Overview

- Recall last class: Boosting is a way of generating a strong classifier as a weighted ensemble of weak ones
- Today: Support Vector Machine (SVM) training generates a strong classifier directly
- Case Study: Dalal and Triggs pedestrian detector

Support Vector Machines

SVM slides from Kristen Grauman, UT-Austin

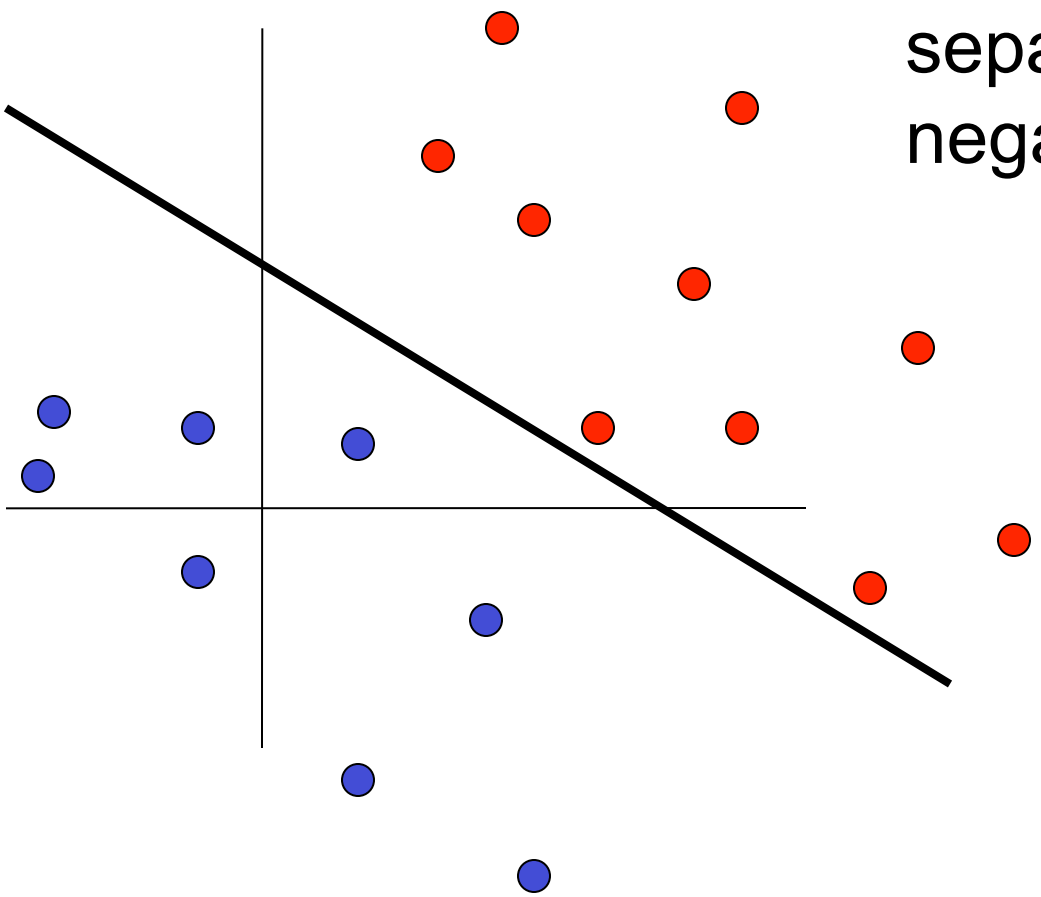
Other good resources:

Presentation slides from Christoph Lampert:
https://sites.google.com/site/christophlampert/teaching/kernel-methods-for-object-recognition

Simple tutorial document by Chris Williams
http://www.inf.ed.ac.uk/teaching/courses/iaml/docs/svm.pdf

