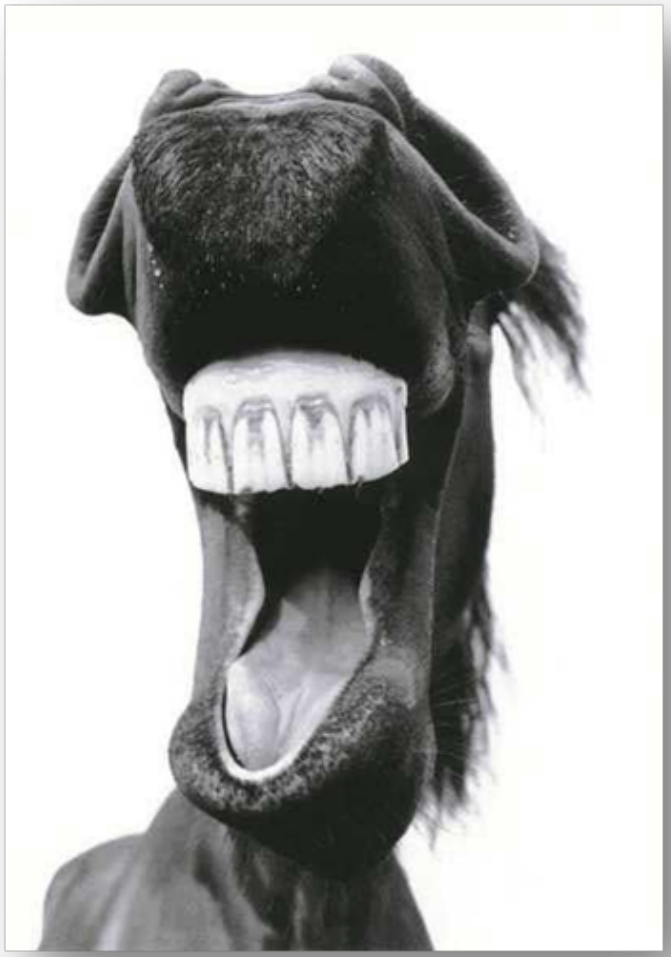
Objects

Object recognition is the answer to different questions:

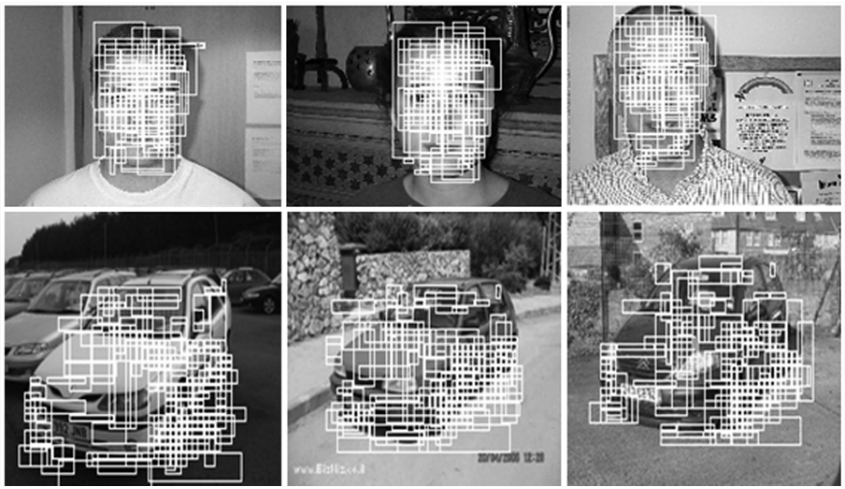
- Object **Identification**: Who is this?
- Object/category **Recognition**: What is this?
- Object **Interpretation**: To identify category instances in an image, together with their corresponding image locations, and to identify and localize its parts and subparts at multiple levels.
- Object **Re-identification**: Recognising an object at a different location after one has been captured by a camera at an early location.
- **Fine-grained** object **categorization**: To distinct into species of animals and plants, of car and motorcycle models, of architectural styles, etc. In psycology, this problem is known as *subordinate categorization*.

Identification

Recognition



Interpretation



Re-Identification

Fine-grained recognition

Roles

Object recognition plays different roles:

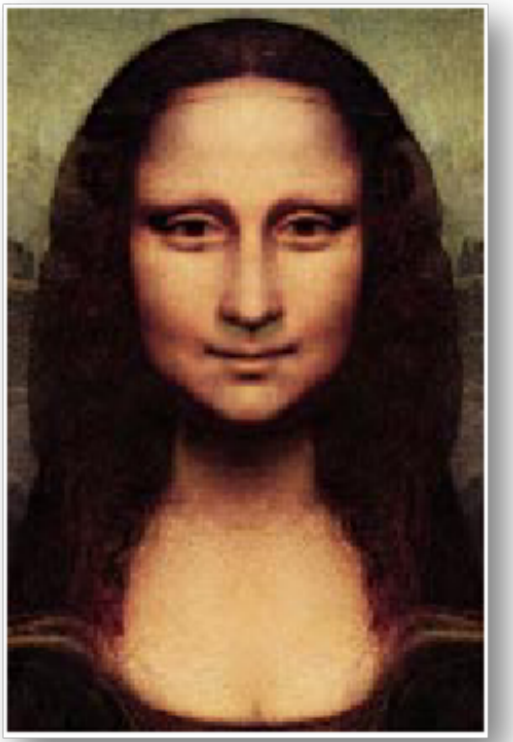
- Identification is a basic need for social life: Is this my son? Should I share my food with you?...
- Classification on the way to identification can aid in a number of ways:
  - It restricts the space of relevant models (indexing).
  - It allows the use of class-specific information (i.e, neutralizing facial expressions!). (Class-based processing)
  - It makes it possible to generalize from restricted object -specific information (i.e. generalizing from a single view of a novel face).
- (...)

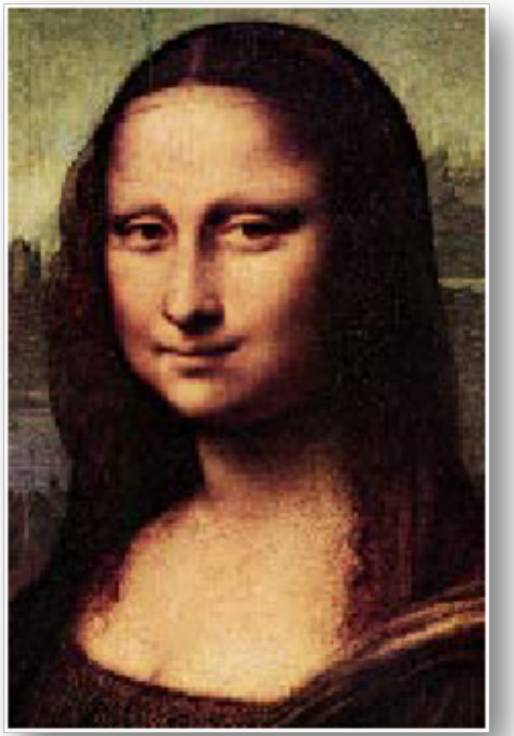
Roles

Object recognition plays different roles:

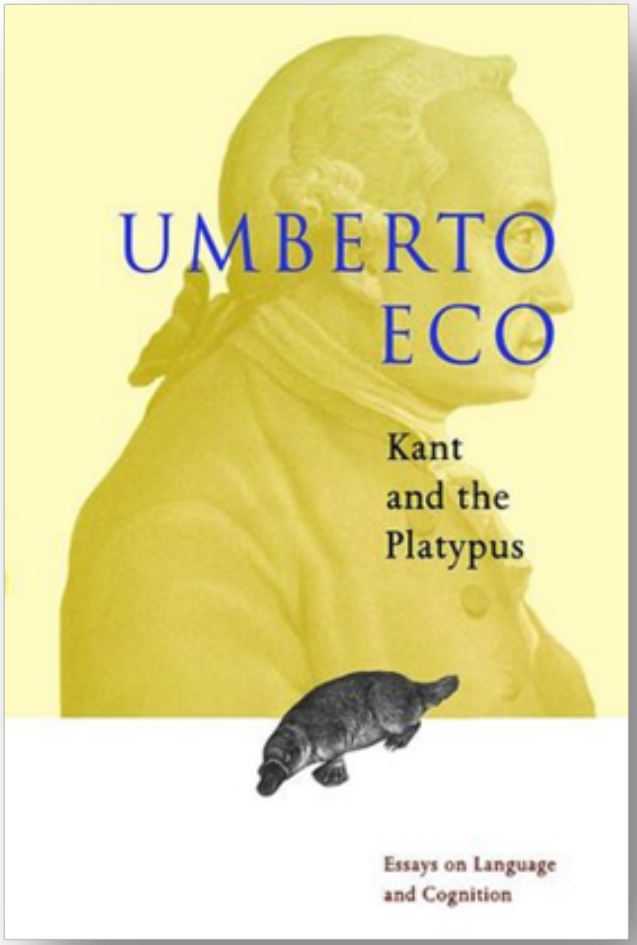
- (...)
- Recognizing novel objects. Recognition serves an important function in inferring likely properties of both known and novel objects, based on properties common to other objects on the same class (i.e. the first time I see a tiger!, or a platypus!)
- Perception of function: We can perceive the 3D shape, texture, material properties, without knowing about objects. But, the concept of category encapsulates also information about what can we do with those objects

Use of class specific information





Recognition of novel objects



Roles

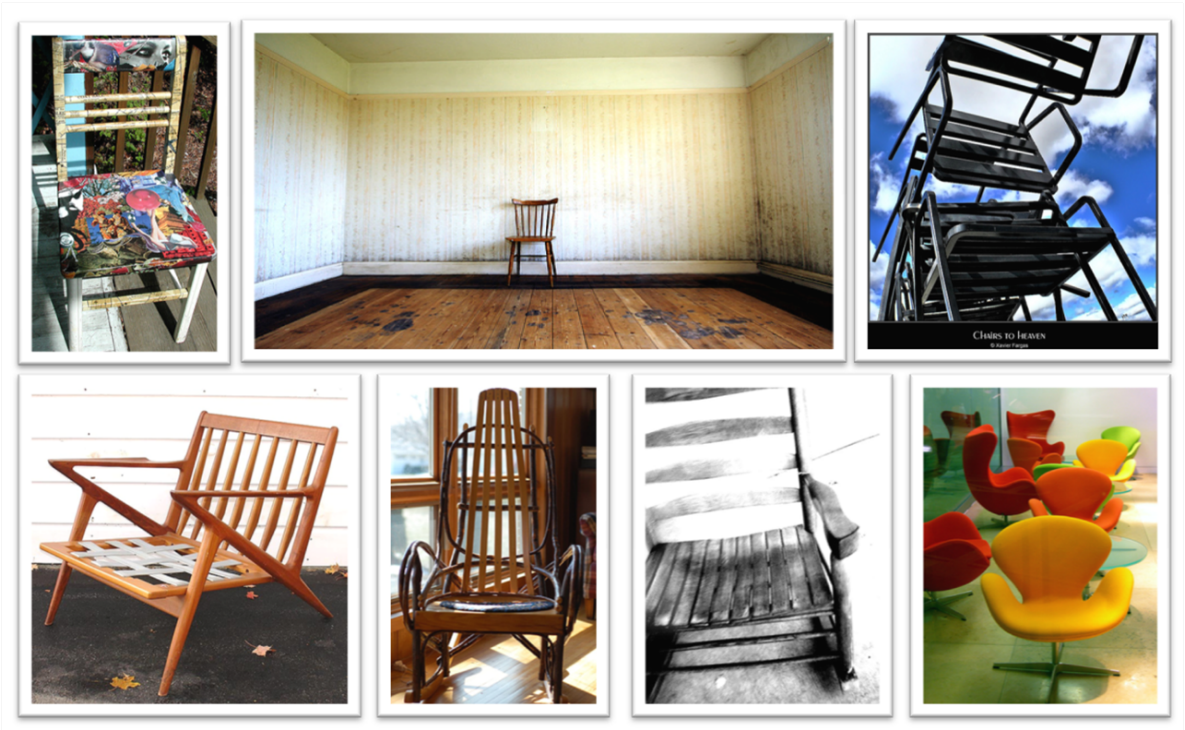
The recognition of object categories in images is one of the most challenging problems in computer vision, especially when the number of categories is large ($\mathcal{O}(30.000)$, I.Biederman).

Humans are able to recognize thousands of object types, whereas most of the existing object recognition systems are trained to recognize only a few.

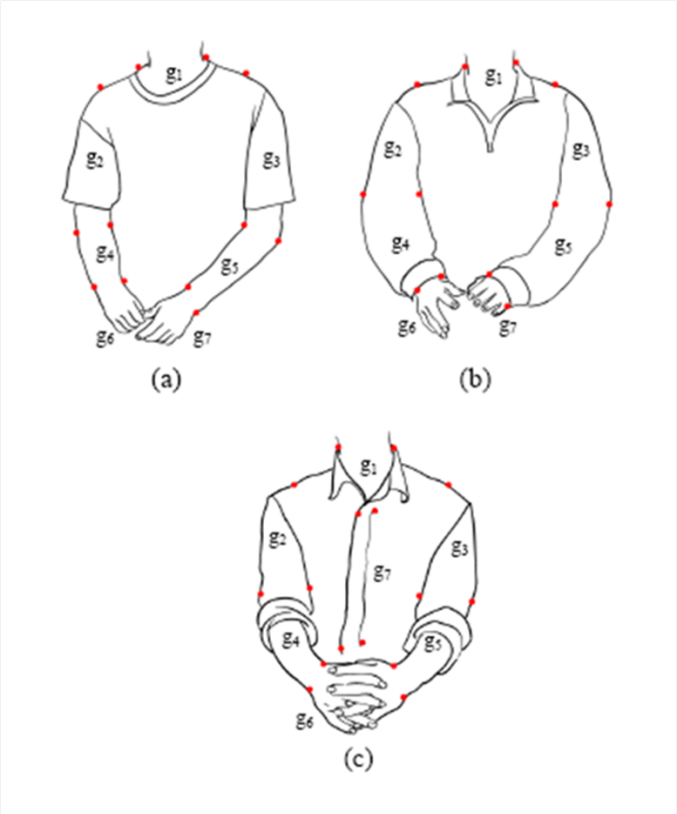
Why it is hard?

- Photometric invariance: illumination, color, texture.
- Geometric invariance: viewpoint, deformation.
- Semantic fuzziness: most categories are not defined by visual similarity (p.e flying machines).
- Functional subcategories .
- Composition.
- Occlusion.

Why it is hard?



Why it is hard?



Why it is hard?

Why it is hard?

Why do we care about recognition?

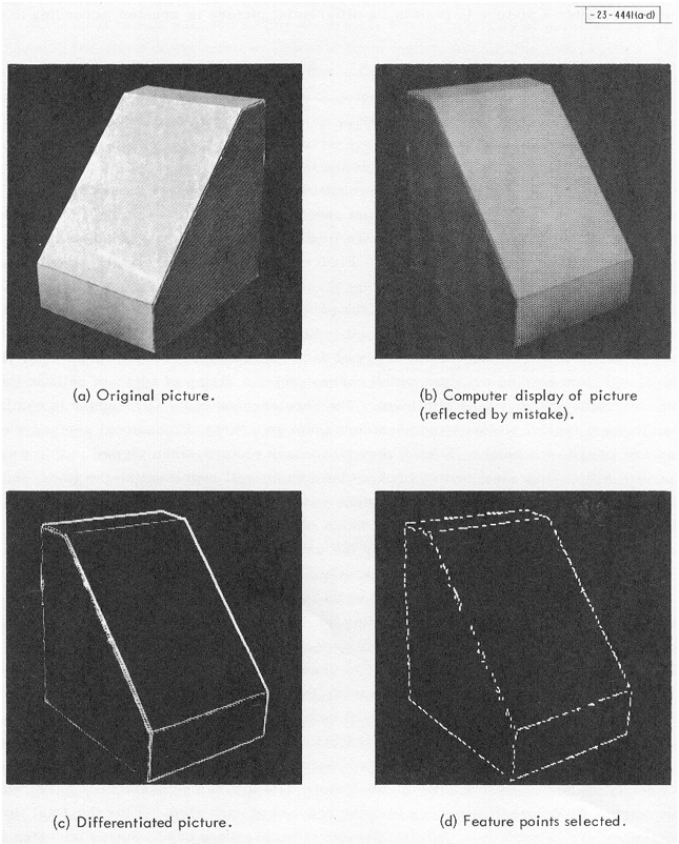
Why do we care about recognition?

Why do we care about recognition?

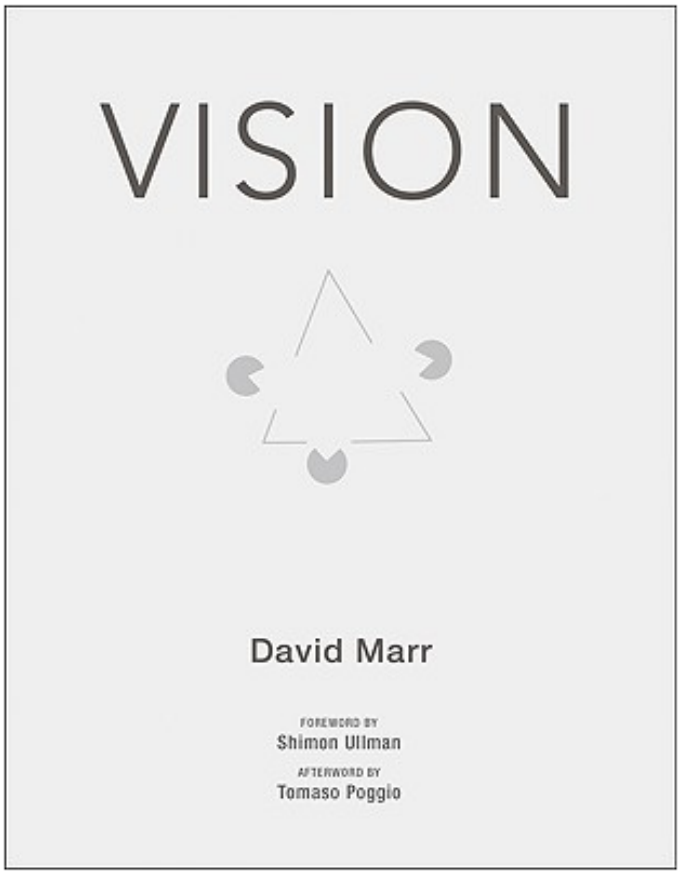
Historic perspective

- Computer Vision became a scientific discipline during the 70's.
- At that time, computing and memory resources were a major limitation factor for research and because of this, earlier researchers focused their work on the development of abstract category-based features and specific programs for their processing.
- This approach was developed for idealized visual worlds (such as the block world) and became obsolete when images of the real world became available to the computers.

The 70's


-23-444(1a-d)
(a) Original picture.
(b) Computer display of picture (reflected by mistake).
(c) Differentiated picture.
(d) Feature points selected.

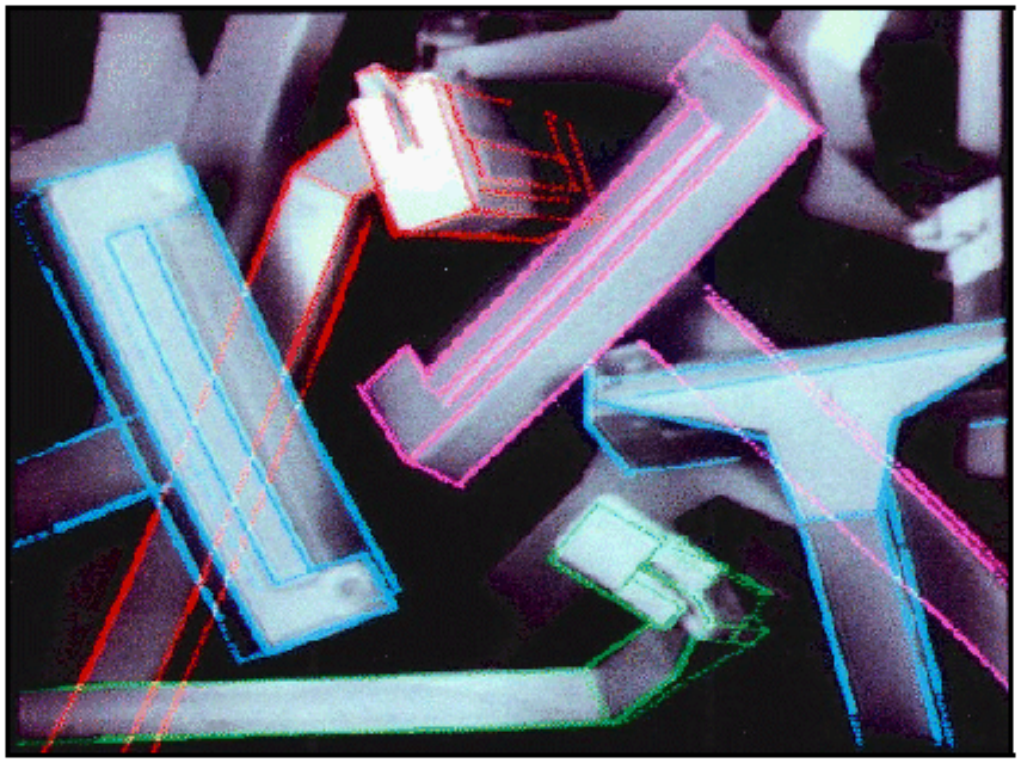
Historic perspective: 1982



Historic perspective

- In the 80's, the visual recognition problem was approached with the use of expert systems (or rule based systems) that were able to work with simple textureless images.
- Objects were represented by abstract exemplar-based features (such as contours) that were the result of simple perceptual organization processes.

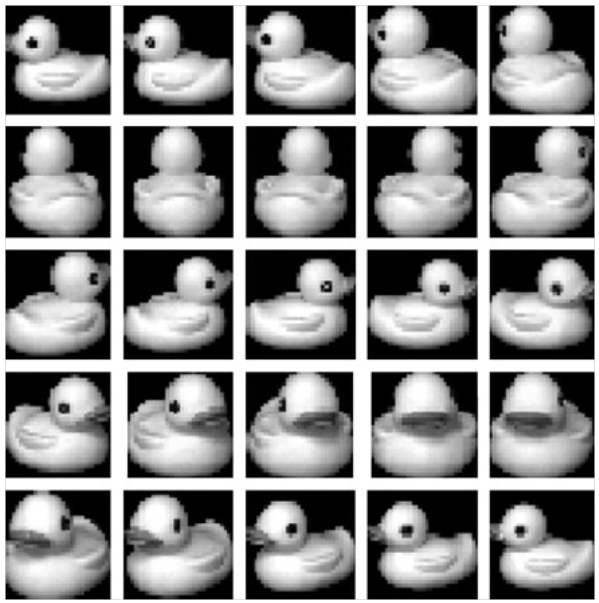
The 80's

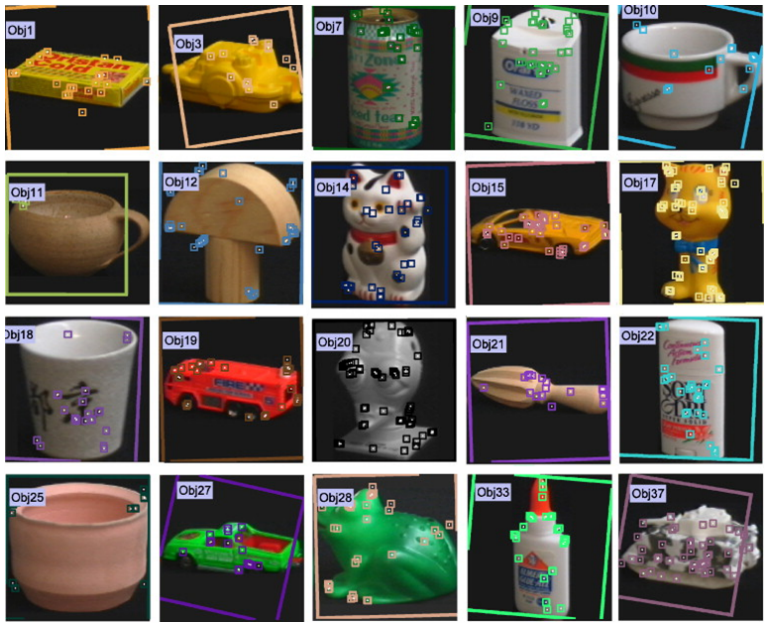


Historic perspective

- The next step was started in the 90's by the publication of public domain databases that depicted sets of non-cluttered complex objects such as cars, toys, etc, producing a new approach that became hegemonic until our days: the *appearance-based representation*.
- This new approach was rapidly adopted by most of the research community due to the performance it showed in problems that were regarded as hard until then, such as the face detection and recognition problem.
- A key component of this success was the intensive application of machine learning and statistical methods to this kind of visual representations.