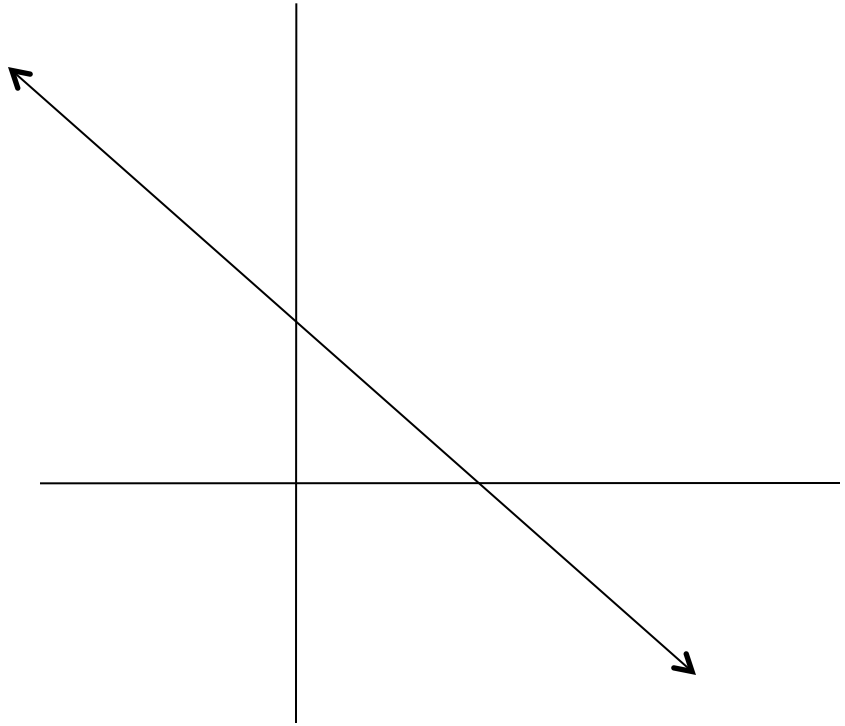
Video lecture by Pat Winston:
https://www.youtube.com/watch?v=_PwhiWxHK8o

Linear classifiers



Find linear function to separate positive and negative examples

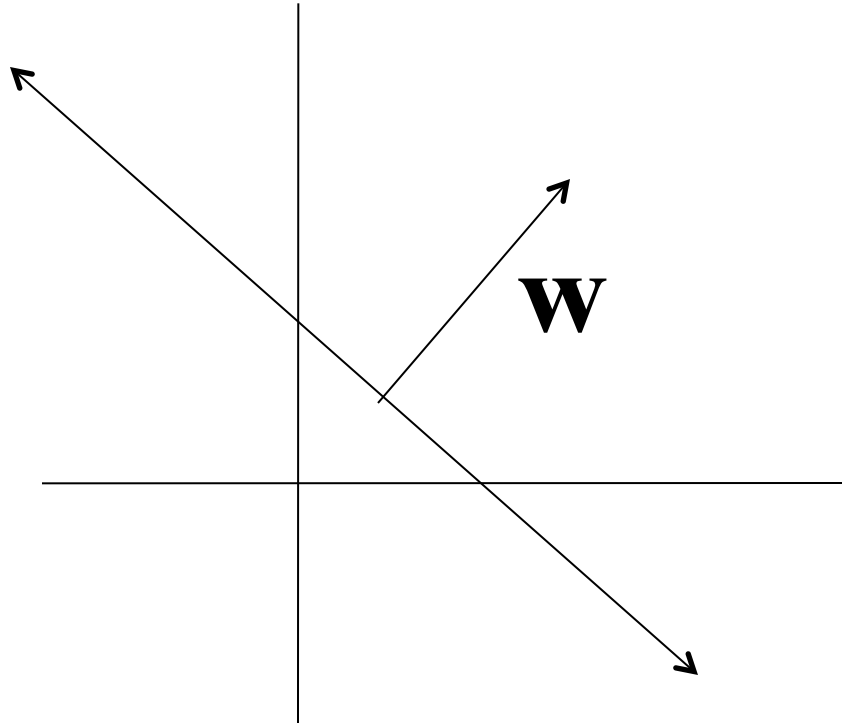
Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

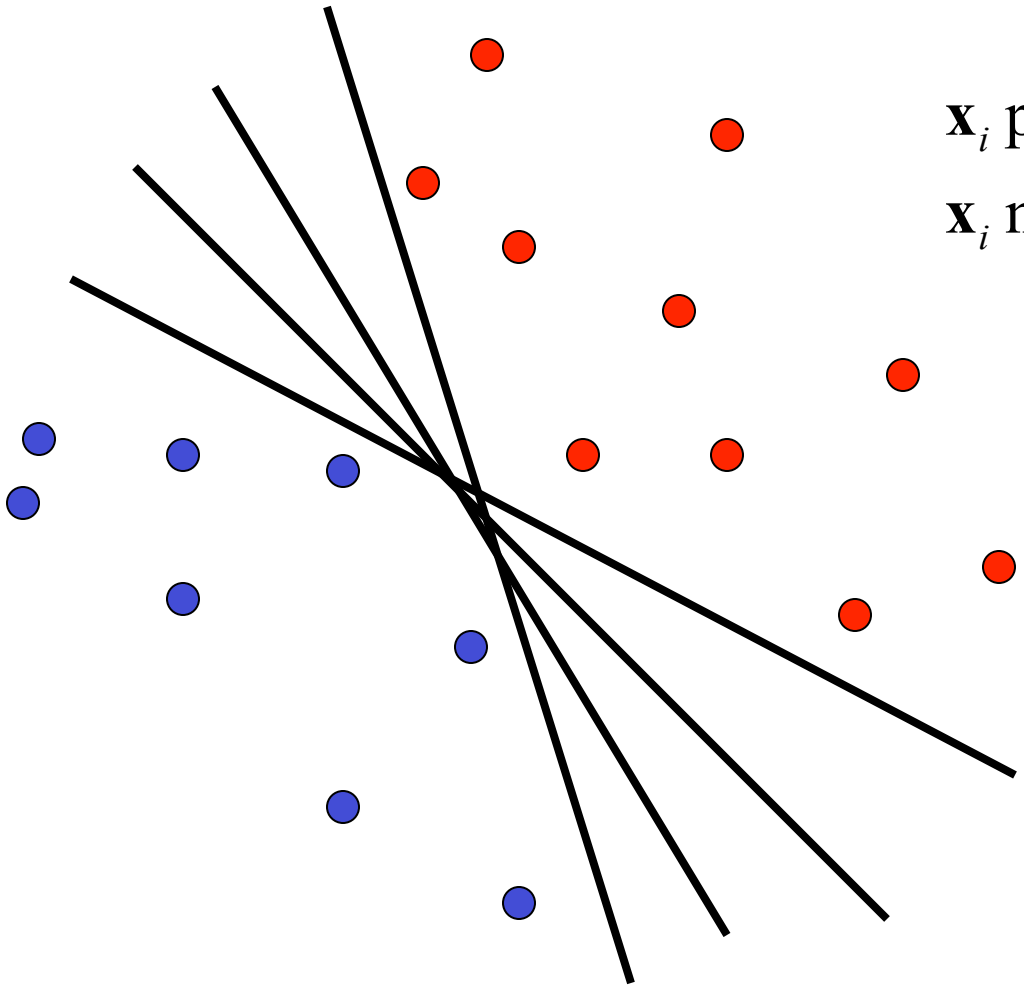
$$ax + cy + b = 0$$

$$\updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

Linear classifiers

- Find linear function to separate positive and negative examples

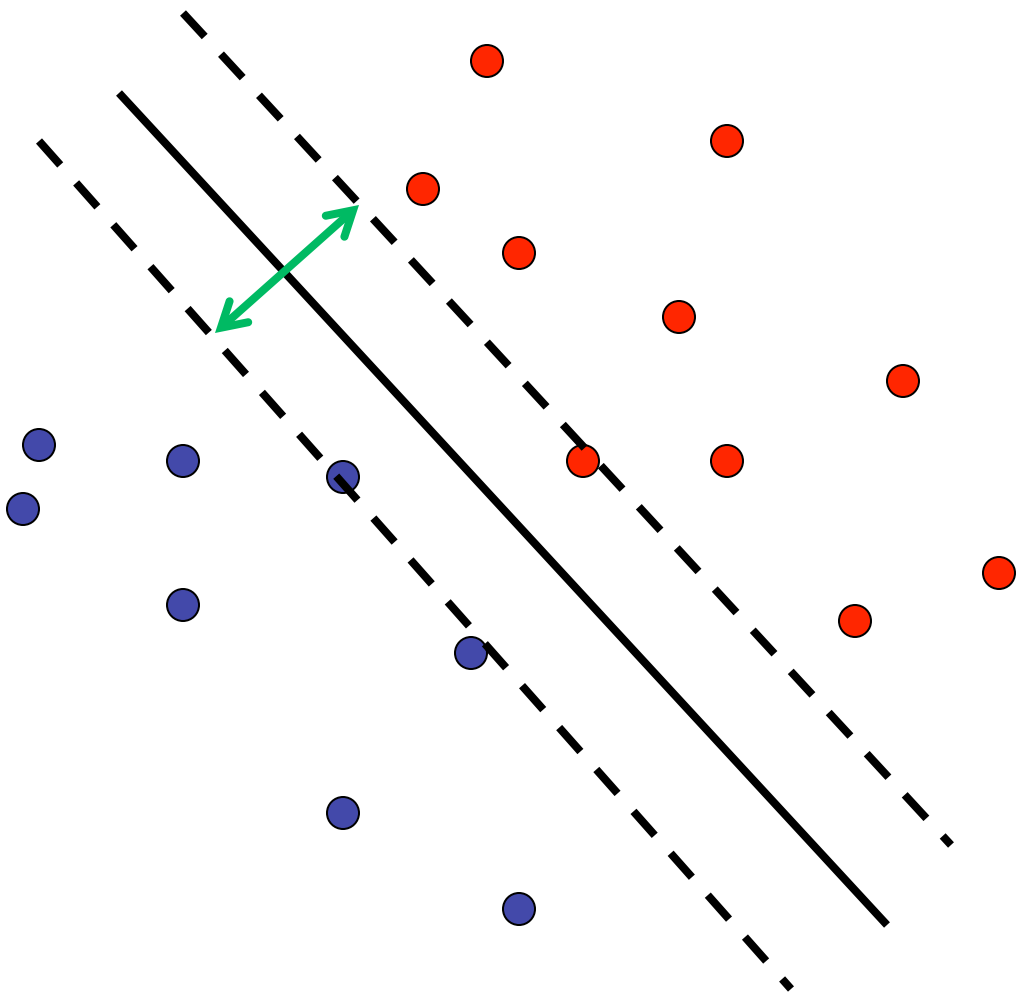


$$\mathbf{x}_i \text{ positive}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$