The 90's




Obj1
Obj3
Obj7
Obj9
Obj10
Obj11
Obj12
Obj14
Obj15
Obj17
Obj18
Obj19
Obj20
Obj21
Obj22
Obj25
Obj27
Obj28
Obj33
Obj37

How to do it?

We can think abut the object as whole, but it is known that the visual system can use local, informative image *fragments* of a given object, rather than the whole object, to classify it into a familiar category.

This approach has some advantages over holistic methods developed in the 80's and 90's (f.e. face detection and recognition).

Recognition with Local Features

Our reading algorithm

I cnduo't bvleiee taht I culod aulaclty uesdtannrd waht I was rdnaieg. Unisg the icndeblire pweor of the hmuan mnid, aocdcrnig to rseecrah at Cmabrigde Uinervtisy, it dseno't mttaer in waht oderr the lterets in a wrod are, the olny irpoamtnt tihng is taht the frsit and lsat ltteer be in the rhgit pclae. The rset can be a taotl mses and you can sitll raed it whoutit a pboerlm. Tihs is bucseae the huamn mnid deos not raed ervey ltteer by istlef, but the wrod as a wlohe. Aaznmig, huh? Yaeh and I awlyas tghhuot slelinpg was ipmorantt! See if yuor fdreins can raed tihs too.

Recognition with Local Features

How do we represent *words* as a whole?

- One plausible explanation is that we use Open N-Grams.
- An Open *N*-Gram is a binary detector for a specific spatial configuration of *N* letters. For $n = 2$,

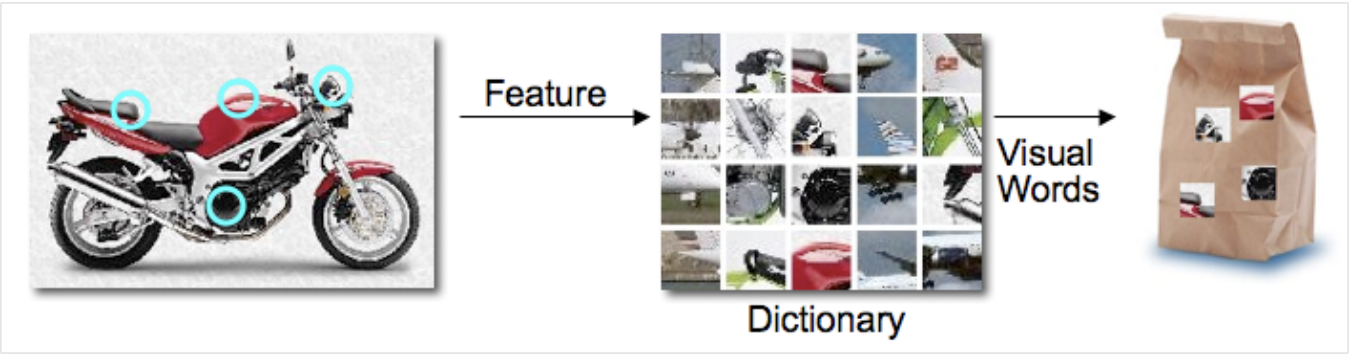
$$\texttt{hello} = \{\ \texttt{\_h},\ \texttt{he},\ \texttt{hl},\ \texttt{ho},\ \texttt{el},\ \texttt{eo},\ \texttt{ll},\ \texttt{lo},\ \texttt{o\_}\ \}$$
$$\texttt{hlelo} = \{\ \texttt{\_h},\ \texttt{hl},\ \texttt{he},\ \texttt{ho},\ \texttt{le},\ \texttt{ll},\ \texttt{lo},\ \texttt{el},\ \texttt{eo},\ \texttt{o\_}\ \}$$

- It is useful to consider the beginning and the end of a word as a special case.
- This representation is able to recognize 95% of the words by using a simple nearest neighbor rule where the distance is defined as the number of elements in common.

Recognition with Local Features

The most basic approach is called the *bag of words* approach (it was inspired in techniques used by the natural language processing community).

Recognition with Local Features



Main assumptions: Independent features + Histogram representation.

Recognition with Local Features

A more advanced approach involves several steps:

1. Find image locations where we can reliably find correspondences with other images.
2. Image content is transformed into local features (that are invariant to translation, rotation, and scale).
3. Verify if they belong to a consistent configuration

Recognition with Local Features

A wonderful example of these stages can be found in David Lowe's (2004) "Distinctive image features from scale-invariant keypoints" paper, which describes the development and refinement of his Scale Invariant Feature Transform (SIFT).

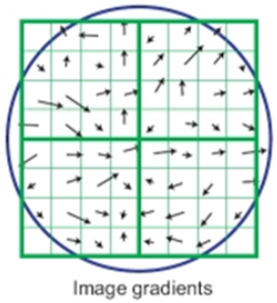
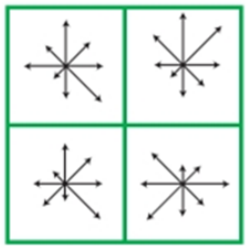


Image gradients

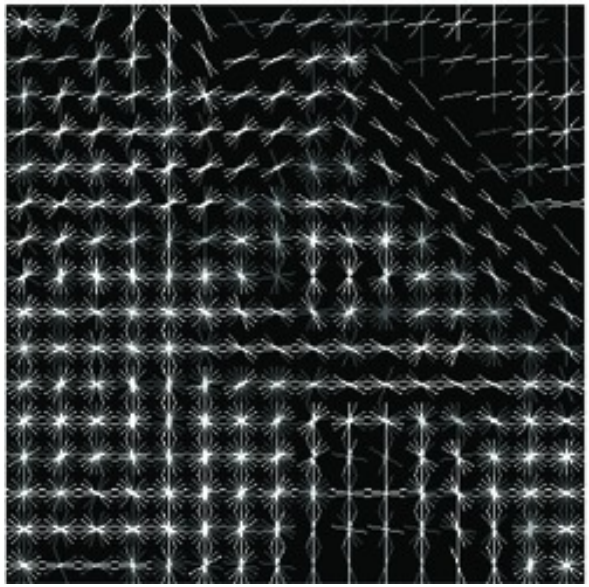


Keypoint descriptor

Recognition with Local Features

- Today, the state of the art for object category representation is based on the *Histogram of Oriented Gradients* (HoG) feature, developed by Dalal and Triggs in 2006.
- HOG decomposes an image into small squared cells, computes an histogram of oriented gradients in each cell, normalizes the result using a block-wise pattern, and return a descriptor for each cell.
- Stacking the cells into a squared image region can be used as an image window descriptor for object detection, for example by means of a SVM.

HoG



HoG

- HOG exists in many variants. The most important are: the UoCTTI variant and the original Dalal-Triggs variant.
- The main difference is that the UoCTTI variant computes both directed and undirected gradients as well as a four dimensional texture-energy feature, but projects the result down to 31 dimensions.
- Dalal-Triggs works instead with undirected gradients only and does not do any compression, for a total of 36 dimension.

HoG

The main HoG parameters are:

- The number of cells ($n \times m$) for representing a model.
- The number of orientations in the histograms.
- The bilinear orientation assignment of the gradient

Support Vector Machines

Problem: Binary Classification

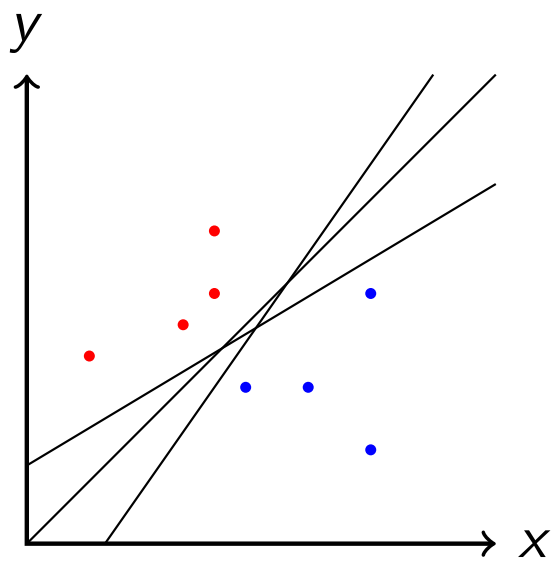
Let $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ be a set of linearly separable data points $\mathbf{x}_i \in \mathbb{R}^D$ having corresponding labels $y_i \in \{+1, -1\}$.

- Find the *best* vector $\mathbf{w}$ for computing a linear decision function of the form $f(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x} - b)$.

Observation: $\mathbf{w}$ (weight vector) determines a discriminant $(D-1)$-dimensional hyperplane and the scalar b (bias) the offset of the hyperplane with respect to the origin.

Which is the best?

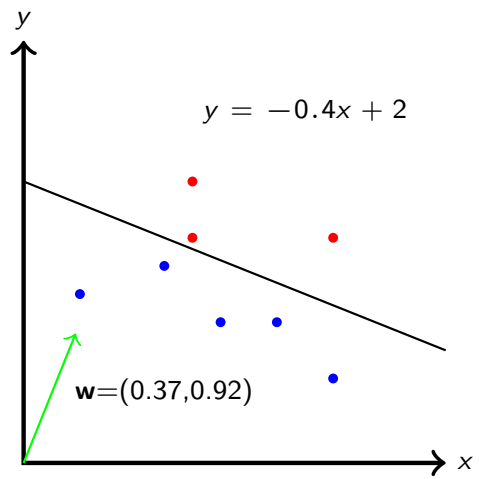
There are infinite separating hyperplanes. A separating hyperplane with no margin can be misleading.



An intuitive solution to this problem is separating data points with a hyperplane that has the *largest margin*.

Which is the best?

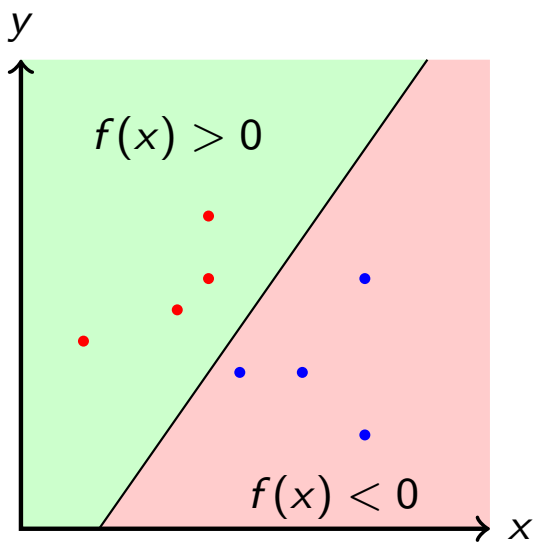
A decision hyperplane **w** can be defined by an intercept term w_0 and a normal vector $\{w_1, \ldots, w_D\}$ which is perpendicular to the hyperplane.



Which is the best?

This hyperplane partitions the data space in two half-spaces by means of $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$.

Note: In order to simplify notation we implicitly consider the bias by writting $\mathbf{x} = (1, x_1, \ldots, x_D)$ *and* $\mathbf{w} = (w_0, \ldots, w_D)$



The problem is how to find (the best) $\mathbf{w}$.

Which is the best?

To find the best $\mathbf{w}$ we will use two different criteria:

1. Enforce $sign\langle \mathbf{w}, \mathbf{x}_i \rangle = y_i$ for $i = 1, \ldots, N$.
2. Try to ensure $sign\langle \mathbf{w}, \mathbf{x} \rangle = y$ for the unseen $(\mathbf{x}, y)$ as well.

If we assume that $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ are i.i.d samples from a distribution, then futures samples will be similar to current ones. For this reason the second criterion can be rewritten in this way:

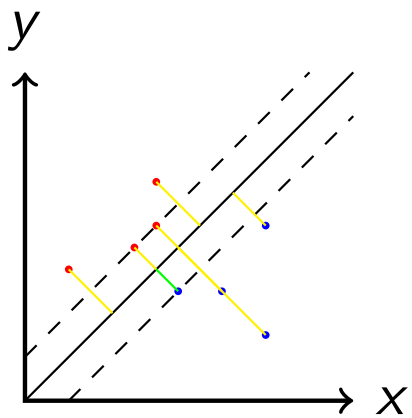
1. Enforce $sign\langle \mathbf{w}, \mathbf{x}_i \rangle = y_i$ for $i = 1, \ldots, N$.
2. Find $\mathbf{w}$ such that we can maximally perturb the input samples without introducing misclassifications.

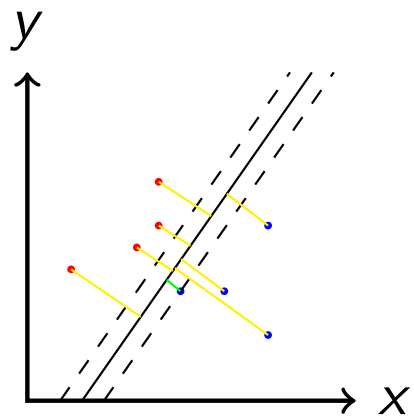
Robustness = Large Margin

- The margin of a sample, $margin(\mathbf{x})$ can be defined as the distance of $\mathbf{x}$ to the decision plane $\mathbf{w}$, $\langle \frac{\mathbf{w}}{||\mathbf{w}||}, \mathbf{x} \rangle$.
- The margin of a decision plane can be defined as

$$\min_{\mathbf{x} \in \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}} margin(\mathbf{x})$$

.





Which is the best?

The *maximum margin solution* can be determined by solving this optimization problem:

$$\max_{\upsilon \in \mathbb{R}^+, \mathbf{w} \in \mathbb{R}^D} \upsilon$$

subject to

$$sign\langle \mathbf{w}, \mathbf{x}_i \rangle = y_i \quad \text{for } i = 1, \ldots, N \tag{1}$$

$$\left| \langle \frac{\mathbf{w}}{||\mathbf{w}||}, \mathbf{x}_i \rangle \right| \geq \upsilon \quad \text{for } i = 1, \ldots, N \tag{2}$$

Which is the best?

That is equivalent to:

$$\max_{\upsilon \in \mathbb{R}^+, \mathbf{w} \in \mathbb{R}^D, ||\mathbf{w}||^2 = 1} \upsilon$$

subject to

$$y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq \upsilon \text{ for } i = 1, \ldots, N$$

Which is the best?

That is equivalent to:

$$\max_{\upsilon \in \mathbb{R}^+, \mathbf{w} \in \mathbb{R}^D, ||\mathbf{w}||^2 = 1} \upsilon$$

subject to

$$y_i \langle \frac{\mathbf{w}}{\upsilon}, \mathbf{x}_i \rangle \geq 1 \text{ for } i = 1, \ldots, N$$

Support Vector Machine

We can define $\hat{\mathbf{w}} = \frac{\mathbf{w}}{v}$ so that $v = \frac{1}{||\hat{\mathbf{w}}||}$. Then, the maximization problem becomes a minimization problem:

SVM for linearly separable data

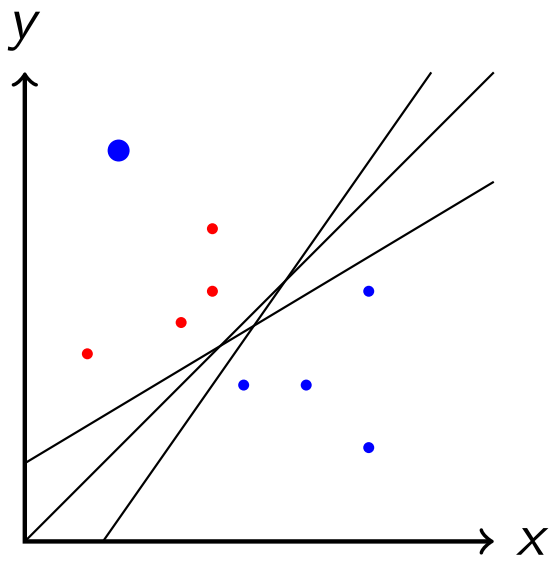
$$\min_{\hat{\mathbf{w}} \in \mathbb{R}^N} ||\hat{\mathbf{w}}||^2$$

subject to

$$y_i \langle \hat{\mathbf{w}}, \mathbf{x}_i \rangle \geq 1 \text{ for } i = 1, \ldots, N$$

This is an easy problem (the objective function is differentiable and convex) that can be solved using standard software.

The case of non separable data



The case of non separable data

- In general, if data is non-separable this means that for at least one data point, say x_r, the term $y_r(\mathbf{w} \cdot \mathbf{x}_r)$ will fail to exceed the threshold value 1 no matter what $\mathbf{w}$ is set to.
- Although we might perhaps like to minimize the number of misclassified points, this is an NP-hard problem. Instead, to handle the case of points on the wrong side of the fence, SVMs introduce slack variables.
- These variables allow all the constraints to be satisfied.
- A corresponding slack variable penalty term is added to the objective, such that the more the slack variables are relied upon, the worse the value of the objective.

The case of non separable data

This SVM variant takes a parameter C that determines how hard points violating the constraint should be penalized. This parameter appears in the objective function of the problem, which now is formulated as:

$$\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}|| + \frac{C}{N}\sum_i \xi_i$$

$$\text{s.t. } y_i(\mathbf{x}_i \cdot \mathbf{w}) \geq 1 - \xi_i, \xi_i \geq 0, \forall i \in \{1, \ldots, N\}.$$

Large C means high penalty, and in the limit $C \to \infty$ we obtain the separable case.

The case of non separable data

Observe that if $y_j(\mathbf{w} \cdot \mathbf{x}_j) \geq 1$, then $\xi_j = 0$ and in this case there is no contribution to the penalty term, but if margin $y_j(\mathbf{w} \cdot \mathbf{x}_j) < 1$, then $\xi_j > 0$ and the penalty terms increases by $\frac{C}{N}\xi_j$:

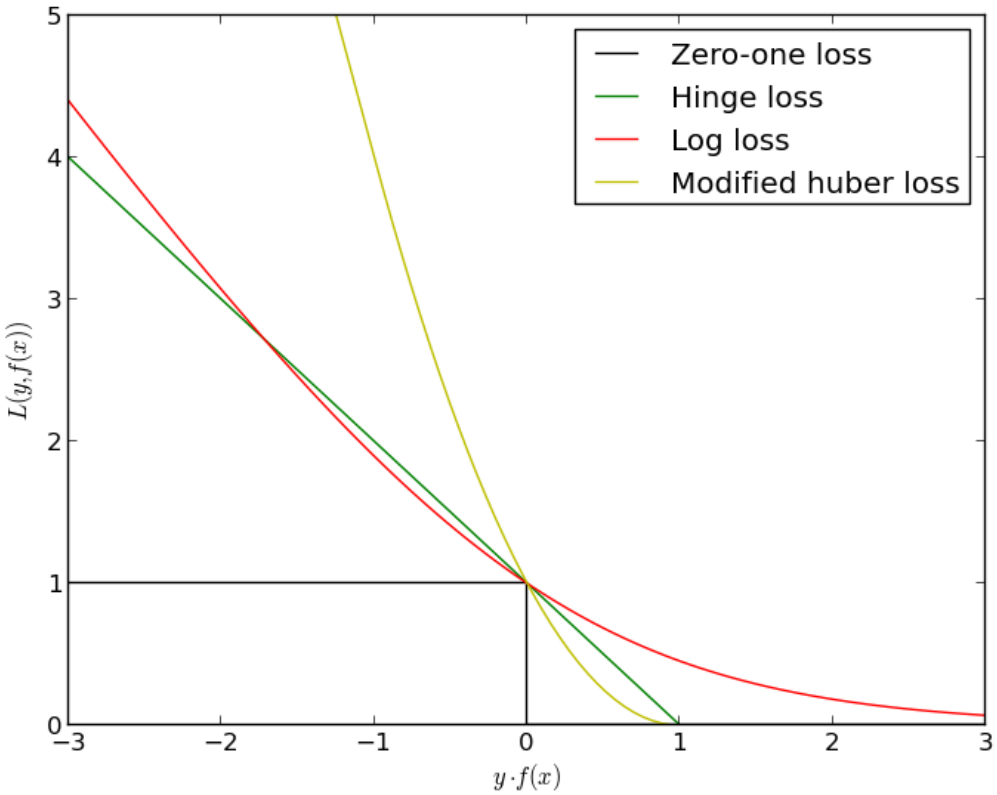
$$\xi_j = \max(0, 1 - y_i(\mathbf{w} \cdot \mathbf{x}_j))$$

Hinge Loss

This corresponds to a general loss function called *Hinge Loss*:

$$L(y, f(\mathbf{x})) = \max(0, 1 - yf(\mathbf{x}))$$

Hinge Loss


Zero-one loss
Hinge loss
Log loss
Modified huber loss
$L(y, f(x))$
$y \cdot f(x)$
5
4
3
2
1
0
−3
−2
−1
0
1
2
3

Primal form

- The SVM formulation we have seen is called the *primal* form of the SVM problem.
- The application of the Lagrangian method to this problem, by introducing one dual variable α_i for each constraint (i.e. for each training point), allows us to express this minimization problem as:

$$L(\mathbf{w}, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{N} \alpha_i (y_i(\mathbf{w} \cdot \mathbf{x}_i) - 1)$$

- This problem can be solved by standard *quadratic programming techniques*.