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

Which line is best?

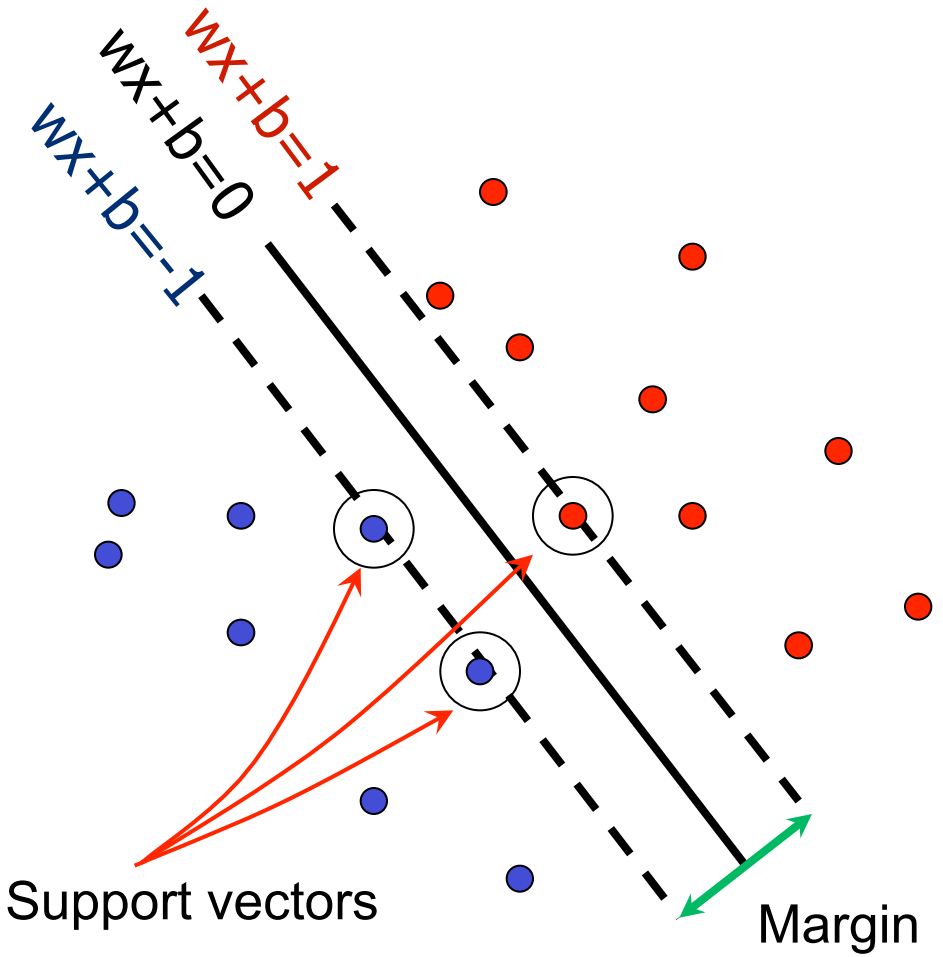
Support Vector Machines (SVMs)



- Discriminative classifier based on *optimal separating line (for 2d case)*
- Maximize the *margin* between the positive and negative training examples

Support vector machines

- Want line that maximizes the margin.