QP

QP

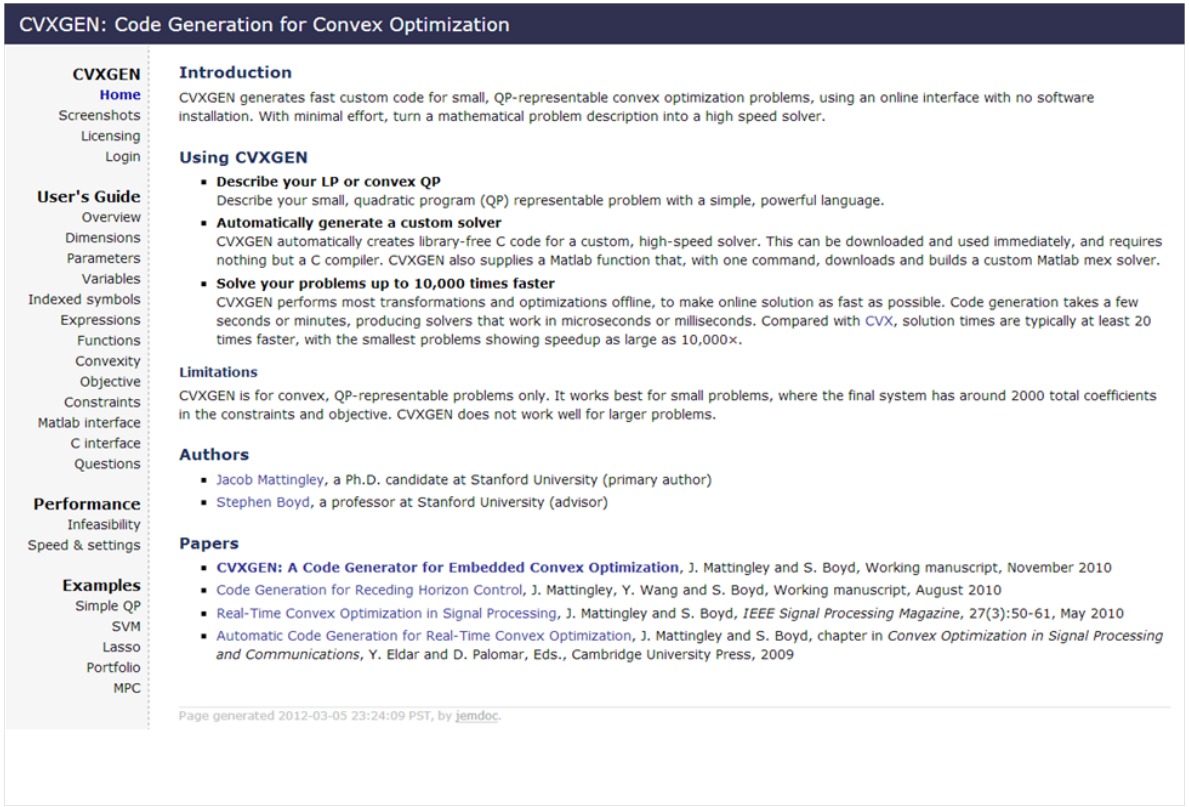
Quadratic programming (QP) is a special type of mathematical optimization problem. It is the problem of optimizing (minimizing or maximizing) a quadratic function of several variables subject to linear constraints on these variables.

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x}$$

s.t.

$$A\mathbf{x} \leq b, E\mathbf{x} = d.$$

QP



CVXGEN: Code Generation for Convex Optimization

CVXGEN
Home
Screenshots
Licensing
Login

User's Guide
Overview
Dimensions
Parameters
Variables
Indexed symbols
Expressions
Functions
Convexity
Objective
Constraints
Matlab interface
C interface
Questions

Performance
Infeasibility
Speed & settings

Examples
Simple QP
SVM
Lasso
Portfolio
MPC

Introduction

CVXGEN generates fast custom code for small, QP-representable convex optimization problems, using an online interface with no software installation. With minimal effort, turn a mathematical problem description into a high speed solver.

Using CVXGEN

- **Describe your LP or convex QP**
  Describe your small, quadratic program (QP) representable problem with a simple, powerful language.
- **Automatically generate a custom solver**
  CVXGEN automatically creates library-free C code for a custom, high-speed solver. This can be downloaded and used immediately, and requires nothing but a C compiler. CVXGEN also supplies a Matlab function that, with one command, downloads and builds a custom Matlab mex solver.
- **Solve your problems up to 10,000 times faster**
  CVXGEN performs most transformations and optimizations offline, to make online solution as fast as possible. Code generation takes a few seconds or minutes, producing solvers that work in microseconds or milliseconds. Compared with CVX, solution times are typically at least 20 times faster, with the smallest problems showing speedup as large as 10,000×.

Limitations

CVXGEN is for convex, QP-representable problems only. It works best for small problems, where the final system has around 2000 total coefficients in the constraints and objective. CVXGEN does not work well for larger problems.

Authors

- Jacob Mattingley, a Ph.D. candidate at Stanford University (primary author)
- Stephen Boyd, a professor at Stanford University (advisor)

Papers

- **CVXGEN: A Code Generator for Embedded Convex Optimization**, J. Mattingley and S. Boyd, Working manuscript, November 2010
- Code Generation for Receding Horizon Control, J. Mattingley, Y. Wang and S. Boyd, Working manuscript, August 2010
- Real-Time Convex Optimization in Signal Processing, J. Mattingley and S. Boyd, *IEEE Signal Processing Magazine*, 27(3):50-61, May 2010
- Automatic Code Generation for Real-Time Convex Optimization, J. Mattingley and S. Boyd, chapter in *Convex Optimization in Signal Processing and Communications*, Y. Eldar and D. Palomar, Eds., Cambridge University Press, 2009

Page generated 2012-03-05 23:24:09 PST, by jemdoc.

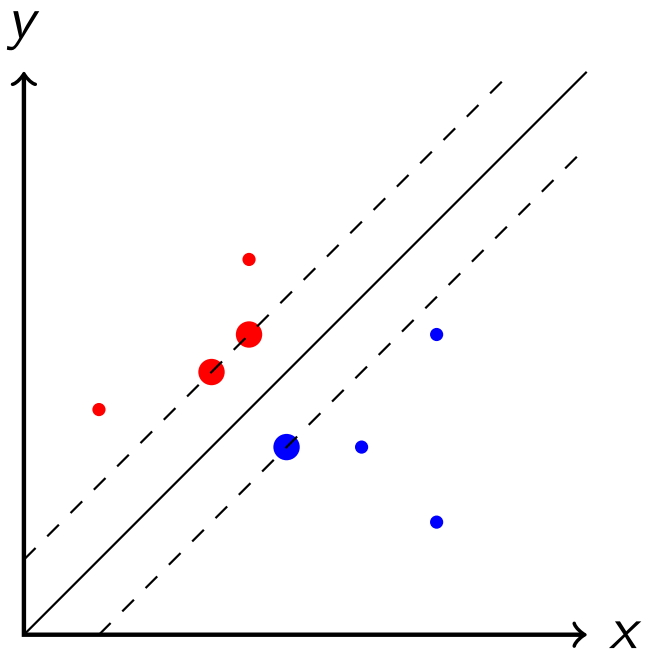
Primal form

- There is an interesting theoretical result about this class of problems (Karush-Kuhn-Tucker condition): the solution can be expressed as a linear combination of the training vectors.

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$

- Only a few α_i will be greater than 0, and their corresponding $\mathbf{x}_i$ are called the *support vectors* of the solution.
- The support vectors lie in the margin and satisfy $y_i(\mathbf{w}\mathbf{x}_i) = 1$.

Support Vectors



Dual formulation

- Using the fact that $||\mathbf{w}||^2 = \mathbf{w} \cdot \mathbf{w}$ and substituting $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$ we can write the dual form of the problem:

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \tag{3}$$

  under the constraints $\alpha_i \geq 0$ (for $i = 1, \ldots, N$) and $\sum_{i=1}^{N} \alpha_i y_i = 0$.
- This problem can also be easily solved using standard optimization software.
- $\mathbf{w}$ can be computed from α terms: $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$.

The kernel trick

- The original SVM algorithm, proposed by Vladimir Vapnik in 1963, was a linear classifier, but there is a way (the *kernel trick*) to design non-linear classifiers.
- The *trick* is based on the observation that in (3) all data terms are exclusively used to compute dot products:

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

- This fact suggest a way to compute SVM solutions in higher dimensional spaces: to find a function $k(\mathbf{x}_i, \mathbf{x}_j)$, called the *kernel*, that corresponds to the dot product of $\mathbf{x}_i$ and $\mathbf{x}_j$ in such a space.

The kernel trick

- In this case, the problem can be written as:

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

and the optimization problem does not change.

The kernel trick

These function exists, and some of the most used are:

- Polinomial kernel: $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^d$.
- Gaussian Radial Basis kernel: $k(\mathbf{x}_i, \mathbf{x}_j) = exp(-\lambda||(\mathbf{x}_i - \mathbf{x}_j)||^2)$ for $\lambda > 0$.
- Histogram Intersection kernel: $k(\mathbf{x}_i, \mathbf{x}_j) = exp(\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2})$.

The kernel trick

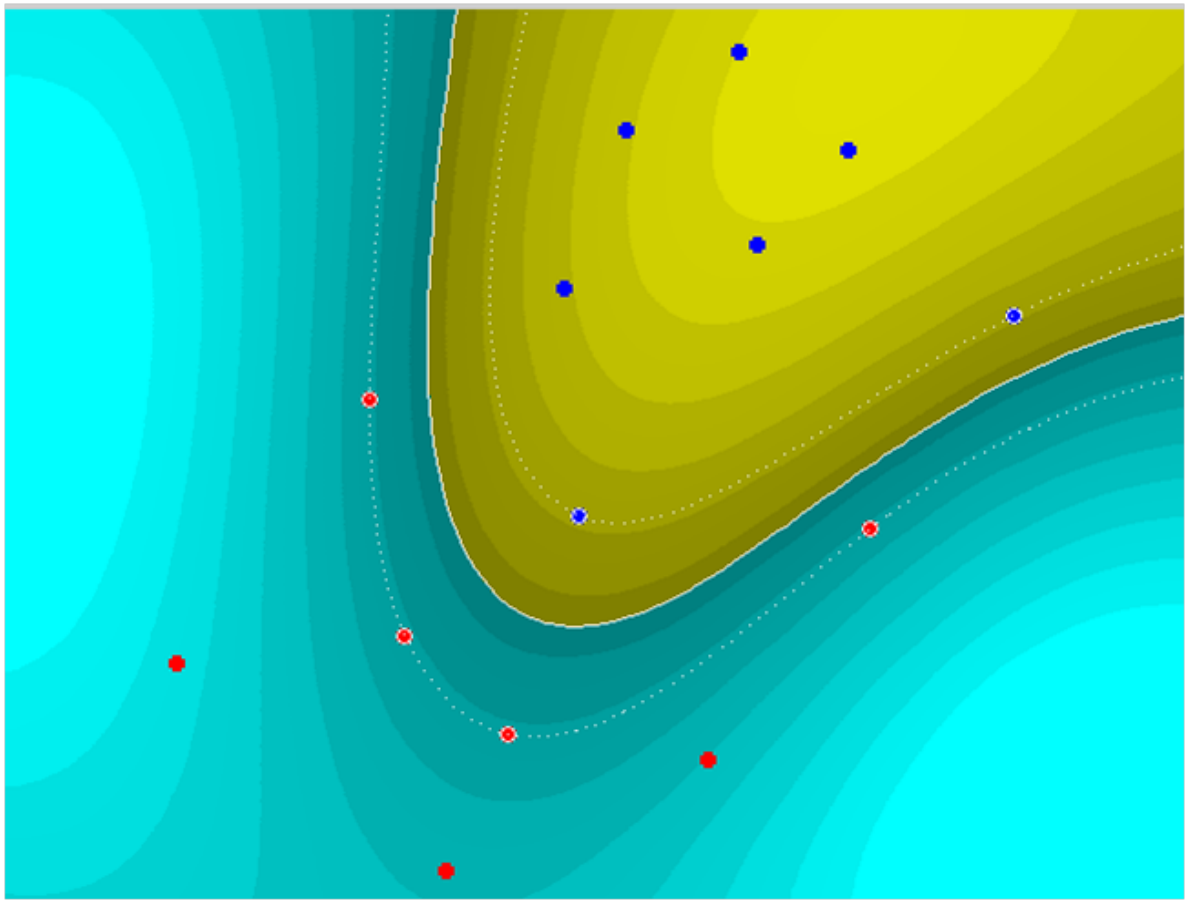


Figure : $k(\mathbf{x}_i, \mathbf{x}_j) = exp(-\lambda||(\mathbf{x}_i - \mathbf{x}_j)||^2)$ for $\lambda > 0$s

The kernel trick

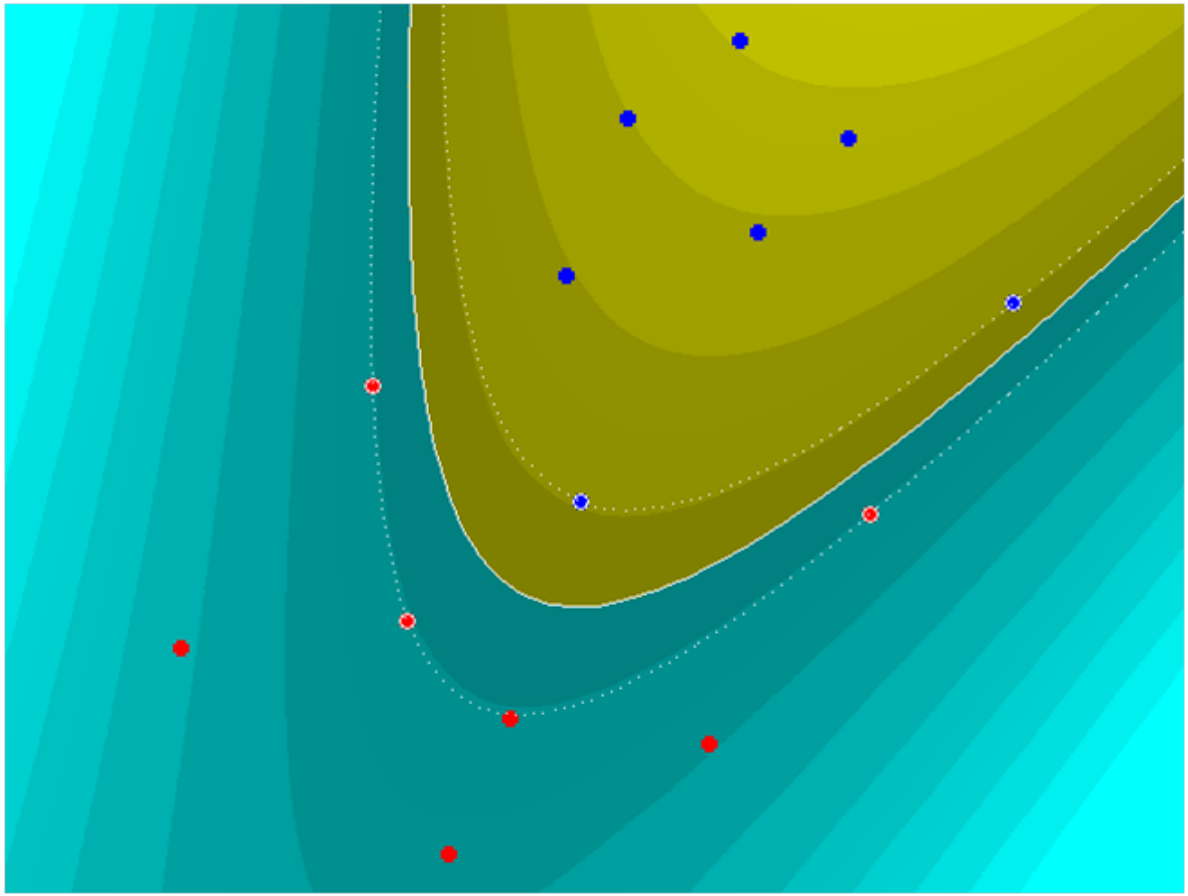


Figure : $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^2$

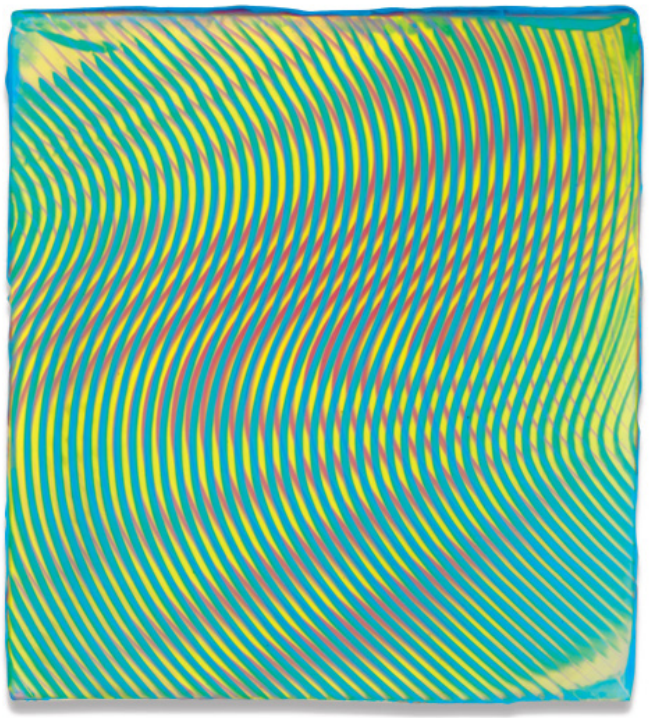
The kernel trick

Dot products with $\mathbf{w}$ for classification can again be computed by the kernel trick, i.e.

$$\mathbf{w} \cdot \phi(\mathbf{x}) = \sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x})$$

In general, there does not in general exist a value $\mathbf{w}'$ such that $\mathbf{w} \cdot \phi(\mathbf{x}) = k(\mathbf{w}', \mathbf{x})$.

SVM-based Object Recognition



Poselets + SVM

- One the most successful applications of SVM to object detection is called *Poselets* [1].

Poselets

A *poselet* is an image *fragment* that captures part of the pose of an object from a given viewpoint. Examples may differ visually but have common semantics.

- *Poselets* have been specially successful for detecting humans, but the same approach can be directly applied to other objects.

[1] L. Bourdev, J. Malik, Poselets: Body Part Detectors Trained Using 3D Human Pose Annotations, ICCV 2009.

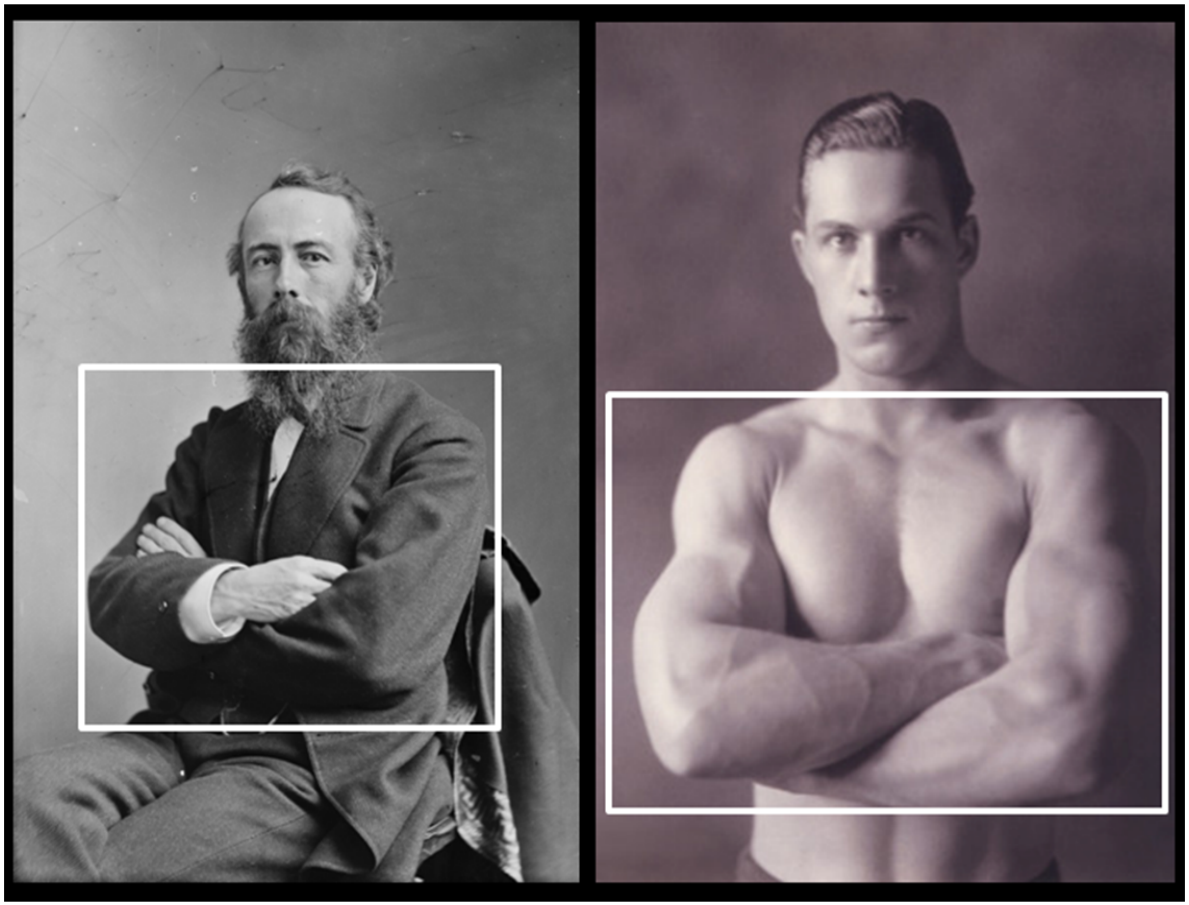
Poselets + SVM

Poselets + SVM

The first problem is how to create a training set of *poselets* to detect them in an image.

Let's suppose we have a large database depicting human bodies. Given a part of human pose, how do we find a similar pose configuration in the training set?

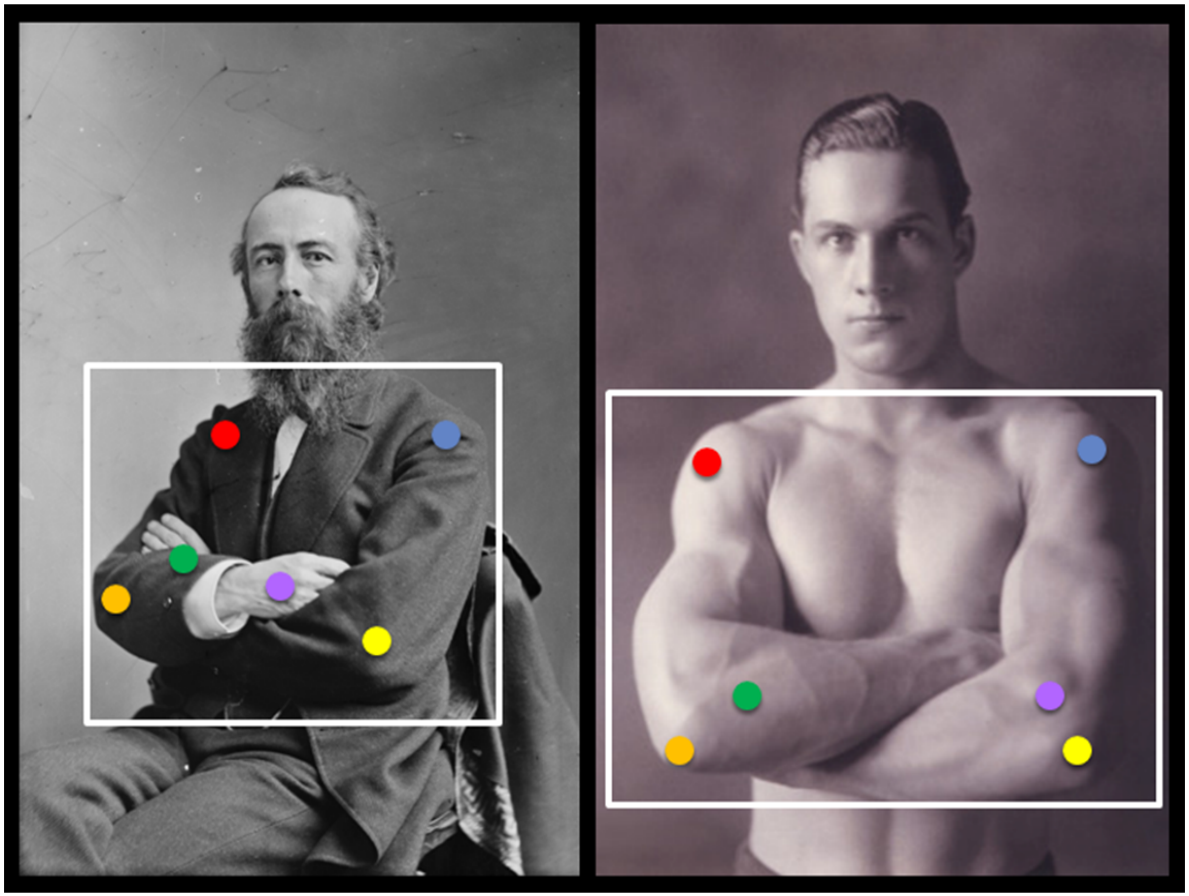
Poselets + SVM



Poselets + SVM

- We can use keypoints to annotate the joints, eyes, nose, etc. of people and then compute the *best* geometric correspondence between any two patches of the images.
- Then, we define a set of *meaningful* patches of every person by finding image fragments that include a minimum set of labeled points.

Poselets + SVM



Poselets + SVM

- In order to choose which poselets should we train, we can choose thousands of random windows, generate poselet candidates, train linear SVMs.
- Then, we can select a small set of poselets that are *individually effective* and *complementary*.

Poselets + SVM

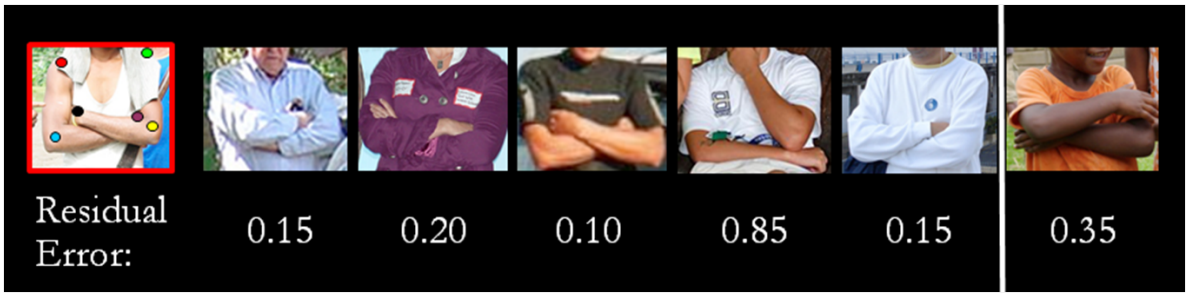
Algorithm 1: Poselet selection

repeat
  repeat
    Do a sliding window over the image;
    Estimate the residual of the transformation with a least squares fit $d_s(r) = \sum_i w_s(i)||x_s(i) - x_r(i)||^2(1 + h_{s,r}(i))$;
    Threshold the matches with a good fit;
  until *all images have been inspected*;
  Use them as positive training examples to train a linear SVM with HOG features.
until *all annotations have been processed*;

Poselets + SVM

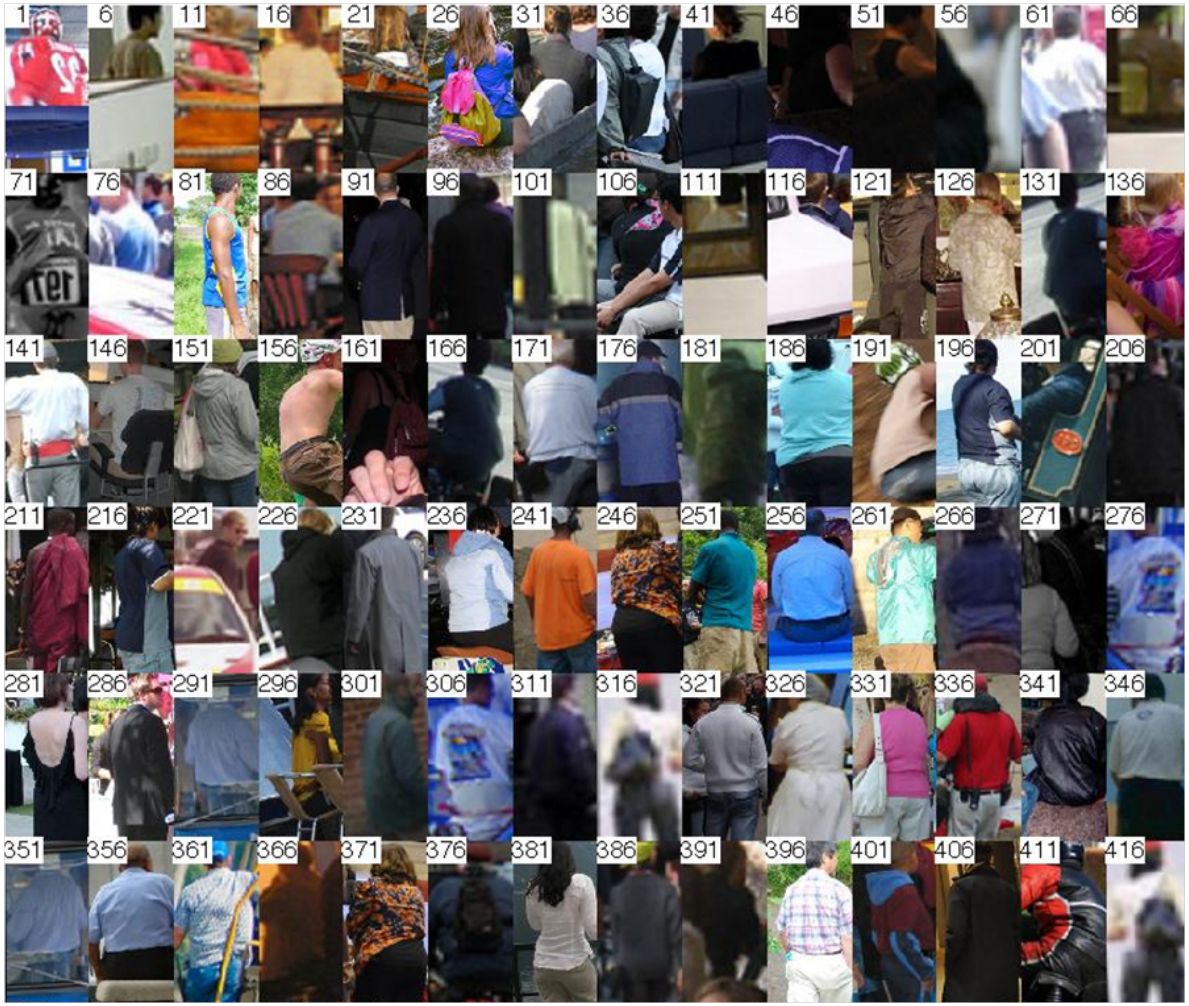
- $d_s(r)$ is the asymmetric distance in configuration space from example s to exemplar r.
- $\mathbf{x}_s(i) = (x, y, z)$ are the normalized 3D coordinates of the i-th point of s.
- The weight term $w_i(s)$ is a Gaussian with a mean in the center of the patch.
- $h_{r,i}$ is a penalty based on the visibility mismatch of keypoint i in the two examples. If i is visible or invisible in both examples, then $h_{r,i} = 0$. Otherwise, $h_{r,i} = a, a > 0$.

Poselets + SVM



Poselets + SVM

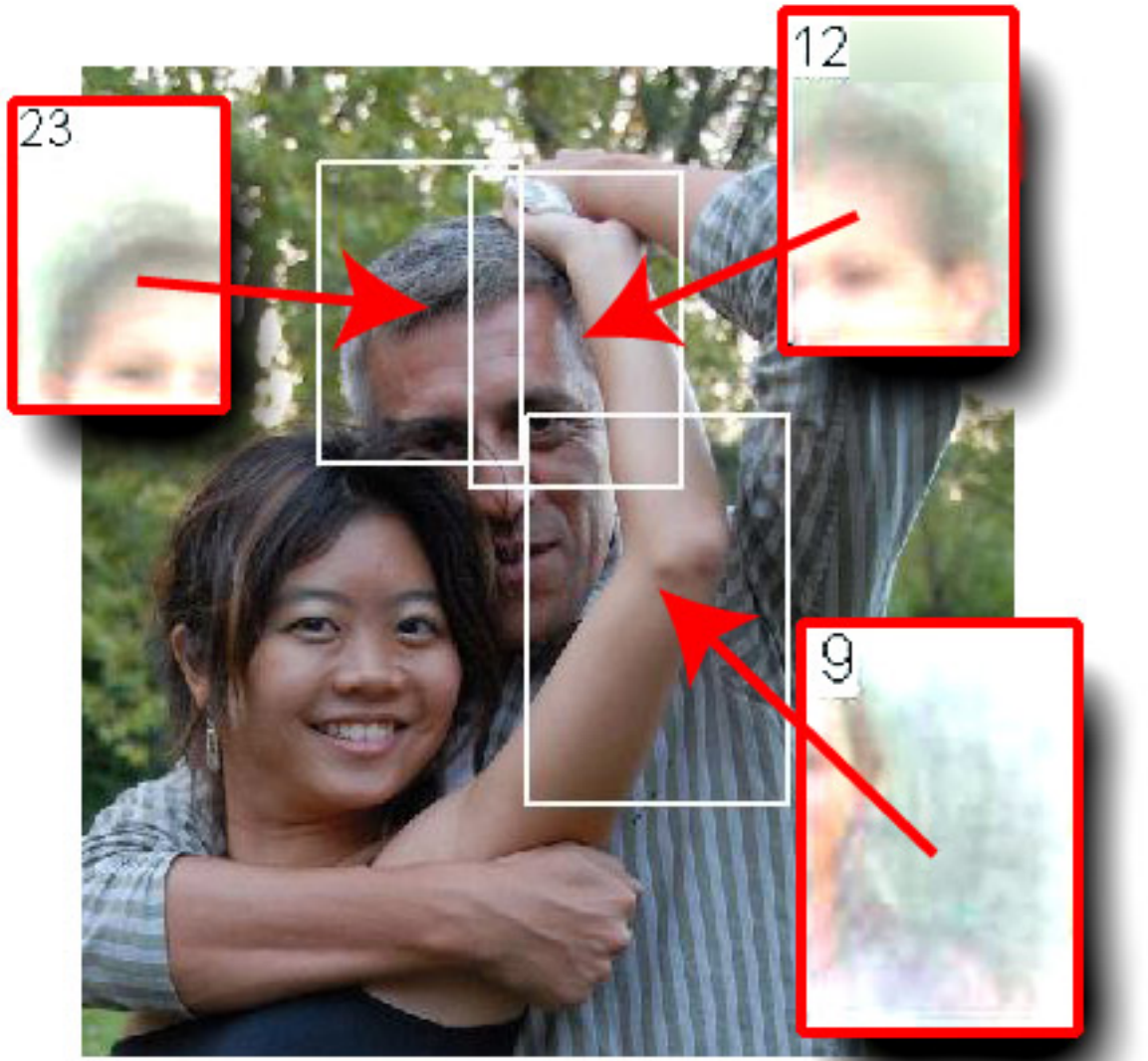
Poselets + SVM



Poselets + SVM

- A good poselet is easy to find (clusters in the appearance space) and determines a subset of the body configuration (clusters in 3D).
- If we run each poselet detector at every position and scale of the input image we can make hypothesis about the presence of an object.
- To this end, we collect all poselet detections and cluster nearby object/part landmark point hypothesis. Then, each cluster casts a vote for the object/part location.

Poselets + SVM



Poselets + SVM

- The probability of detecting the object O at position $\mathbf{x}$ is $P(O|\mathbf{x}) \propto \sum_i \mathbf{w}_i a_i(\mathbf{x})$, where $a_i(\mathbf{x})$ is the score that a poselet classifier assigns to location $\mathbf{x}$ and $\mathbf{w}_i$ is the weight of the poselet.
- To find the peaks in Hough space we cluster the cast votes using agglomerative clustering and we compute the sum over the poselets within each cluster.

Poselets + SVM

Poselets + SVM

- The weights **w** are used to account for the fact that (1) some poselets are more discriminative than others and (2) the responses of some subsets of poselets are redundant.
- Note also that we train separate weights for each task as weights are task-dependent. For example, a frontal face poselet is more discriminating for detecting an eye than for detecting an elbow.

Poselets + SVM

- We can use the *Max Margin Hough Transform* (M2HT) to learn the weights [2].
- When we compute object/part detection peaks in voting space, intuitively, some peaks in voting space correspond to true detections and others are false positives. M2HT is a discriminative technique that finds the set of weights that maximize the true positive peaks and minimize the false positives.

[2] S. Maji and J. Malik. Object Detection Using a Max-Margin Hough Transform. CVPR 2009.

Poselets + SVM

- To train for the weights, for each detection task we compute all voting peaks in the training set (with $\mathbf{w}_i = 1$).
- We compute a matrix A with A_i^j correponding to the score of poselet j when voting for peak i.
- Then we find $\mathbf{w}$ using the following optimization problem:

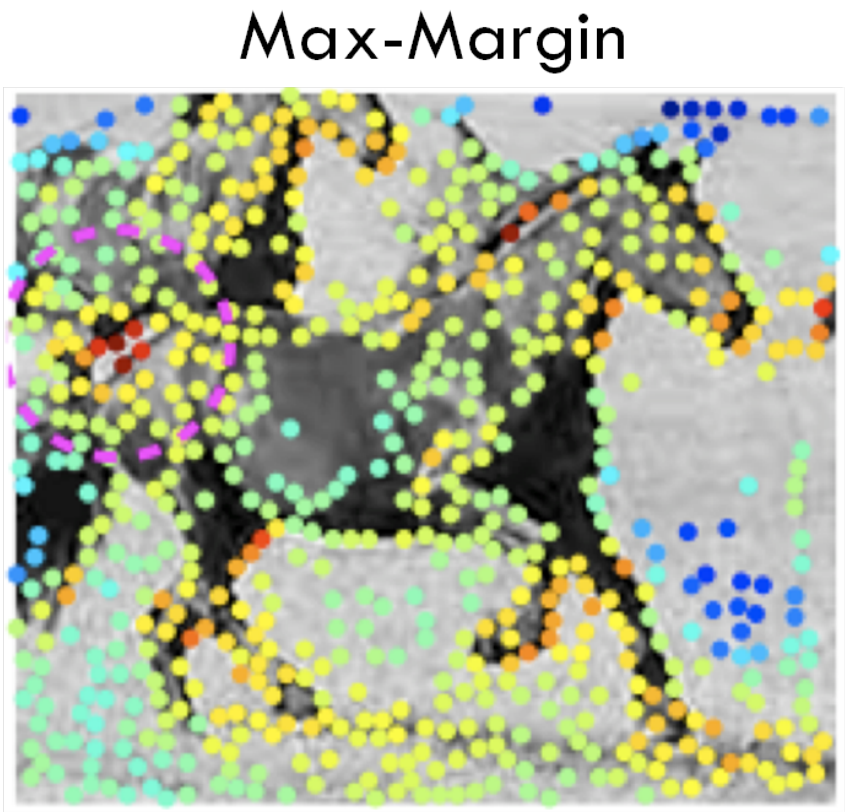
$$\min_{\mathbf{w},\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i$$

subject to

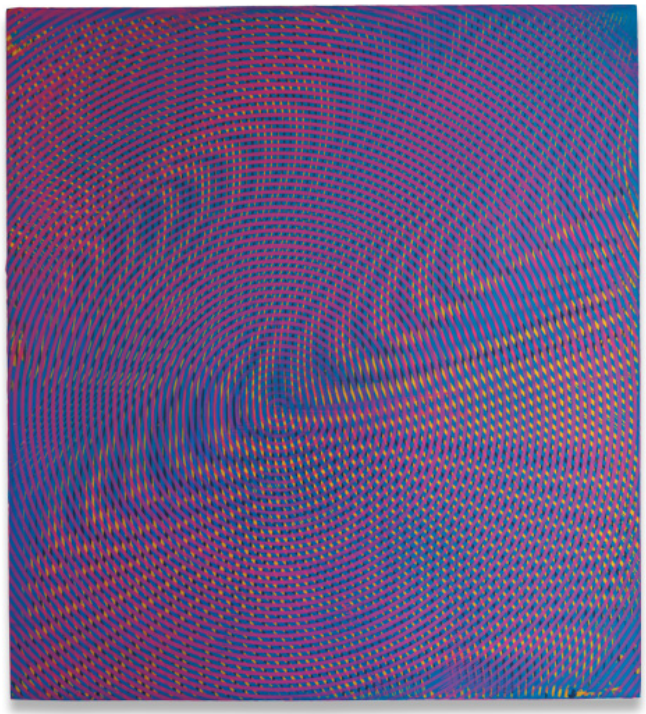
$$y_i\langle \mathbf{w}^T A_i\rangle \geq 1 - \xi_i, \mathbf{w} \geq 0, \xi_i \geq 0, \forall i = 1, \ldots, N.$$

where $y_i = 1$ if the peak is true positive and -1 if false positive.

Poselets + SVM



Pedestrian Detection



Pedestrian detection: HoG+ SVM

- **Histogram of Oriented Gradients** (HoG) descriptor by Dalal and Triggs [3] assumes that the local object appearance and shape within an image can be described by the distribution of occurrences of gradient orientation computed on a dense grid of uniformly spaced cells with overlapping local contrast normalization for improved accuracy.
- The implementation of these descriptors is achieved:
  - by dividing the image into small connected regions (cells)
  - computing a histogram of gradient directions (i.e. edge orientations) for the pixels within each cell.
  - normalizing histograms by the intensity of a block of cells.

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

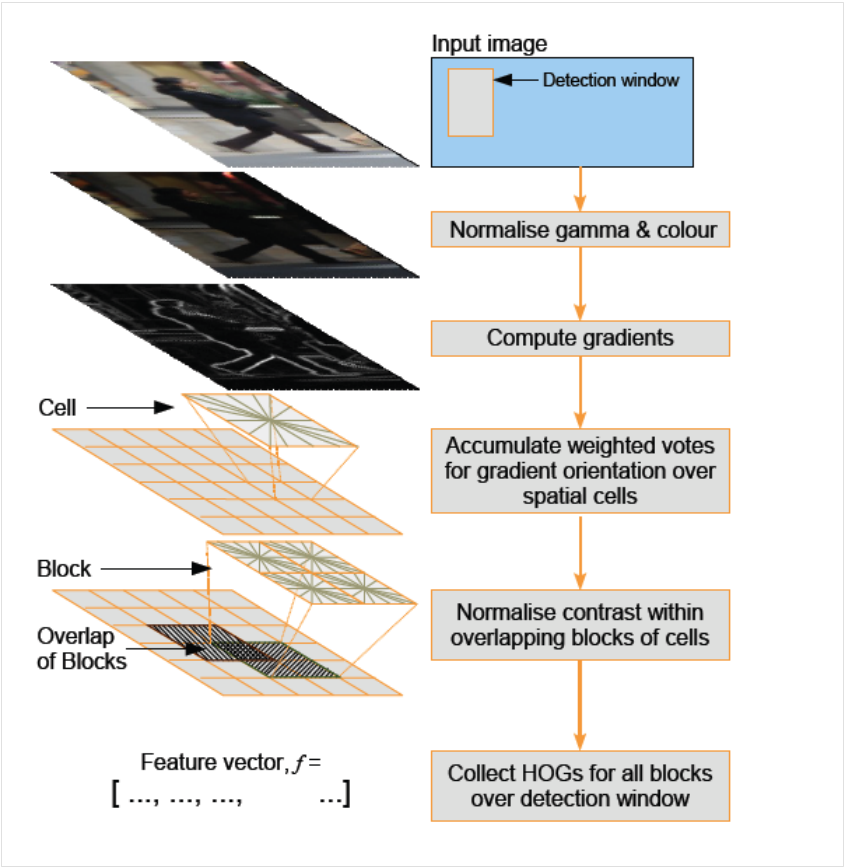
Pedestrian detection: HoG+ SVM

Pedestrian detection: HoG+ SVM

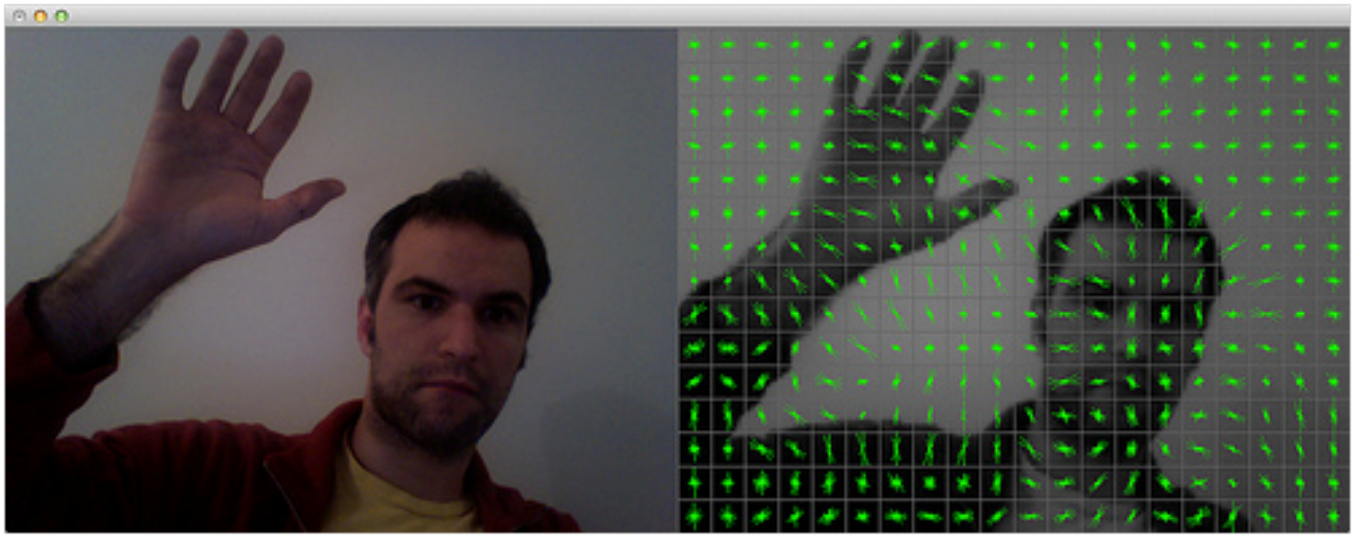
The steps are:

- Normalize the image (gamma and color).
- Compute de gradients.
- Perform a weighted voting into spatial and orientation cells.
- Normalize contrast over overlapping spatial blocks.
- Collect HoG over detection window.
- Training with a linear SVM.