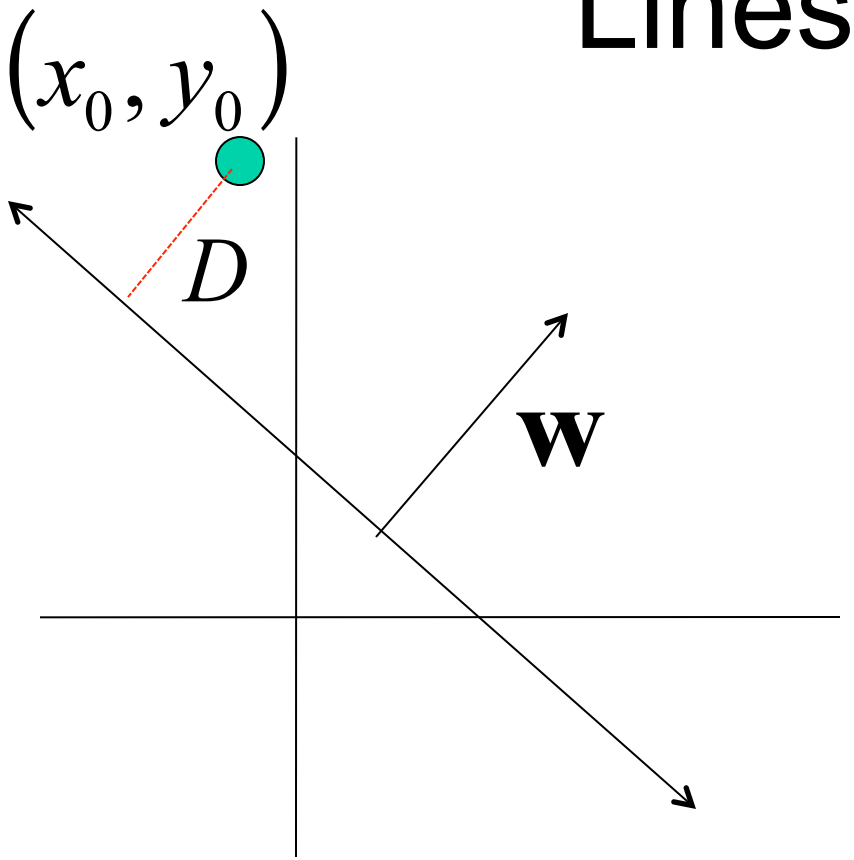
$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

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

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

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

Lines in R^2



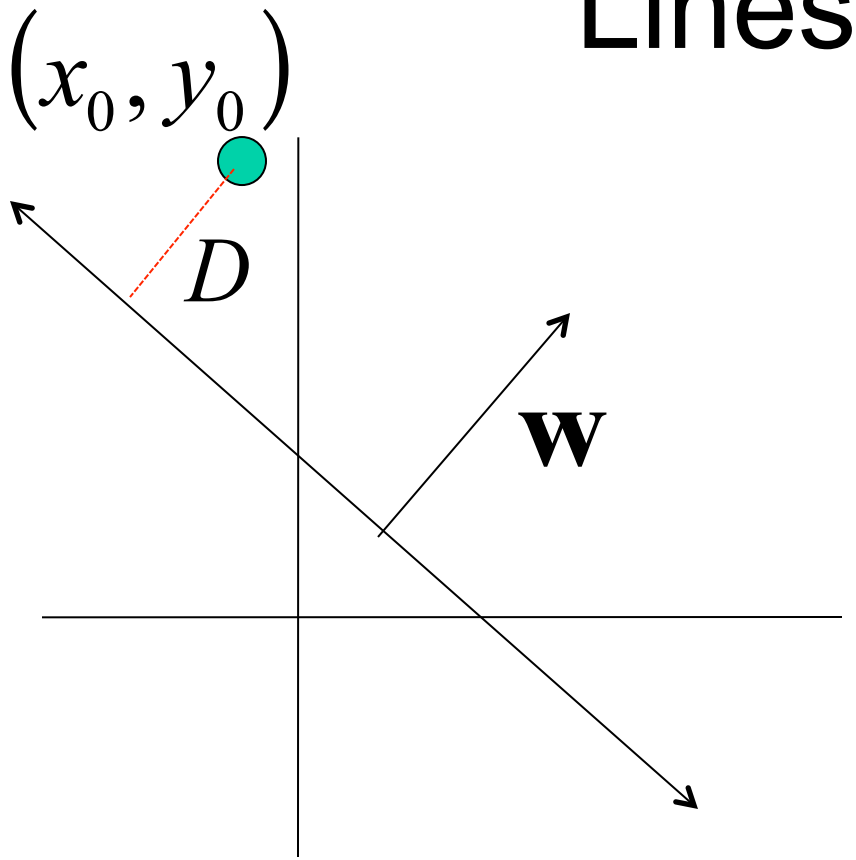
Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