Pedestrian detection: HoG+ SVM



Pedestrian detection: HoG+ SVM



Pedestrian detection: HoG+ SVM

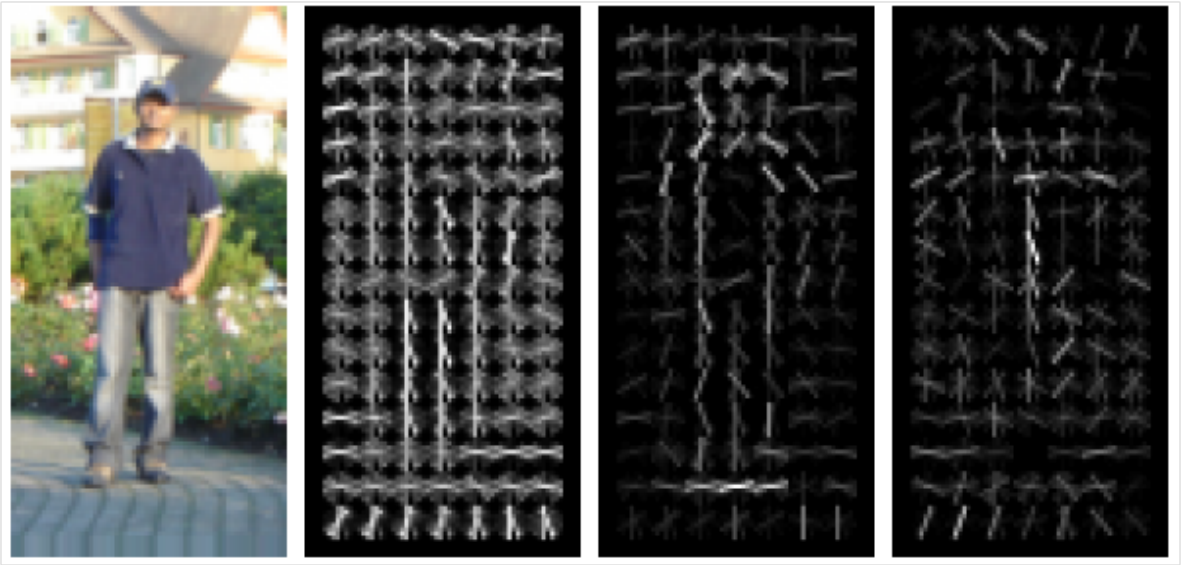


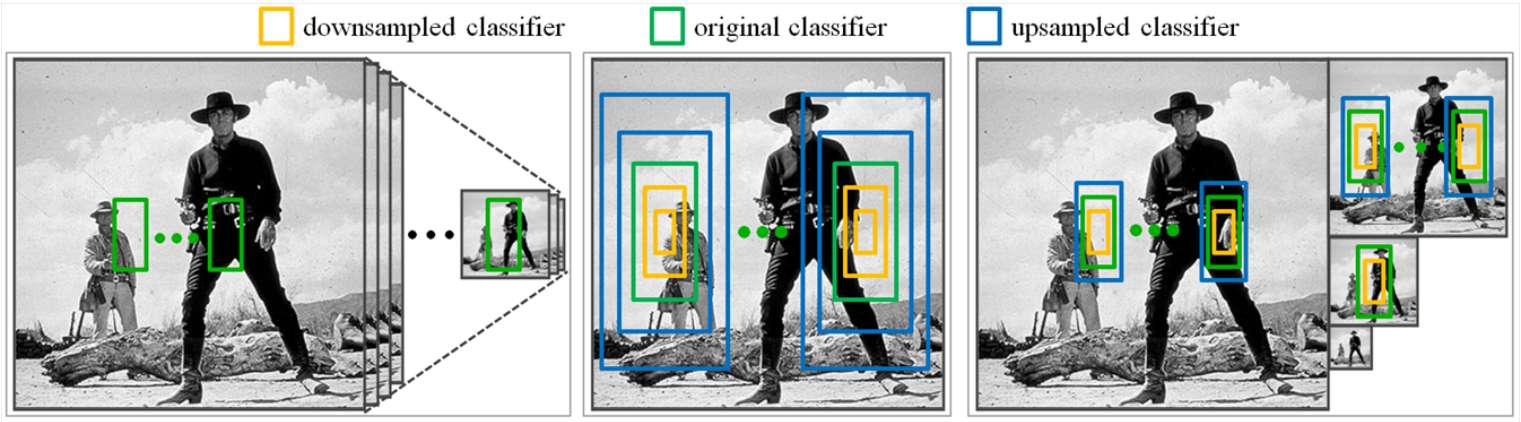
Figure : Original image, HoG representation of the image, SVM positive weights, SVM negative weights.

Pedestrian detection: HoG+ SVM

Pedestrian detector = Histogram of Oriented Gradients features computed across a multiscale pyramid (50 levels) + Linear SVM Model + A large dataset to learn.

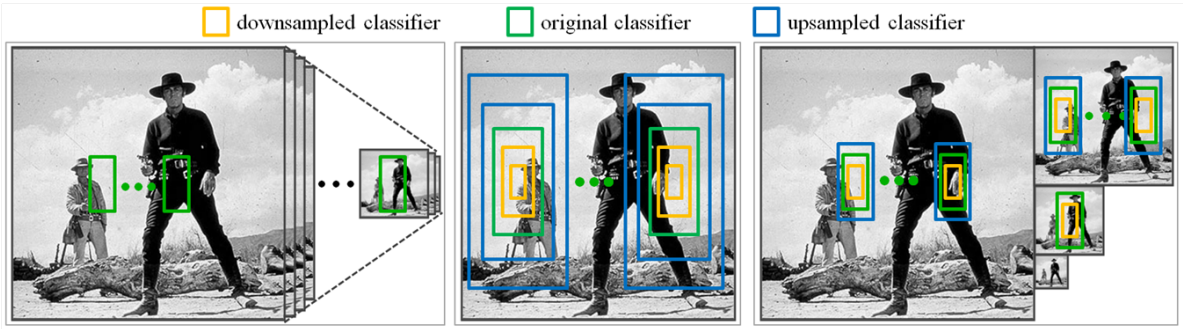
HoG + SVM: Performance issues

A standard pipeline for performing modern multiscale detection is to create a densely sampled image pyramid, compute features at each scale, and finally perform sliding window classification (with a fixed scale model).



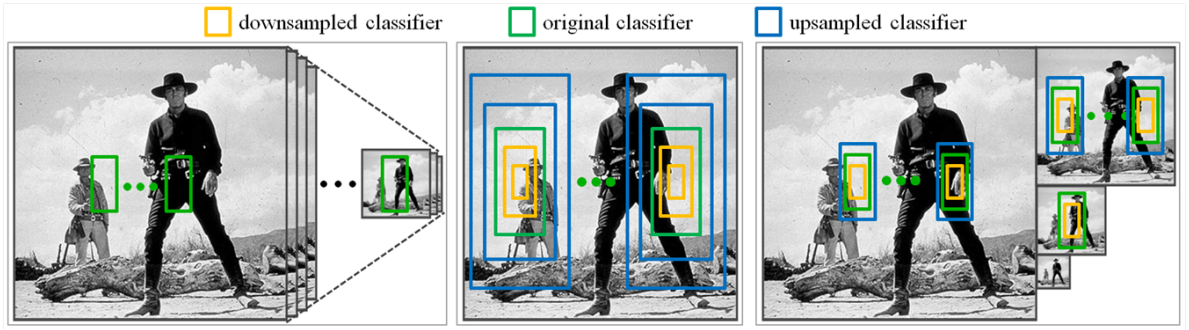
HoG + SVM: Performance issues

Viola and Jones utilized shift and scale invariant features, allowing a trained detector to be placed at any location and scale without relying on an image pyramid. Constructing such a classifier pyramid results in fast multiscale detection.
Unfortunately, most features are not scale invariant, including gradient histograms, significantly limiting the generality of this scheme.



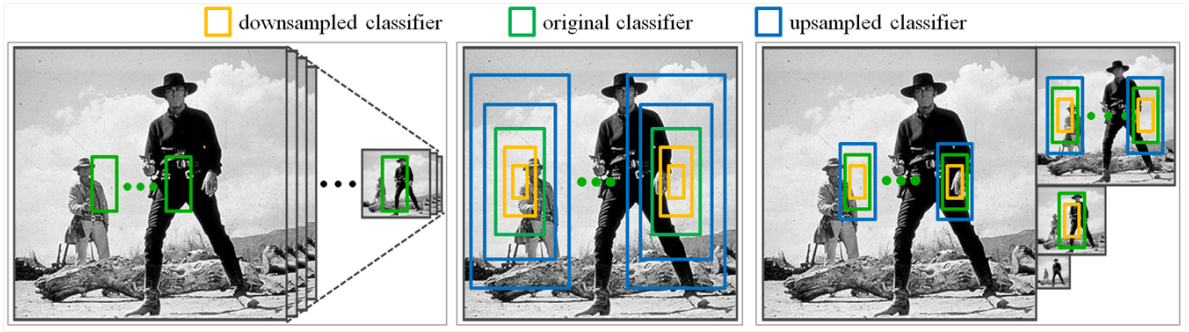
HoG + SVM: Performance issues

A faster alternative is to *approximate* features at multiple scales using a sparsely sampled image pyramid with a step size of an entire octave and within each octave we use a classifier pyramid.



HoG + SVM: Performance issues

This approach achieves nearly the same accuracy as using densely sampled image pyramids, with nearly the same speed as using a classifier pyramid applied to an image at a single scale.



HoG + SVM: Performance issues

Given gradients computed at one scale, is it possible to approximate gradient histograms at a different scale?

- Intuitively the information content of an **upsampled** image by a factor of k is the same as that of the original image (upsampling does not create new image structure).
- It can be easily shown that

$$\sum_{i=1}^{kn}\sum_{j=1}^{km} M_k(i,j) \approx \sum_{i=1}^{kn}\sum_{j=1}^{km} \frac{1}{k} M(i/k, j/k)$$

$$= k^2 \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{1}{k} M(i,j) = k \sum_{i=1}^{n}\sum_{j=1}^{m} M(i,j)$$

where M_k is the gradient magnitude in the upsampled image.

HoG + SVM: Performance issues

- Thus, the sum of gradient magnitudes in the pair of images is related by a factor of k.
- Therefore, according to the definition of gradient histograms, the relationship between an histogram bin h_q (computed in a window in the original image) and an histogram bin h'_q (computed over the *upsampled* window) is simply: $h'_q \approx k h_q$.

HoG + SVM: Performance issues

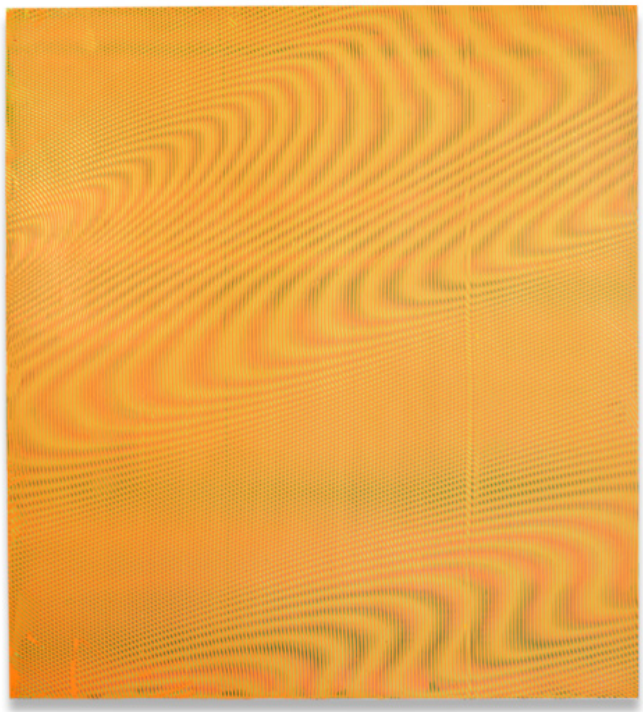
While the information content of an upsampled image is roughly the same as that of the original image, information is typically lost during **downsampling**. In general, downsampling results in loss of high frequency content and its gradient energy.

- Making use of statistical knowledge about natural images, we can expect that on average, the difference in channel energy between an image and a downsampled version of the image is independent of the scale of the original image and depends only on the relative scale between the pair of images.

HoG + SVM: Performance issues

- It can be shown that this assumtion implies that given a feature energy $f(I, 0)$ computed over I at the original scale is related to $f(I, s)$, the feature energy after downsampling by a factor of 2^s, in the following way: $f(I, s) \approx f(I, 0) a \exp^{\lambda s}$.
- λ and a depend on the type of feature and object type, but can be estimated from data.
- For oriented gradient magnitudes of pedestrian images, $a = 0.89$ and $\lambda = 1.099$.

From one model to N models.



HoG + Exemplar-SVM

There is a conceptually simple but surprisingly powerful method which combines the effectiveness of a discriminative object detector with the explicit correspondence offered by a nearest-neighbor approach.

- The method is based on training a separate linear SVM classifier for every exemplar in the training set. Each of these Exemplar-SVMs is thus defined by a single positive instance and millions of negatives.
- While each detector is quite specific to its exemplar, we empirically observe that an ensemble of such Exemplar-SVMs offers surprisingly good generalization.

HoG + Exemplar-SVM

- The Exemplar-SVM objective function is a variation of the linear SVM:

$$L_E(\mathbf{w}) = ||\mathbf{w}||^2 + C_1 h(\mathbf{w}^T \mathbf{x}_E) + C_2 \sum_{\mathbf{x} \in \mathcal{X}_{NE}} h(-\mathbf{w}^T \mathbf{x})$$

  where $h(\mathbf{x})$ is te hinge loss function and $\mathcal{X}_{NE}$ are the background samples.
- Because each Exemplar-SVM is defined by a single positive instance, we can use different features for each exemplar (f.e. the number of cells of its HoG descriptor).
- To detect an object, we should apply each Exemplar-SVM to test image in a sliding-window fashion.

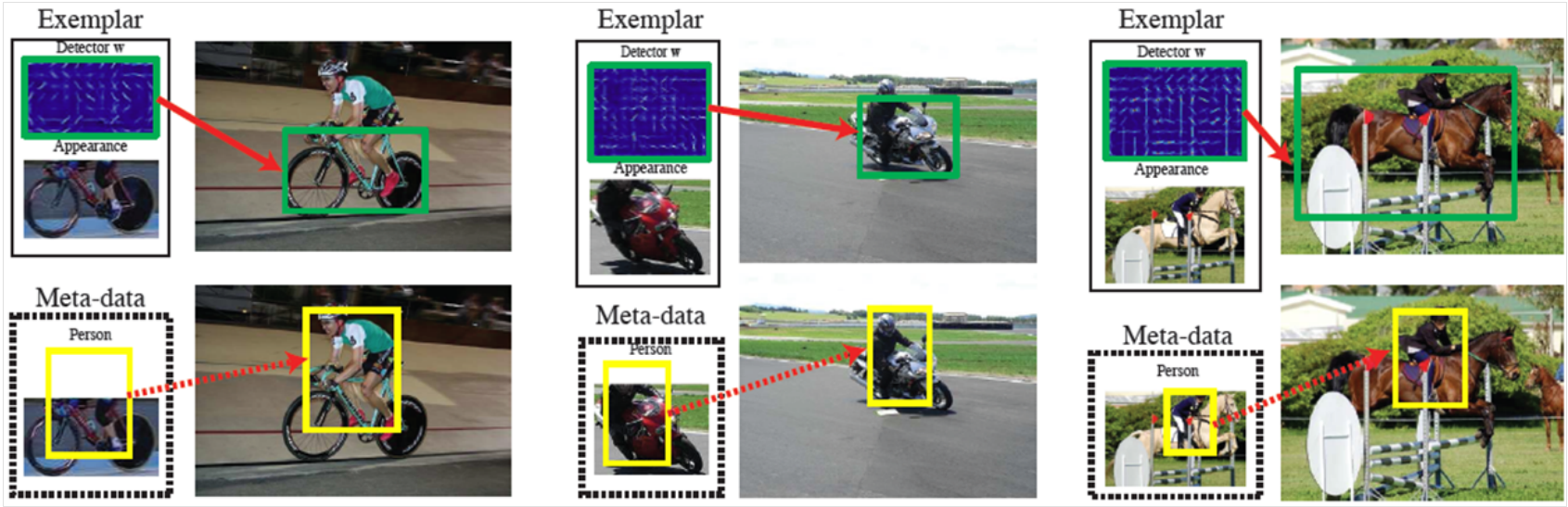
HoG + Exemplar-SVM

HoG + Exemplar-SVM

HoG + Exemplar-SVM.

- An interesting added value of this method is its ability for *object priming*.
- Exemplars often show an interplay of multiple objects, thus any other objects which sufficiently overlap with the exemplar can be viewed as additional meta-data belonging to the exemplar.
- This suggests using detectors of one category to help "prime" objects of another category.
- So, we can think in tasks such as predicting a bounding box for "person" given a detection of category X, where X is either a horse, motorbike, or bicycle.

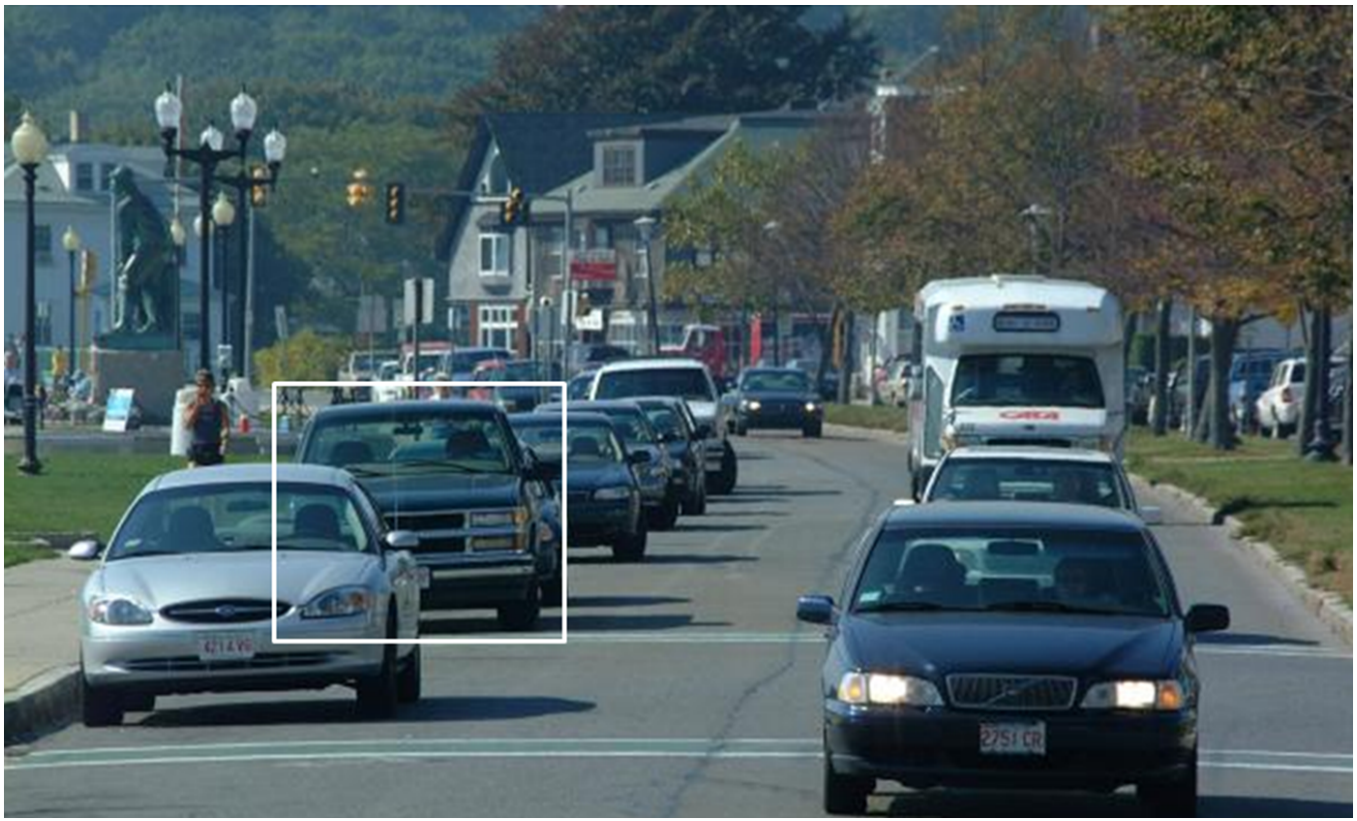
HoG + Exemplar-SVM.



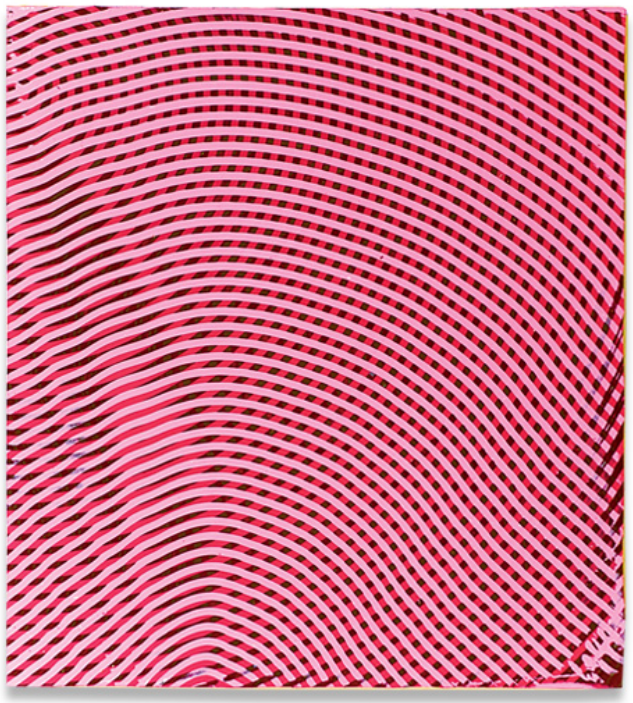
How to deal with object truncation?

Wa cannot train a different model for each possible object truncation.

How to deal with object occlusions?



Structured outputs.



Structured Output Learning.

Problem: Structured Output Learning

Let $d(\mathbf{x}, \mathbf{y}), \mathbf{x} \in \mathbb{R}^{D_1}, \mathbf{y} \in \mathbb{R}^{D_2}$ be an unknown true data distribution.
Let $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$ be a set of N i.i.d. samples from $d(\mathbf{x}, \mathbf{y})$.
Let $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ be a loss function that represents the price we are willing to pay by predicting an estimated value instead of of the true value for an instance of data.
Let $\phi : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^D$ be a feature function thats extracts some feature vector from a given sample and label. It depends on the application.
Let $f(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} \langle \mathbf{w}, \phi(\mathbf{x}, \mathbf{y}) \rangle$ be a decision rule.

- Find the weight vector $\mathbf{w}^*$ that minimizes the expected loss $\mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim d(\mathbf{x},\mathbf{y})}\{\Delta(\mathbf{y}, f(\mathbf{x}))\}$.

Structured Output Learning.

Example $\mathbf{y}, \Delta$: Bounding Box for Object Detection.

Structured Output Learning.

Observations:

1. $d(\mathbf{x}, \mathbf{y})$ is unknown;
2. $\Delta(\mathbf{y}, \arg\max_{\mathbf{y}\in\mathcal{Y}}\langle \mathbf{w}, \phi(\mathbf{x}, \mathbf{y})\rangle)$ is a discontinuous, piecewise constant function.

In order to solve the first problem, we can do two different things:

- To replace $\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim d(\mathbf{x},\mathbf{y})}\{\Delta(\mathbf{y}, f(\mathbf{x}))\}$ by its empirical estimate $\frac{1}{N}\sum_{(\mathbf{x}^n,\mathbf{y}^n)}\{\Delta(\mathbf{y}, f(\mathbf{x}))\}$.
- To add a regularizer, e.g. $\lambda||\mathbf{w}||^2$, to the minimization problem.

Structured Output Learning.