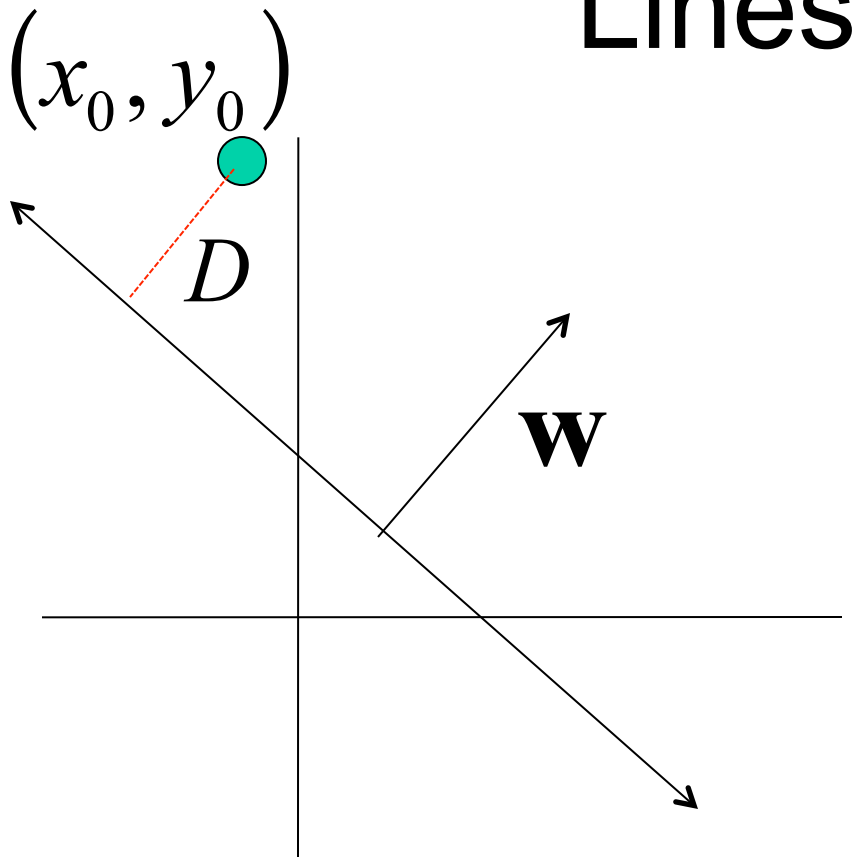
$$\updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

$$D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}}$$

distance from point to line

Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\updownarrow$$

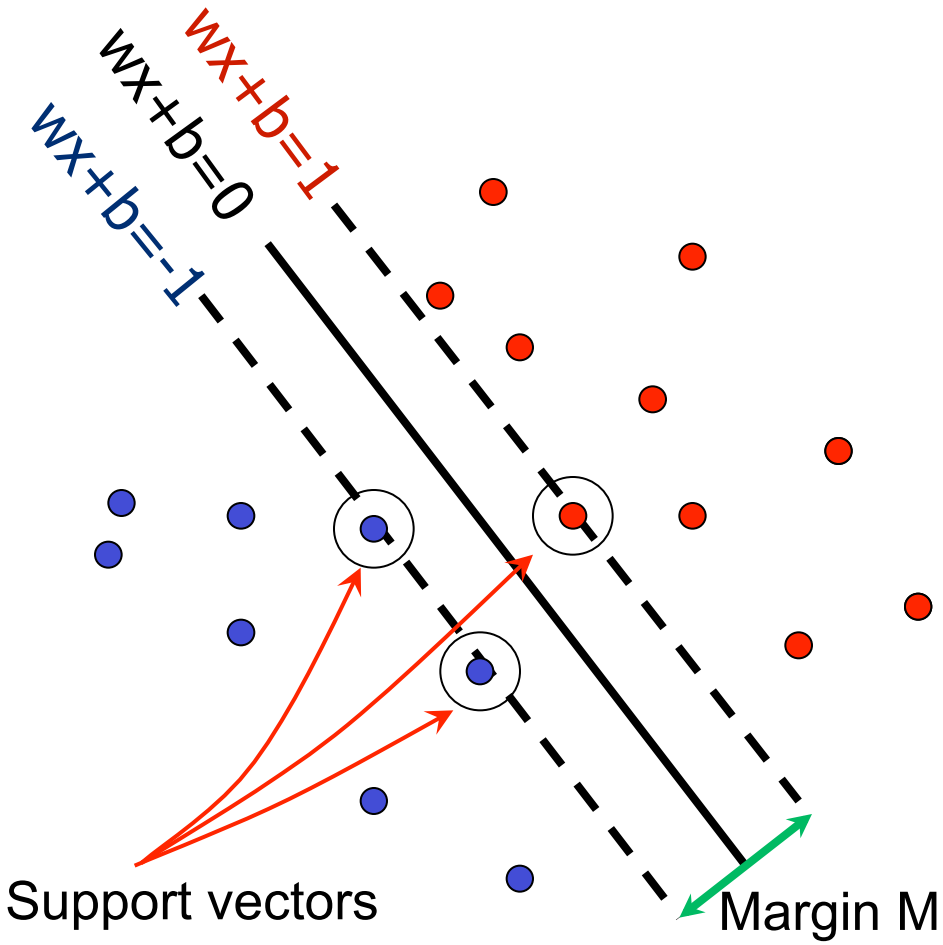
$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

$$D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}} = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{x} + b}{\|\mathbf{w}\|}$$

distance from point to line

Support vector machines

- Want line that maximizes the margin.



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

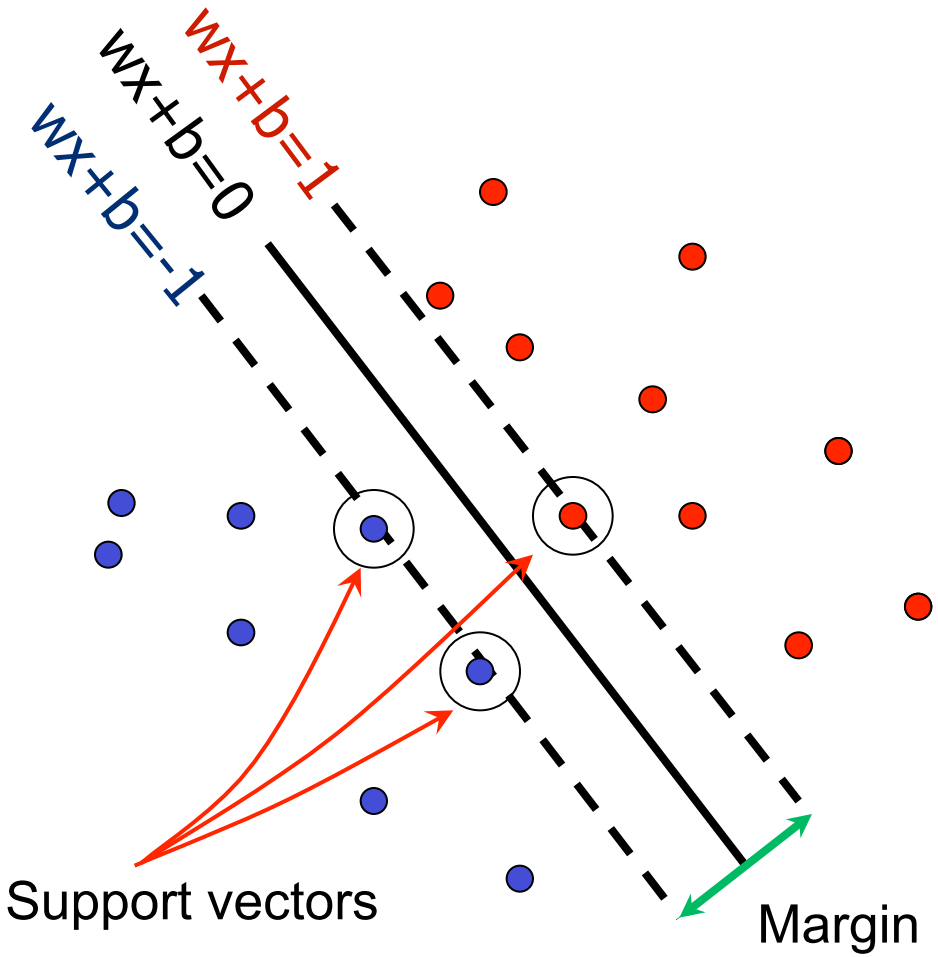
Distance between point and line: $\quad \dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

For support vectors:

$$\frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|} = \frac{\pm 1}{\|\mathbf{w}\|} \qquad M = \left| \frac{1}{\|\mathbf{w}\|} - \frac{-1}{\|\mathbf{w}\|} \right| = \frac{2}{\|\mathbf{w}\|}$$

Support vector machines

- Want line that maximizes the margin.