The new problem is to find the weight vector $\mathbf{w}^*$ that minimizes:

$$\min_{\mathbf{w}} \lambda ||\mathbf{w}||^2 + \frac{1}{N} \sum_{n=1}^{N} \Delta(y^n, \arg\max_{\mathbf{y} \in \mathcal{Y}} \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y}) \rangle) \qquad (4)$$

Structured Output Learning.

In order to solve the second problem, we can replace $\Delta(\mathbf{y}, \mathbf{y}')$ with a well behaved (continuous and convex w.r.t. w) upper bound function $\ell(\mathbf{x}, \mathbf{y}, \mathbf{w})$.
The new problem is to find the weight vector $\mathbf{w}^*$ that minimizes:

$$\min_{\mathbf{w}} \lambda ||\mathbf{w}||^2 + \frac{1}{N} \sum_{n=1}^{N} \ell(\mathbf{x}^n, \mathbf{y}^n, \mathbf{w}) \tag{5}$$

Hinge Loss

Hinge Loss

The hinge loss fuction

$$\ell(\mathbf{x}^n, \mathbf{y}^n, \mathbf{w}) := \max_{\mathbf{y} \in \mathcal{Y}} [\Delta(\mathbf{y}^n, \mathbf{y}) + \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y}) \rangle - \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y}^n) \rangle] \quad (6)$$

is a continuous and convex upper bound of Δ.

Observation: If we consider that $\mathbf{y}' = \arg\max_{\mathbf{y}} \langle \mathbf{w}, \phi(\mathbf{x}, \mathbf{y}) \rangle$, it is easy to see that

$$\begin{aligned} \Delta(\mathbf{y}^n, \mathbf{y}') &\leq \Delta(\mathbf{y}^n, \mathbf{y}) + \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y}') \rangle - \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y}^n) \rangle \\ &\leq \max_{\mathbf{y} \in \mathcal{Y}} [\Delta(\mathbf{y}^n, \mathbf{y}) + \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y}) \rangle - \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y}^n) \rangle] \end{aligned}$$

Structured Output Support Vector Machine, v1.0

Structured Output Support Vector Machine

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{N}\sum_{n=1}^{N} \max_{\mathbf{y}\in\mathcal{Y}} [\Delta(\mathbf{y}^n, \mathbf{y}) + \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y})\rangle - \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y}^n)\rangle] \tag{7}$$

This is an unconstrained optimization problem with a convex, non-differentiable objective function.

Structured Output Support Vector Machine, v2.0

Structured Output Support Vector Machine

$$\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{N}\sum_{n=1}^{N}\xi^n$$

s.t. for $n = 1, \ldots, N$.

$$\max_{\mathbf{y}\in\mathcal{Y}}[\Delta(\mathbf{y}^n, \mathbf{y}) + \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y})\rangle - \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y}^n)\rangle] \leq \xi^n$$

This is an optimization problem with N non linear constraints and a convex, differentiable objective function.

Structured Output Support Vector Machine, v3.0

Structured Output Support Vector Machine (N-slack, margin rescaling)

$$\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{N}\sum_{n=1}^{N}\xi^n$$

s.t. for $n = 1, \ldots, N$.

$$\Delta(\mathbf{y}^n, \mathbf{y}) + \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y})\rangle - \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y}^n)\rangle \leq \xi^n$$

for all $\mathbf{y} \in \mathcal{Y}\backslash\{\mathbf{y}^n\}$.

This is an optimization problem with $N|\mathcal{Y}|$ linear constraints and a convex, differentiable objective function.

Structured Output Support Vector Machine, v3.0

Another variant:

Structured Output Support Vector Machine (N-slack, slack rescaling)

$$\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{N}\sum_{n=1}^{N}\xi^n$$

s.t. for $n = 1, \ldots, N$.

$$\langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y}^n)\rangle - \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y}^)\rangle \geq 1 - \frac{\xi^n}{\Delta(\mathbf{y}^n, \mathbf{y})}$$

for all $\mathbf{y} \in \mathcal{Y}\backslash\{\mathbf{y}^n\}$.

This is an optimization problem with $N|\mathcal{Y}|$ linear constraints and a convex, differentiable objective function.

Margin rescaling vs. Slack rescaling

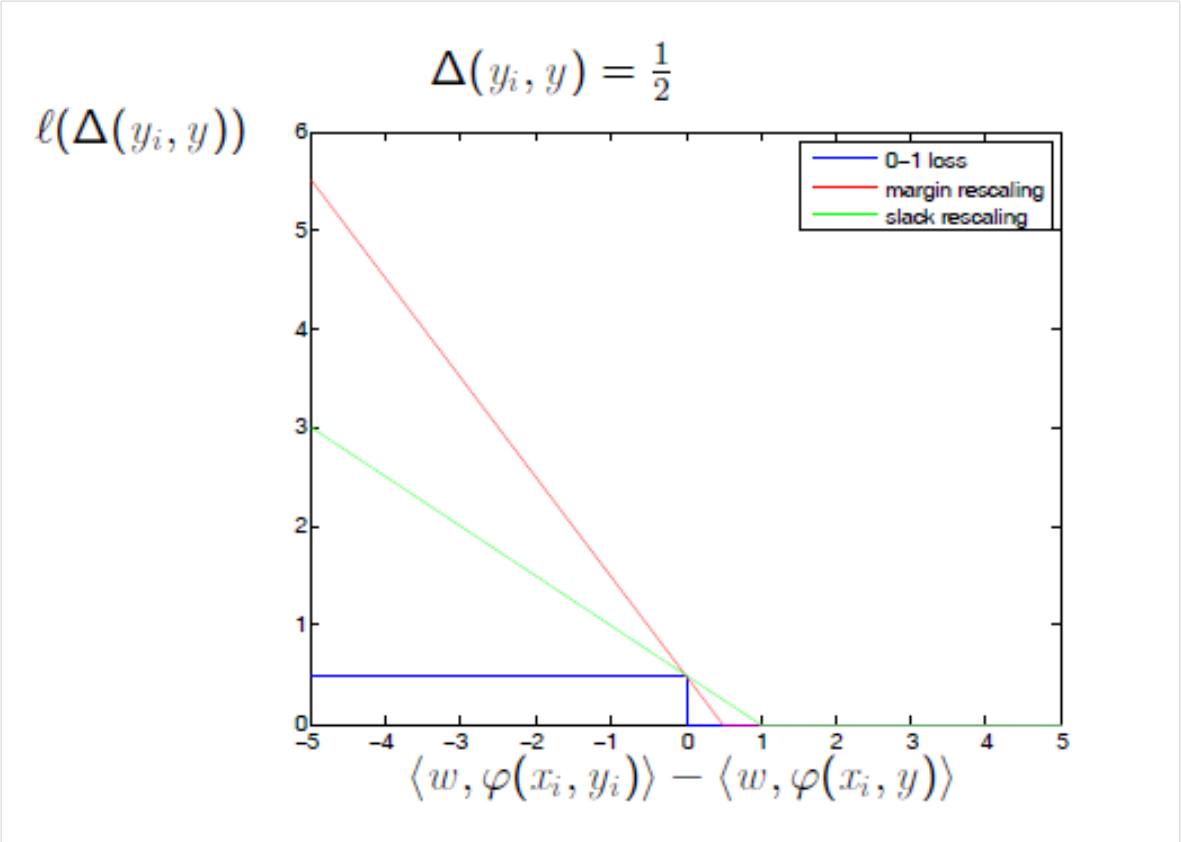


Figure : Margin rescaling vs. Slack rescaling

Training a structural SVM

Algorithm 2: Training a structural SVM (N-slack, margin rescaling)

Data: Input: $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}, C, \epsilon$

Result: $(\mathbf{w}, \xi)$

$\mathcal{W} \leftarrow \emptyset, \xi_i \leftarrow 0$ for all $i = 1, \ldots, N$;

repeat

  foreach $i = 1, \ldots, N$ **do**

    $\mathbf{y}' \leftarrow \arg\max_{\mathbf{y} \in \mathcal{Y}} \{\Delta(\mathbf{y}_i, \mathbf{y}) - \mathbf{w}[\phi(\mathbf{x}_i, \mathbf{y}_i) - \phi(\mathbf{x}_i, \mathbf{y})]\}$;

    if $\Delta(\mathbf{y}_i, \mathbf{y}) - \mathbf{w}((\phi(\mathbf{x}_i, \mathbf{y}_i) - \phi(\mathbf{x}_i, \mathbf{y})) > \xi_i + \epsilon$ **then**

      $\mathcal{W} \leftarrow \mathcal{W} \cup \Delta(\mathbf{y}_i, \mathbf{y}') - \mathbf{w}[\phi(\mathbf{x}_i, \mathbf{y}_i) - \phi(\mathbf{x}_i, \mathbf{y}')]$;

      $(\mathbf{w}, \xi) \leftarrow \arg\min_{\mathbf{w}, \xi \geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{N}\sum_{i=1}^{N} \xi_i$ s.t. $\mathcal{W}$

until $\mathcal{W}$ *has not changed during iteration*;

Training a structural SVM

When applying a structural SVM to a problem we only need to define and supply the following information:

1. The feature function $\phi(\mathbf{x}, \mathbf{y})$.
2. The loss function $\Delta(\mathbf{y}, \mathbf{y}')$
3. The constraint generation function (to identify which is the most incorrect output y that the current model still considers to be compatible with $\mathbf{x}$):
   $\arg\max_{\mathbf{y} \in \mathcal{Y}} \{\Delta(\mathbf{y}_i, \mathbf{y}) - w[\phi(\mathbf{x}_i, \mathbf{y}_i) - \phi(\mathbf{x}_i, \mathbf{y})]\}$

Training a structural SVM

Algorithm 3: Training a structural SVM

Data: Input: $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}, C, \epsilon$

Result: $(\mathbf{w}, \xi)$

$\mathcal{W} \leftarrow \emptyset, \xi_i \leftarrow 0$ for all $i = 1, \ldots, N$;

repeat

  foreach $i = 1, \ldots, N$ **do**

    $\mathbf{y}' \leftarrow \arg\max_{\mathbf{y} \in \mathcal{Y}} \{\Delta(\mathbf{y}_i, \mathbf{y}) - w[\phi(\mathbf{x}_i, \mathbf{y}_i) - \phi(\mathbf{x}_i, \mathbf{y})]\}$;

    if $\Delta(\mathbf{y}_i, \mathbf{y}) - \mathbf{w}((\phi(\mathbf{x}_i, \mathbf{y}_i) - \phi(\mathbf{x}_i, \mathbf{y})) > \xi_i + \epsilon$ **then**

      $\mathcal{W} \leftarrow \mathcal{W} \cup \mathbf{y}'$;

      $(\mathbf{w}, \xi) \leftarrow \arg\min_{\mathbf{w}, \xi \geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{N}\sum_{i=1}^{N} \xi_i$ s.t. $\mathcal{W}$

until $\mathcal{W}$ *has not changed during iteration*;

Training a structural SVM

Each update of **w** needs 1 *argmax*-prediction per example.
An alternative is to use the one-slack formulation of SSVM (that is equivalent to N-slack SSVM formulation by $\xi = \frac{1}{N}\sum_N \xi^n$). In this case we add one strong constraint per iteration intead of n weak ones.

Structured Output Support Vector Machine, v4.0

Structured Output Support Vector Machine (one-slack)

$$\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + C\xi$$

s.t.

$$\sum_{n=1}^{N} [\Delta(\mathbf{y}^n, \hat{\mathbf{y}}^n) + \langle \mathbf{w}, \phi(\mathbf{x}^n, \hat{\mathbf{y}}^n) \rangle - \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y}^n) \rangle] \leq N\xi$$

for all $(\hat{\mathbf{y}}^1, \ldots, \hat{\mathbf{y}}^N) \in \mathcal{Y} \times \cdots \times \mathcal{Y}$.

This is an optimization problem with $|\mathcal{Y}|^N$ linear constraints and a convex, differentiable objective function.

Kernelized SSVM

We can solve a SSVM like a non-linear SVM by computing the Lagrangian dual. In this case *min* becomes *max*, the original variables $\mathbf{w}, \xi$ disappear, and new variables α appear (one per constraint of the original problem).

Dual Structured Output Support Vector Machine

$$\max_{\alpha} \sum_{n,\mathbf{y}} \alpha_{n,\mathbf{y}} \Delta(\mathbf{y}_n, \mathbf{y})$$

$$-\frac{1}{2} \sum_{n,\mathbf{y},n',\mathbf{y}'} \alpha_{n,\mathbf{y}} \alpha_{n',\mathbf{y}'} \langle \delta\phi(\mathbf{x}_n, \mathbf{y}_n, \mathbf{y}), \delta\phi(\mathbf{x}'_n, \mathbf{y}'_n, \mathbf{y}') \rangle$$

s.t. $\sum_{\mathbf{y} \in \mathcal{Y}} v \leq \frac{C}{N}$ for all $n = 1, \ldots, N$.

Kernelized SSVM

- This is a problem with N linear constraints and a convex, differentiable objective with $N|\mathcal{Y}|$ variables.
- We can redefine the optimization in terms of $k((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')) = \langle\phi(\mathbf{x},\mathbf{y}),\phi(\mathbf{x}',\mathbf{y}')\rangle$:

$$\begin{aligned}
&\langle\delta\phi(\mathbf{x}_n,\mathbf{y}_n,\mathbf{y}),\delta\phi(\mathbf{x}_{n'},\mathbf{y}_{n'},\mathbf{y}')\rangle \\
&= \langle\phi(\mathbf{x}_n,\mathbf{y}_n)-\phi(\mathbf{x}_n,\mathbf{y}),\phi(\mathbf{x}_{n'},\mathbf{y}_{n'})-\phi(\mathbf{x}_{n'},\mathbf{y}')\rangle \\
&= \langle\phi(\mathbf{x}_n,\mathbf{y}_n),\phi(\mathbf{x}_{n'},\mathbf{y}_{n'}\rangle - \langle\phi(\mathbf{x}_n,\mathbf{y}_n),\phi(\mathbf{x}_{n'},\mathbf{y}'\rangle \\
&\quad -\langle\phi(\mathbf{x}_n,\mathbf{y}),\phi(\mathbf{x}_{n'},\mathbf{y}_{n'}\rangle + \langle\phi(\mathbf{x}_n,\mathbf{y}),\phi(\mathbf{x}_{n'},\mathbf{y}'\rangle \\
&= k((\mathbf{x}_n,\mathbf{y}_n),(\mathbf{x}_{n'},\mathbf{y}_{n'})) - k((\mathbf{x}_n,\mathbf{y}_n),(\mathbf{x}_{n'},\mathbf{y}')) \\
&\quad -k((\mathbf{x}_n,\mathbf{y}),(\mathbf{x}_{n'},\mathbf{y}_{n'})) - k((\mathbf{x}_n,\mathbf{y}),(\mathbf{x}_{n'},\mathbf{y}')) \\
&= K_{i,i',\mathbf{y},\mathbf{y}'} \qquad (8)
\end{aligned}$$

Kernelized SSVM

Kernelized SSVM

$$\max_{\alpha} \sum_{i,\mathbf{y}} \alpha_{i,\mathbf{y}} \Delta(\mathbf{y}_n, \mathbf{y}) - \frac{1}{2} \sum_{i,\mathbf{y},i',\mathbf{y}'} \alpha_{i,\mathbf{y}} \alpha_{i',\mathbf{y}'} K_{i,i',\mathbf{y},\mathbf{y}'}$$

s.t.

$$\sum_{\mathbf{y} \in \mathcal{Y}} \alpha_{i,\mathbf{y}} \leq \frac{C}{N}$$

for all $i = 1, \ldots, N$.

The kernelized prediction function is:

$$f(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} \sum_{i,\mathbf{y}} \alpha_{i,\mathbf{y}} k((\mathbf{x}_i, \mathbf{y}_i), (\mathbf{x}, \mathbf{y}))$$

.

Example: Multiclass Classification.

Let $\mathcal{Y} = 1, 2, \ldots, K$ be a finite set of labels.

Let $\Delta(y, y')$ be $\begin{cases} 1, & \text{for } y \neq y' \\ 0, & \text{otherwise} \end{cases}$

Let $\phi(\mathbf{x}, y) = (\mathbb{1}_{\{y=1\}}\phi'(\mathbf{x}), \ldots, \mathbb{1}_{\{y=K\}}\phi'(\mathbf{x}))$ be the feature vector.

Let $f(\mathbf{x}) = \arg\max_y \langle w, \phi(\mathbf{x}, y)\rangle$ the decision rule.

Example: Multiclass Classification.

Multiclass classification

The problem is to solve

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{N}\sum_{n=1}^{N}\xi^n$$

s.t.

$$\langle \mathbf{w}, \phi(\mathbf{x}^n, y^n)\rangle - \langle \mathbf{w}, \phi(\mathbf{x}^n, y)\rangle \geq 1 - \xi^n$$

for $n = 1, \ldots, N$ and for all $y \in \mathcal{Y}\backslash\{y^n\}$

Exercise

We want to learn a $\mathbf{w} \in \mathbb{R}^2$ to separate a set of $2D$ points with binary labels: $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}, x \in \mathbb{R}^2, y \in \{-1, +1\}$.
The structural formulation is

$$f(\mathbf{x}) = \underset{y \in \{-1,+1\}}{\arg\max} \langle \mathbf{w}, \phi(\mathbf{x}, y) \rangle$$

Problem: SSVM formulation of a binary loss SVM

Show that if we define $\phi(\mathbf{x}, y) = \frac{\mathbf{x}y}{2}$ and we use a 01-loss function ($\Delta(y, y') = (1 - yy')/2$) we obtain a SO-SVM formulation that is equivalent to the classical linear SVM.

Exercise

Given a binary data set where the number of samples of one class is much larger than the number of samples of the other class, we can deal with the data **imbalance problem** by paying a higher price for misclassification of samples belonging to the most populated class.

Problem: Data Imabalance

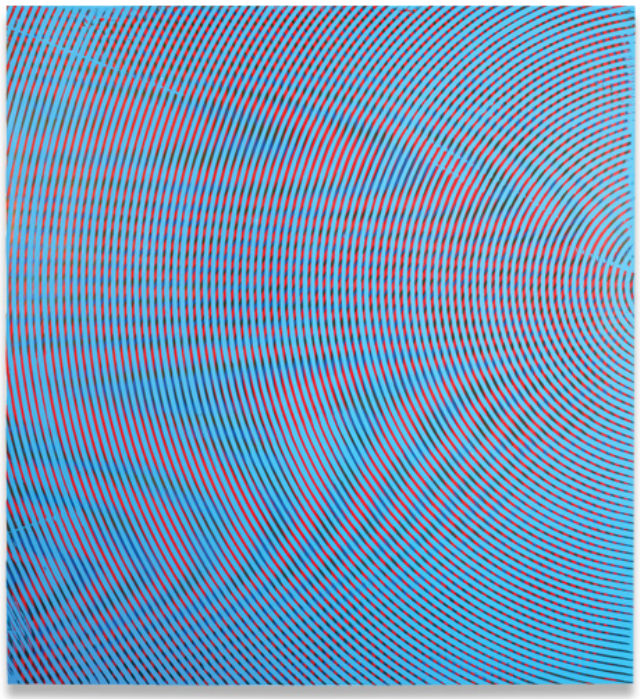
Given a data set composed of n_+ positive samples and n_- negative samples, with $n_+ < n_-$, we can use the following loss function:

$$\Delta(y_i, y) = \begin{cases} 0 & \text{if } y_i = y \\ \frac{1}{n_+} & \text{if } y_i = 1 \wedge y = -1 \\ \frac{1}{n_-} & \text{if } y_i = -1 \wedge y = 1 \end{cases}$$

Applications

- Two classes SSVM with Loss = PASCAL criterion.
- SSVM with Loss = PASCAL criterion and $\mathbf{y} = (+1, x_1, y_1, x_2, y_2)$.
- SSVM with Loss = Tree
- Kernelized SSVM (intersection kernel).
- SVM for early detection of sequence events.

SSVM for Object Recognition.



SSVM for Object Recognition

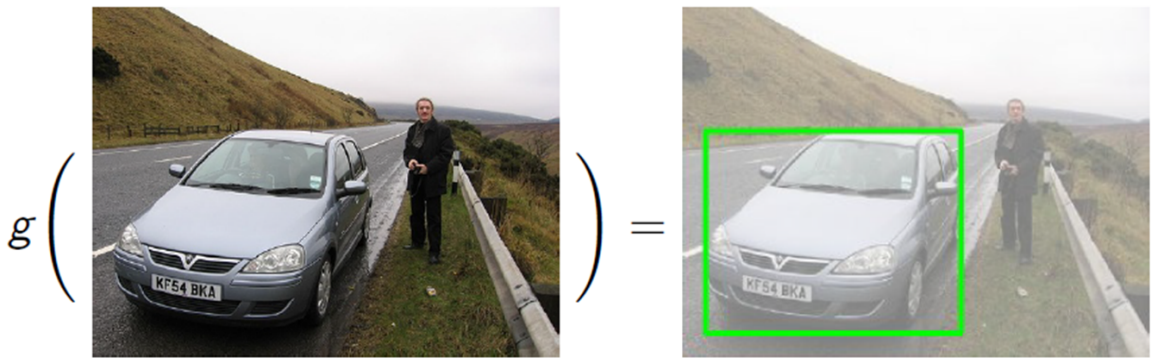
The binary training + sliding window approach to object detection (Where is the object?) is inconsistent because:

- We learn the wrong task. During training, only the sign of f matters, but during testing f is used as a real valued quality measure.
- The training samples do not reflect the test situation. During training, samples show either *complete object* or *no object*, but during testing many subregions show object parts.

SSVM for Object Recognition

We can re-formulate the problem as:

- Learn a true *localization function*:
  $g : \{\text{all images} \rightarrow \text{all boxes}\}$ that predicts object location from images.
- Train in a consistent end-to-end way.
- Training distribution reflects test distribution.



SSVM for Object Recognition

- We are given a set of training pairs $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N) \in \mathcal{X} \times \mathcal{Y}$, where $\mathbf{x}_i$ are images and $\mathbf{y}_i$ are bounding boxes.
- The objective is to learn a mapping $g = \mathcal{X} \rightarrow \mathcal{Y}$ that generalizes from the given examples $g(\mathbf{x}_i) \sim \mathbf{y}_i$, for $i = 1, \ldots, N$.
- To avoid overfitting, we would prefer smooth maps.

SSVM for Object Recognition

This problem can be solved if formulated as a Structured SVM optimization problem.

Structured Output Support Vector Machine

$$\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{N}\sum_{n=1}^{N}\xi^n$$

s.t. for $n = 1, \ldots, N$.

$$\Delta(\mathbf{y}^n, \mathbf{y}) + \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y})\rangle - \langle \mathbf{w}, \phi(\mathbf{x}^n, \mathbf{y}^n)\rangle \leq \xi^n$$

for all $\mathbf{y} \in \mathcal{Y}\backslash\{\mathbf{y}^n\}$.

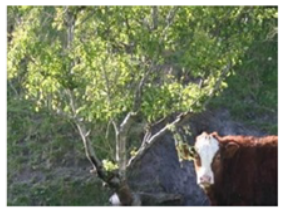
SSVM for Object Recognition

- $\Delta(\mathbf{y}_i, \mathbf{y}) = \begin{cases} \text{small, if } \mathbf{y}_i \approx \mathbf{y} \\ \text{large, if } \mathbf{y}_i \not\approx \mathbf{y} \end{cases}$, $\Rightarrow g(\mathbf{x}_i) \approx \mathbf{y}_i$ for most $(\mathbf{x}_i, \mathbf{y}_i)$.
- We can use a kernelized version of SSVM and use a joint kernel function defined as $k_{joint}((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) = k_{joint}((\mathbf{x}_\mathbf{y}, \mathbf{x}'_{\mathbf{y}'}))$, where $\mathbf{x}_\mathbf{y}$ is an image restricted to a box region.

SSVM for Object Recognition

k_{joint}( , ) = k( , ) is large.

k_{joint}( , ) = k( , ) is small.

k_{joint}( , ) = k( , )

could also be large.

SSVM for Object Recognition

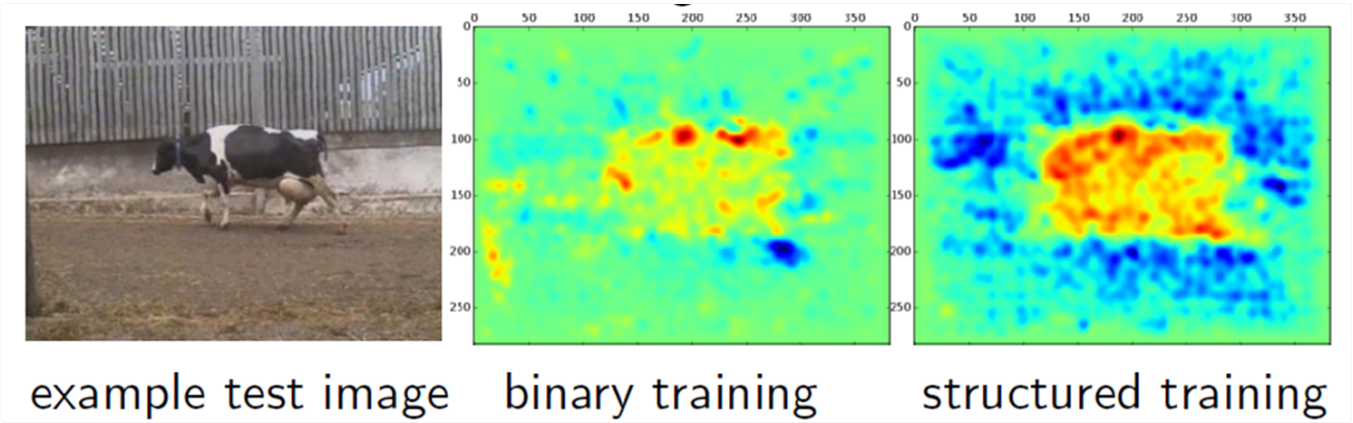
What is a godd loss function $\Delta(\mathbf{y}_i, \mathbf{y})$?

- $\Delta(\mathbf{y}_i, \mathbf{y})$ plays the role that the margin has in SVM.
- $\Delta(\mathbf{y}_i, \mathbf{y})$ measures how far a prediction $\mathbf{y}$ is from a target $\mathbf{y}_i$.
- We can use *area overlap*:

$$\begin{cases} \Delta(\mathbf{y}_i, \mathbf{y}) = 1 - [\text{area overlap between } \mathbf{y}_i \text{ and } \mathbf{y}] = 1 - \frac{area(\mathbf{y} \cap \mathbf{y}_i)}{area(\mathbf{y} \cup \mathbf{y}_i)} \\ \Delta(\mathbf{y}_i, \mathbf{y}) = 0 \text{ iff } \mathbf{y}_i = \mathbf{y} \\ \Delta(\mathbf{y}_i, \mathbf{y}) = 1 \text{ iff } \mathbf{y}_i \cap \mathbf{y} = \emptyset \end{cases}$$

SSVM for Object Recognition

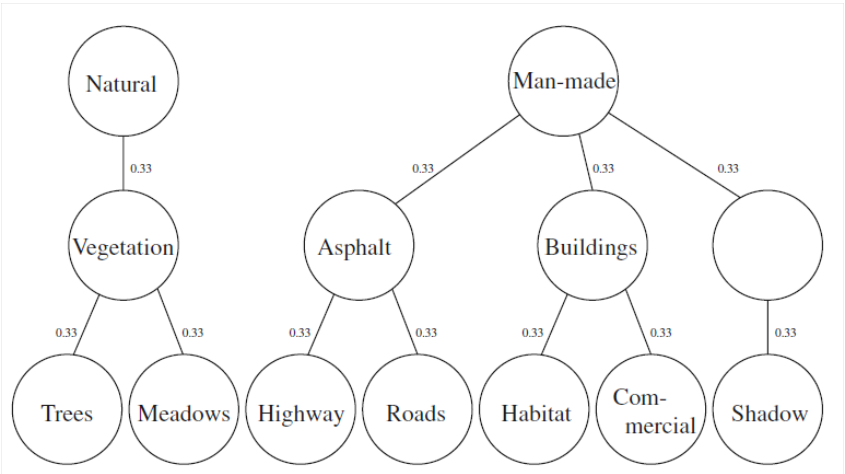
With BoW histograms and a linear kernel, the learned distribution of local *weights* is:



With binary training, weights are concentrated on few distinctive regions. With structured learning, positive weights are inside and negative weights are outside.

SSVM for hierarchical class recognition

Let's suppose we have a multiclass segmentation remote sensing problem with seven classes, even though 13 classes can be created from the structure by grouping physically-similar classes into superclasses (upper levels) [4].



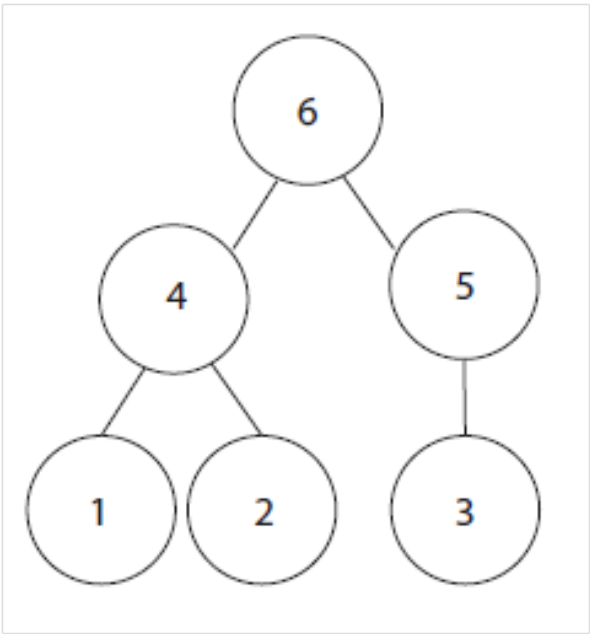
[4] D.Tuia, J.Muoz-Mar, M.Kanevski, G. Camps-Valls. Structured Output SVM for Remote Sensing Image Classification.J Sign Process Syst., DOI 10.1007/s11265-010-0483-8

SSVM for hierarchical class recognition

How can we encode the structure of the tree in a joint input-output mapping ϕ? (Reminder: $\phi(\mathbf{x}, y)$ must be high for the correct class and low for the rest).

- The class `meadows` can be encoded by a vector $c^T(\texttt{meadows}) = (0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0)$, where the first 1 indicates that the sample belongs to the class `meadows`, the second one indicates that the sample belongs to the superclass `vegetation`, and the third one indicates that the sample belongs to the superclass `natural`.
- Then, we can define the a joint input-output feature map: $\phi(\mathbf{x}, y) = (c_1(y) \cdot \mathbf{x}, \ldots, c_K(y) \cdot \mathbf{x})$, where K is the number of nodes in the tree.

SSVM for hierarchical class recognition



$$C(\mathbf{y})^T = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

$$\phi(\mathbf{x}, \mathbf{y})^T = \begin{pmatrix} \mathbf{x} & 0 & 0 & \mathbf{x} & 0 & \mathbf{x} \\ 0 & \mathbf{x} & 0 & \mathbf{x} & 0 & \mathbf{x} \\ 0 & 0 & \mathbf{x} & 0 & \mathbf{x} & \mathbf{x} \end{pmatrix}$$

Observation: We consider a different weight $\mathbf{w}$ for each node of the tree.

SSVM for hierarchical class recognition

- The linear dot product between the two first classes will result in $2(\mathbf{x}\mathbf{x}')$, while between the classes 1 and 3 (and 2 and 3) it is of $(\mathbf{x}\mathbf{x}')$ only.
- This way, the similarity between two outputs sharing common superclasses will be higher than between outputs that are distant in the tree.

SSVM for hierarchical class recognition

- To define the loss function, we can modify the classical 0/1 loss by exploiting the tree-based output structure.
- $\Delta(y, y') = (l-1)/L$ if $y^l = y'^l$, and 1 otherwise, where $l = \{1, \ldots, L\}$ are tree levels.
- Using this loss, errors predicting far away classes are penalized more than close errors.
- This loss function assumes equal distance between the classes and their superclasses: this can be refined by constructing ad hoc class distances from the labeled data or by learning inter-class distances.

Latent Structural SVM

- In many structured prediction tasks, there is useful modeling information that is not available as part of the training data $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$.
- This missing information h, even if not observable, is crucial for expressing high-fidelity models for these tasks. It is important to include these information in the model as *latent variables*.
- To consider latent variables in Structural SVM we extend our joint feature vector $\phi(\mathbf{x}, \mathbf{y})$ to include an extra argument $\mathbf{h}$ to $\phi(\mathbf{x}, \mathbf{y}, \mathbf{h})$.

We want to learn a prediction rule of the form

$$f(\mathbf{x}) = \underset{(\mathbf{y},\mathbf{h}) \in \mathcal{Y} \times \mathcal{H}}{\arg\max} \; w \cdot \phi(\mathbf{x}, \mathbf{y}, \mathbf{h})$$

Latent Structural SVM

This problem gives rise to the following optimization problem

Latent Structural SVM

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2$$

$$+C\sum_{n=1}^{N}\left(\max_{(\mathbf{y}',\mathbf{h}')\in\mathcal{Y}\times\mathcal{H}}\left[\mathbf{w}\cdot\phi(\mathbf{x}_i,\mathbf{y}',\mathbf{h}')+\Delta(\mathbf{y}_i,\mathbf{y}',\mathbf{h}')\right]\right)$$

$$-C\sum_{n=1}^{N}\left(\max_{\mathbf{h}\in\mathcal{H}}\mathbf{w}\cdot\phi(\mathbf{x}_i,\mathbf{y}_i,\mathbf{h})\right) \qquad (9)$$

Training a Latent Structural SVM

Algorithm 4: Training a latent structural SVM

Data: Input: $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}, w_0, \epsilon$

Result: $\mathbf{w}$

$t \leftarrow 0$;

repeat

  Update $\mathbf{h}'_i = \arg\max_{\mathbf{h}_i \in \mathcal{H}} w_t^T \phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i)$;

  Update w_{t+1} by fixing the hidden variables for $\mathbf{y}_i$ to $\mathbf{h}'_i$ and solving $w_{t+1} = \min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{N} \sum_{n=1}^{N} \max_{(\mathbf{y}', \mathbf{h}') \in \mathcal{Y} \times \mathcal{H}} [0, \Delta(\mathbf{y}_i, \mathbf{y}', \mathbf{h}') + w^t(\phi(\mathbf{x}_i, \mathbf{y}'_i, \mathbf{h}'_i) + \phi(\mathbf{x}_i,$

  $t \leftarrow 0$;

until *objective function cannot be decreased below* ϵ;

Applications

- Object Detection.
- pb-SVM
- Semisupervised learning

Part-based Object Recognition

- The L-SVM algorithm.
- Hard negatives.
- Dynamic programming on trees.

Conclusions

From the classification point of view, these methods are limited by the limitations of binary SVMs:

- SVM is not suited for multiclass classification. The most common approach for doing this is to reduce the muticlass problem into a multiple binary classification problem and then to use decision rules such as the one-versus-all or ECOC strategies.
- The loss function is limited to the 0-1 case.

PASCAL VOC Challenge

Problem

Localize and name (detect) 20 basic-level object categories (Airplane, bicycle, bus, cat, car, dog, person, sheep, sofa, monitor, etc.).

1. 11k training images with 500 to 8000 instances / category.
2. Evaluation: bounding-box overlap; average precision (AP).

PASCAL VOC Challenge

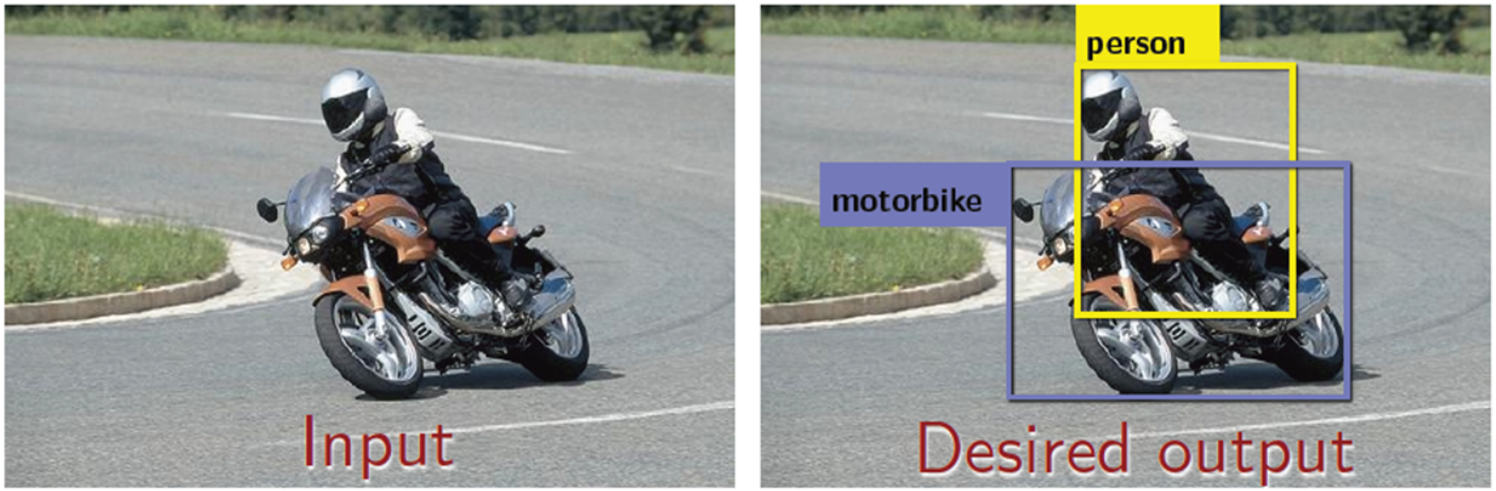
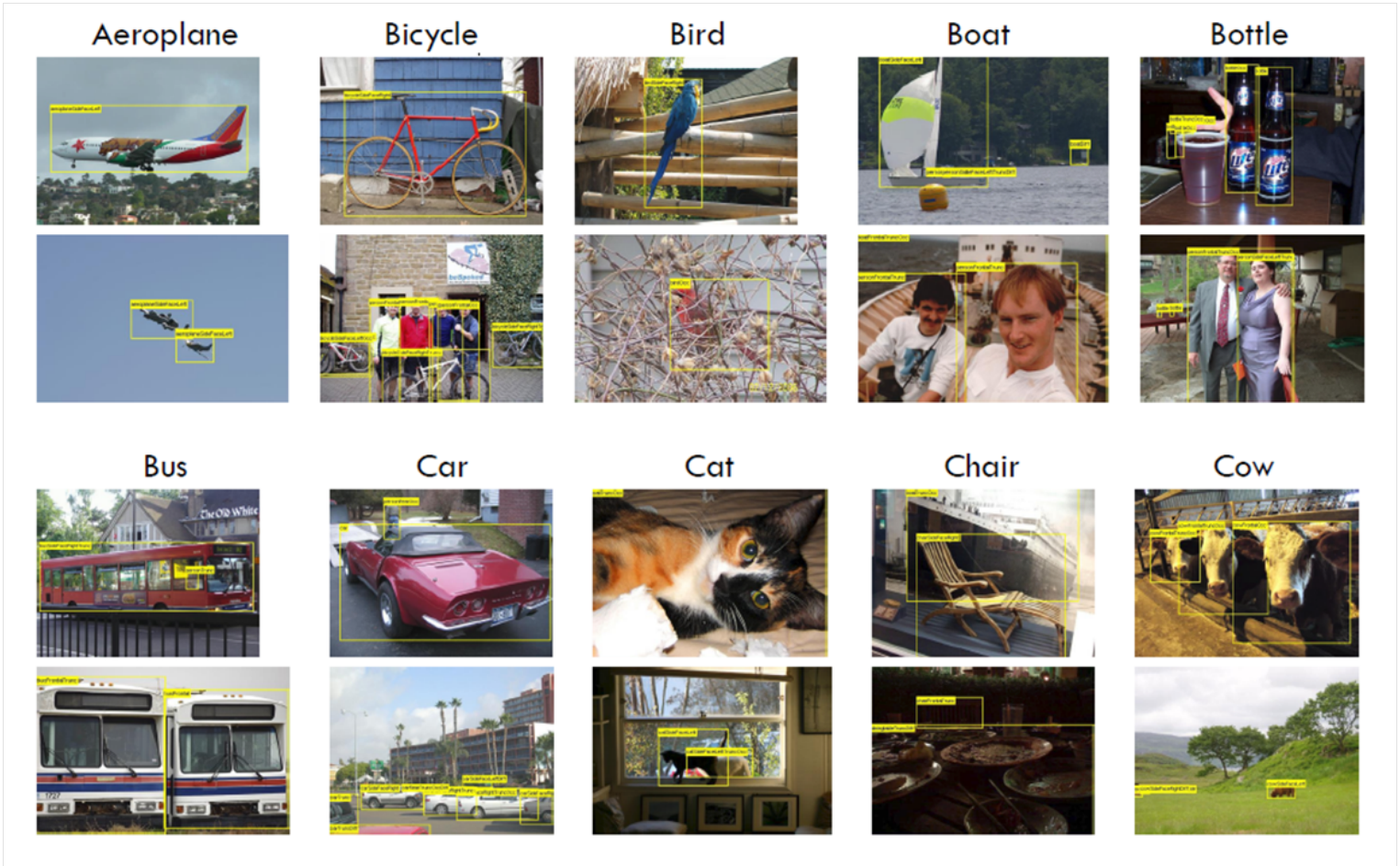


Figure : The PASCAL Challenge for Object Detection.

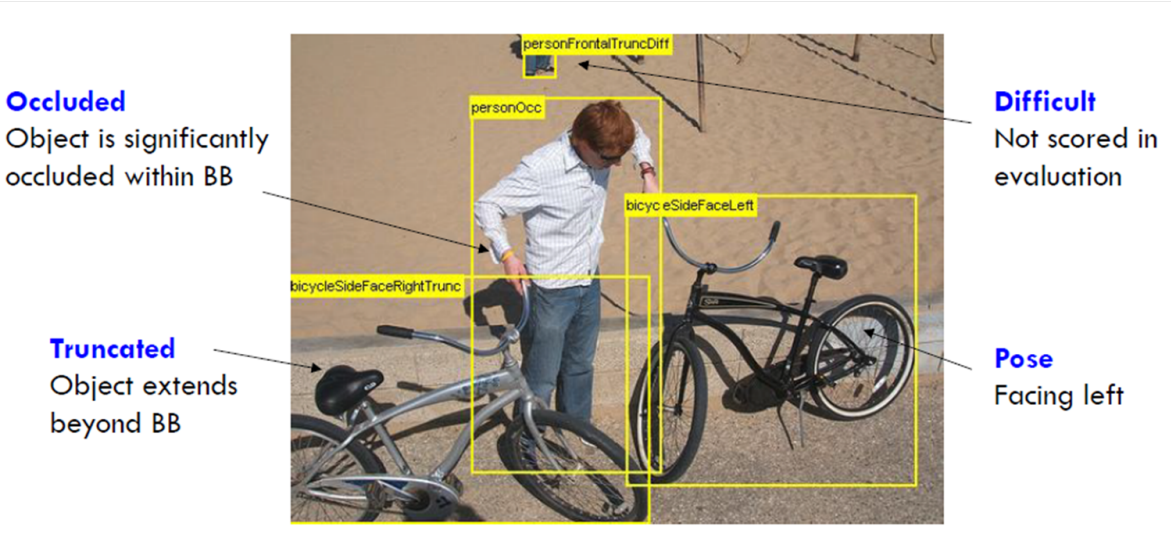
PASCAL VOC Challenge



PASCAL VOC Challenge

PASCAL VOC Challenge

Complete annotation of objects from 20 categories



Precision-Recall

For classification tasks, the terms true positives (tp), true negatives (tn), false positives (fp), and false negatives (fp) compare the results of the classifier under test with trusted external judgments. **Precision** and **Recall** are then defined as:

$$P = \frac{tp}{tp + fp}$$

$$R = \frac{tp}{tp + fn}$$

Precision-Recall

In this figure the true objects are to the left of the straight line while the detected objects are within the oval. The red regions represent errors. On the left these are the true objects not detected (false negatives), while on the right they are the detected windows that are not objects (false positives). Precision and recall are the quotient of the left green region by respectively the oval (horizontal arrow) and the left region (diagonal arrow).

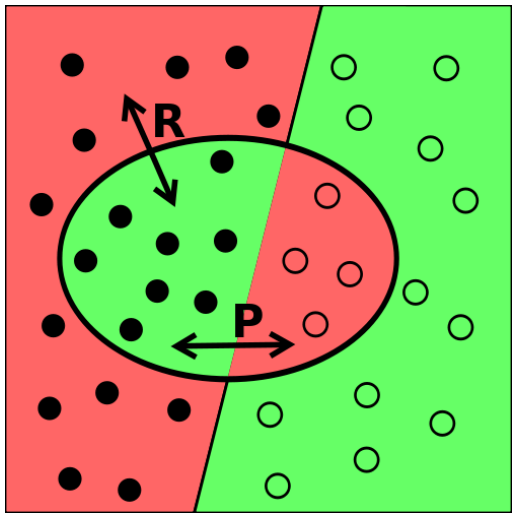


Figure : Precision-Recall

Average Precision

- By relaxing the threshold value of the decision function we can increase the number of windows that our system considers as an object. We could also ask our system to be more strict, and return fewer windows.
- We can move that threshold up and down to get a different set of detections. At each position of the threshold, we would get a different precision and recall value.
- A good way to characterize the performance of a classifier is to look at how precision and recall change as you change the threshold: In the case of a good classifier, its precision will stay high as recall increases. A poor classifier will have to take a large hit in precision to get higher recall.

Average Precision

- Rather than comparing curves, its sometimes useful to have a single number that characterizes the performance of a classifier. A common metric is the **average precision.**
- Average precision is precision averaged across all values of recall between 0 and 1; Thats equal to taking the area under the curve.

PASCAL VOC Challenge

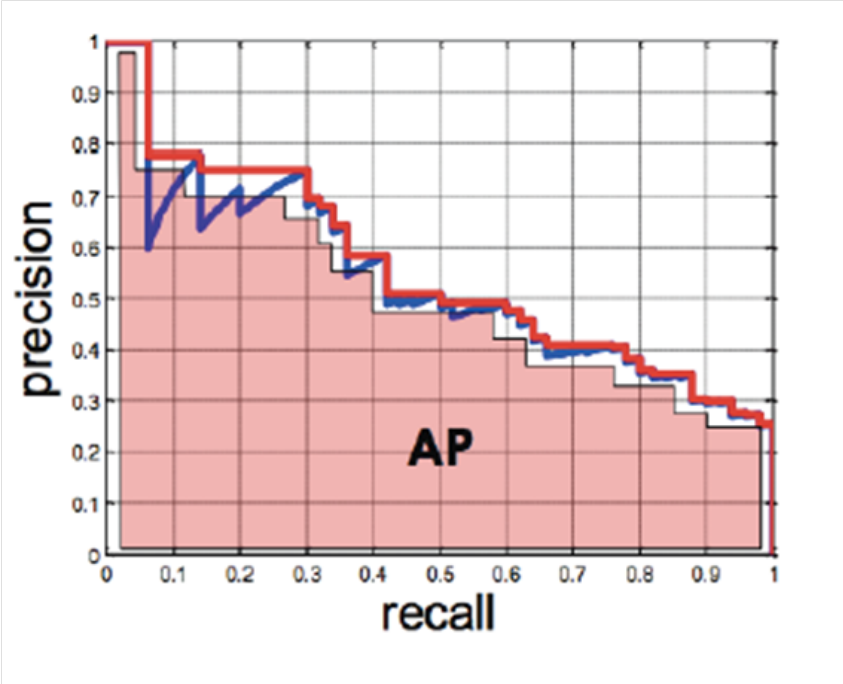


Figure : Average Precision Metric.

PASCAL VOC Challenge

In the VOC challenge detections are assigned to ground truth objects and judged to be true/false positives by measuring bounding box overlap. To be considered a correct detection, the overlap ratio between the predicted bounding box and ground truth bounding box must exceed 0.5:

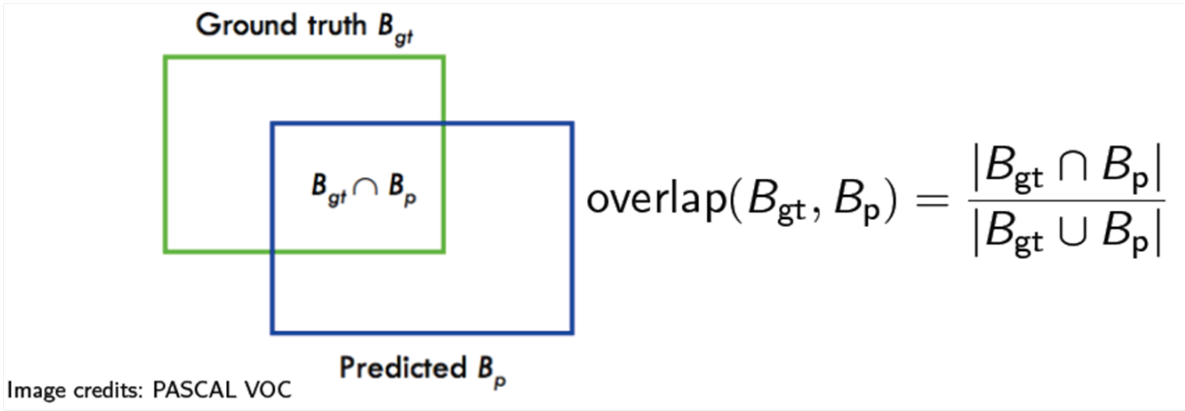


Figure : Bounding Box Intersection Metric.

PASCAL VOC Challenge

State of the art (2012):

aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	sofa	train	tvmonitor
45.4	49.8	15.7	16.0	26.3	54.6	44.8	35.1	16.8	31.3	23.6	26.0	45.6	49.6	42.2	14.5	30.5	28.5	45.7	40.0
51.4	53.7	18.3	15.6	31.6	56.5	47.1	38.6	19.5	32.0	22.1	25.0	50.3	51.9	44.9	11.9	37.7	30.6	50.8	39.3
65.1	46.8	25.0	24.6	16.0	51.0	44.9	51.5	13.0	26.6	31.0	40.2	39.7	51.5	32.8	12.6	35.7	33.5	48.0	44.8
47.5	51.7	14.2	12.6	27.3	51.8	44.2	25.3	17.8	30.2	18.1	16.9	46.9	50.9	43.0	9.5	31.2	23.6	44.3	22.1
50.1	47.0	7.9	3.8	24.8	47.2	42.8	31.2	17.5	24.2	10.0	21.3	43.5	46.4	37.5	7.9	26.4	21.5	43.1	36.7
59.6	54.5	21.9	21.6	32.1	52.5	49.3	40.8	19.1	35.2	28.9	37.2	50.9	49.9	46.1	15.6	39.3	35.6	48.9	42.8
47.2	50.2	18.3	21.4	25.2	53.3	46.3	46.3	17.5	27.8	30.3	35	41.6	52.1	43.2	18	35.2	31.1	45.4	44.4
61.8	52	24.6	24.8	20.2	57.1	44.5	53.6	17.4	33	38.3	42.8	48.8	59.4	35.7	22.8	40.3	39.5	51.1	49.5

PASCAL VOC Challenges 2007-2011

1. 2006: The winner is a HoG-based method (11 % mAP).
2. 2007: The winner is Deformable Part Models & Latent SVM (21 % mAP).
3. 2008, 2009: The winner is Deformable Part Models & Latent SVM (29 % mAP).
4. 2011: The winner method = Deformable Part Models & Latent SVM + LBP image features + richer spatial model (GMM) + more context rescoring (35 % mAP).

What is a part?

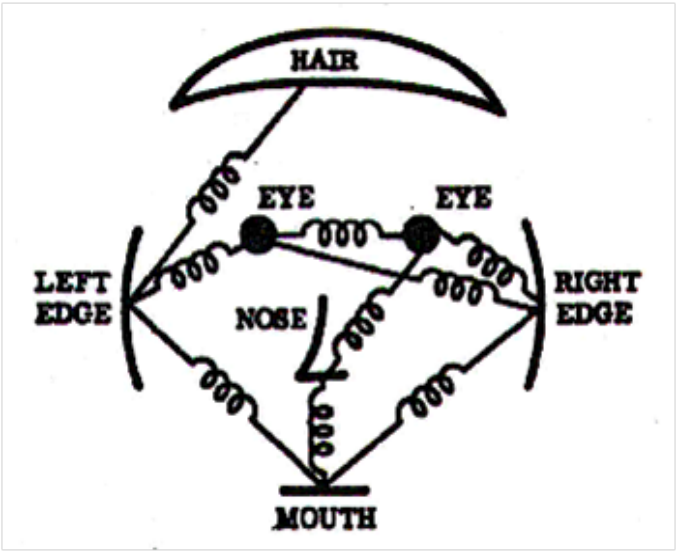


Figure : Figure from [Fischler and Elschlager 73].

Part-based object detection

- Enrich the Dalal & Triggs model using a star-structured part-based model defined by a root filter (analogous to the Dalal-Triggs filter) plus a set of parts filters and associated deformation models.
- Define the score of a star models at a particular position and scale within an image as the score of the root filter at the given location plus the sum over parts of the maximum, over placements of that part, of the part filter score on its location minus a deformation cost measuring the deviation of the part from its ideal location relative to the root.

Part-based object detection

- In the most succesful methods models, the part filters capture features at twice the spatial resolution relative to the features captured by the root filter.
- Part locations are not known at test time, so they are treated as hidden/latent variables.
- To train models using partially labeled data we use a latent variable formulation of SVM called latent SVM (LSVM).

Part-based object detection

- In a latent SVM each example x is scored by a function of the following form:

$$f_{\mathbf{w}} = \max_{\mathbf{z}} \mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{z})$$

- $\mathbf{w}$ is a vector of model parameters (= the concatenation of the root filter, the part filters, and deformation cost weights), $\mathbf{z}$ are latent values (= a specication of the object configuration) and $\Phi(\mathbf{x}, \mathbf{z})$ is a feature vector (= a concatenation of subwindows from a feature pyramid and part deformation features).

Model Representation

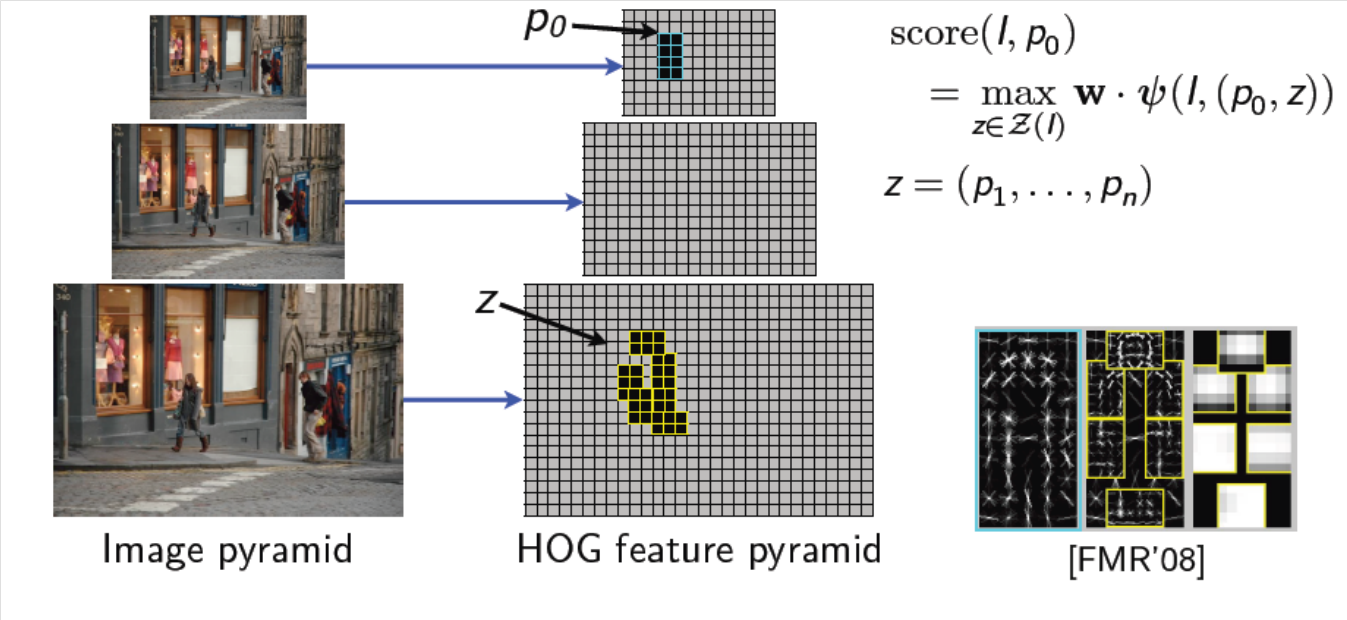


Figure : Object representation. HoG Root + HoG Parts in a deformable configuration **z**.

Object Detection

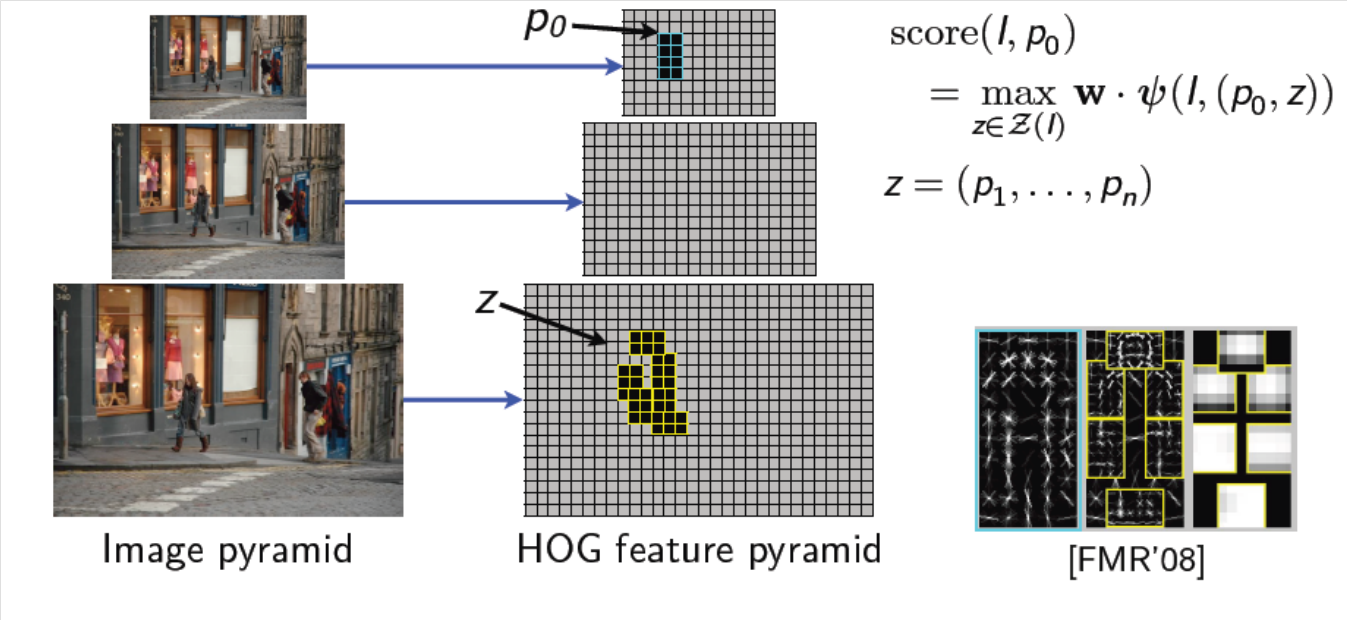


Figure : Scanning window detection with *max* over **z** at each p_0 + **w** learned by latent SVM.

Object Detection

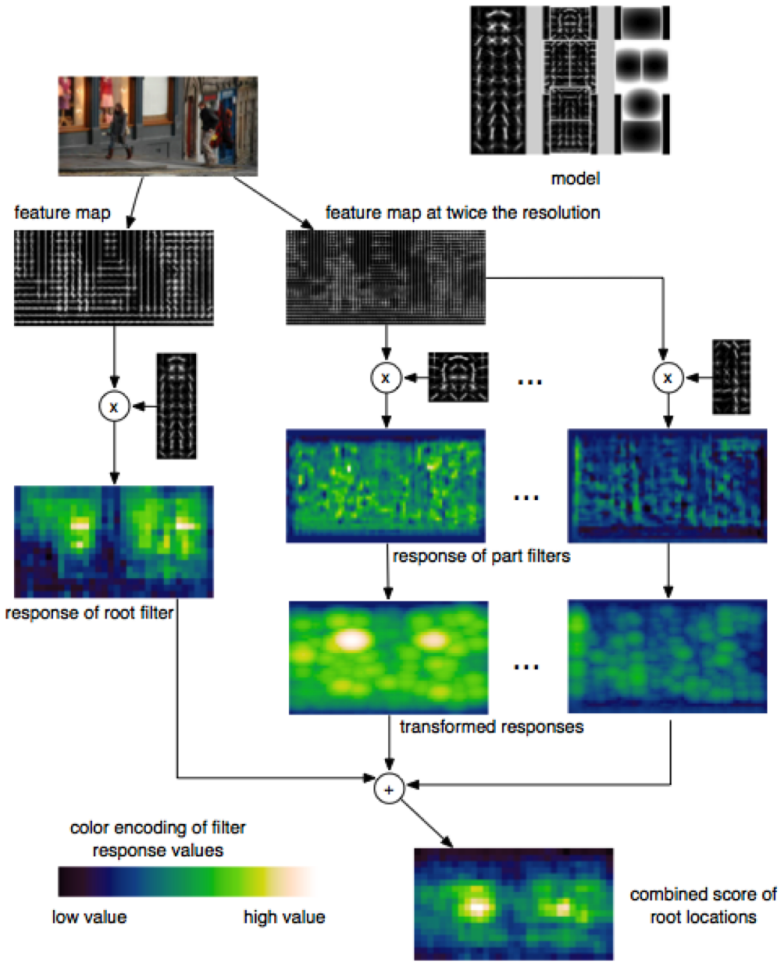
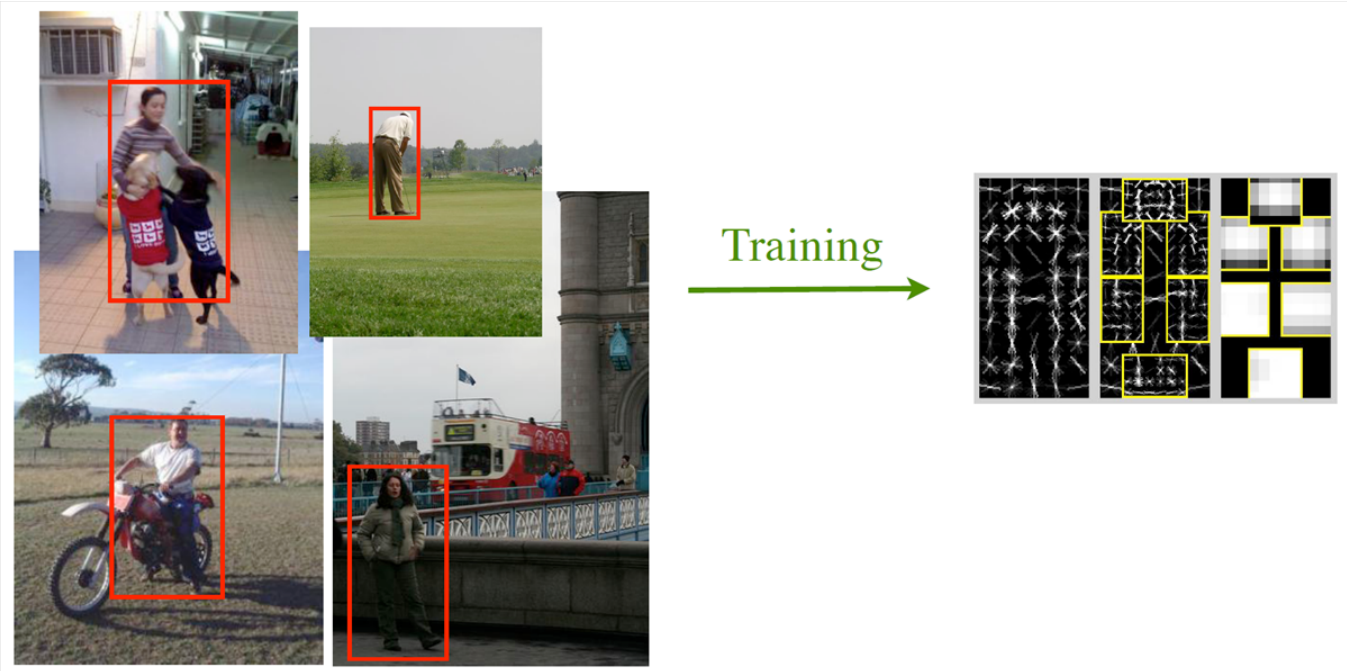


Figure : Object detection

How to train a part-based object detector?

Training data consists of images with labeled bounding boxes. We must define/learn the model structure, filters and deformation costs.



How to train a part-based object detector?

The first step is to build a mixture model for each object category:

1. Cluster training examples by bounding-box aspect ratio.
2. Initialize root filters for each component (cluster) independently.
3. Merge components into mixture model and train with **latent SVM**.

How to train a part-based object detector?

- Linear SVM (convex) when $\mathbf{z}$ is fixed: $f_{\mathbf{w}} = \max_{\mathbf{z}} \mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{z})$.
- Solving by coordinate descent:
  1. Fixed $\mathbf{w}$, find the latent variable $\mathbf{z}$ for the positive examples.
  2. Fixed $\mathbf{z}$, solve the Linear SVM to find $\mathbf{w}$.

Implementation details

- Initialize root filter by training model without latent variables on unoccluded examples.
- Root filter update: get new positives (best score and significant overlap with ground truth), add to positives and retrain.
- Part initialization: place parts to cover high-energy regions of the root filter.

The detector

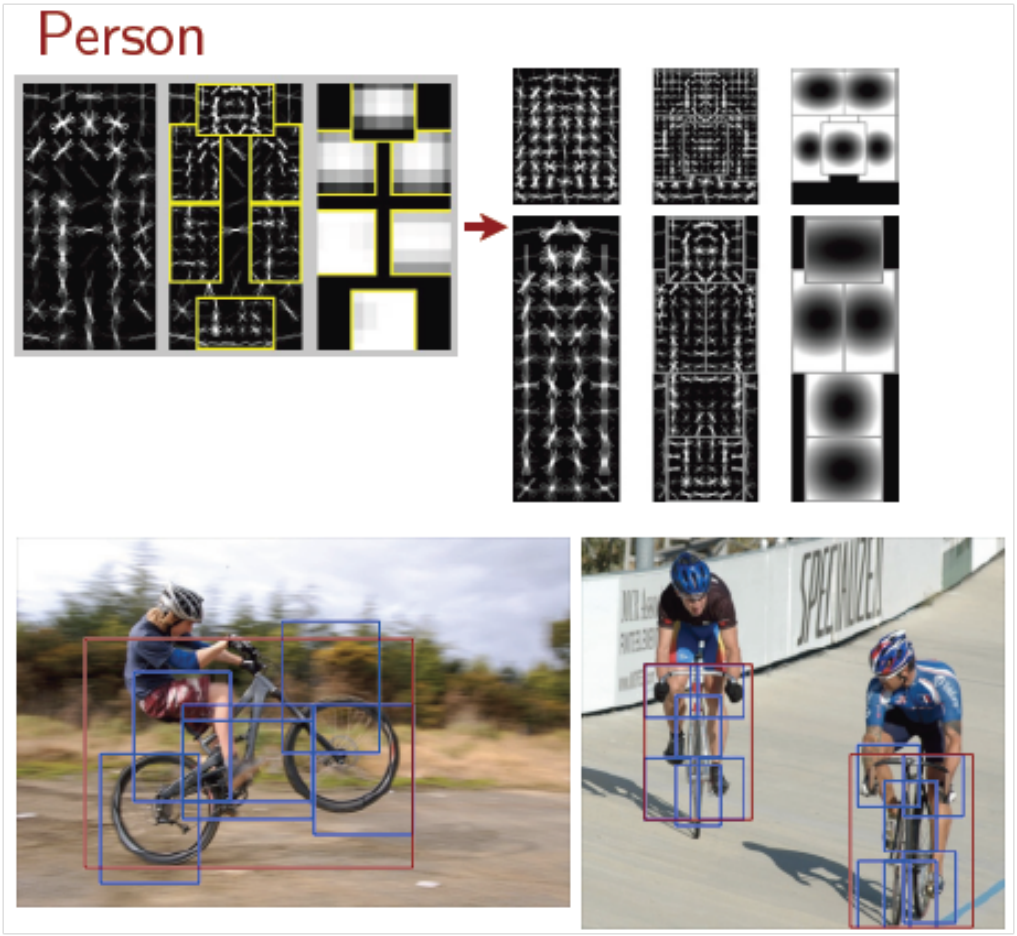


Figure : Object models amd detections.

What's next?

An unsolved problem: heavy-tailed distribution of objects:

- Many modes of object appearance: Pose, view, shape, distance, texture/color.
- Some modes are common (prototypical views), many others are not.
- Much progress due to division of categories into visual subcategories (Often high performance for common modes, poor for others).
- Learning less common modes is important for dealing with variation within and across categories.

What's next?

- Learning from a few (2?) examples.
- Learning to recognize, beyond gradients (shape representation).
- Which are the functional parts?
- Which parts have stable appearance?
- How to infer physical relations (contact, engagement, etc.)?
- How to interpret an object's role in the scene?

What's next?

Not what does it look like, but what is it?

- What is the 3D shape?
- How big is it?
- What are the functional parts?
- What are distinctive markings?
- What can it do?
- In what kind of settings is it likely to appear?
- Etc.