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

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

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

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

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

Finding the maximum margin line

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

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

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

Quadratic optimization problem:

$$\text{Minimize} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$\text{Subject to} \quad y_i(\mathbf{w}\cdot\boldsymbol{x}_i + b) \geq 1$$

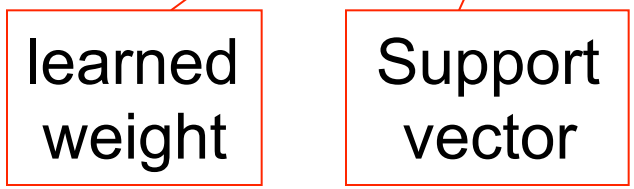
One constraint for each training point.

Note sign trick.

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1

Finding the maximum margin line

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



C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1

Finding the maximum margin line

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

  $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ (for any support vector)

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

- Classification function:

  $$f(x) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + \text{b})$$

  $$= \text{sign}\left(\sum_i \alpha_i yi \mathbf{x}_i \cdot \mathbf{x} + b\right)$$

  If f(x) < 0, classify as negative, if f(x) > 0, classify as positive

- Notice that it relies on an *inner product* between the test point ***x*** and the support vectors $\boldsymbol{x}_i$
- (Solving the optimization problem also involves computing the inner products $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ between all pairs of training points)

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1

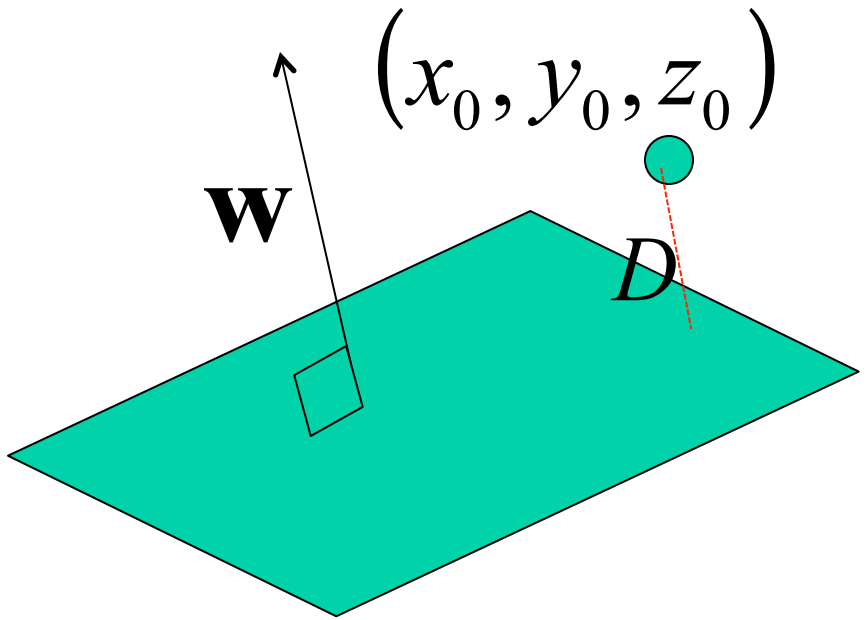
Questions

- **What if the features are not 2d?**
- What if the data is not linearly separable?
- What to do for more than two classes?

Questions

- **What if the features are not 2d?**
  - **Generalizes to d-dimensions – replace line with "hyperplane"**
- What if the data is not linearly separable?
- What to do for more than two classes?

Planes in $\mathbb{R}^3$



Let $\mathbf{w} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

$$ax + by + cz + d = 0$$

$$\updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + d = 0$$

$$D = \frac{|ax_0 + by_0 + cz_0 + d|}{\sqrt{a^2 + b^2 + c^2}} = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{x} + d}{\|\mathbf{w}\|}$$

distance from point to plane

Hyperplanes in R^n

Hyperplane *H* is set of all vectors $\mathbf{x} \in R^n$ which satisfy:

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b = 0$$

$$\updownarrow$$

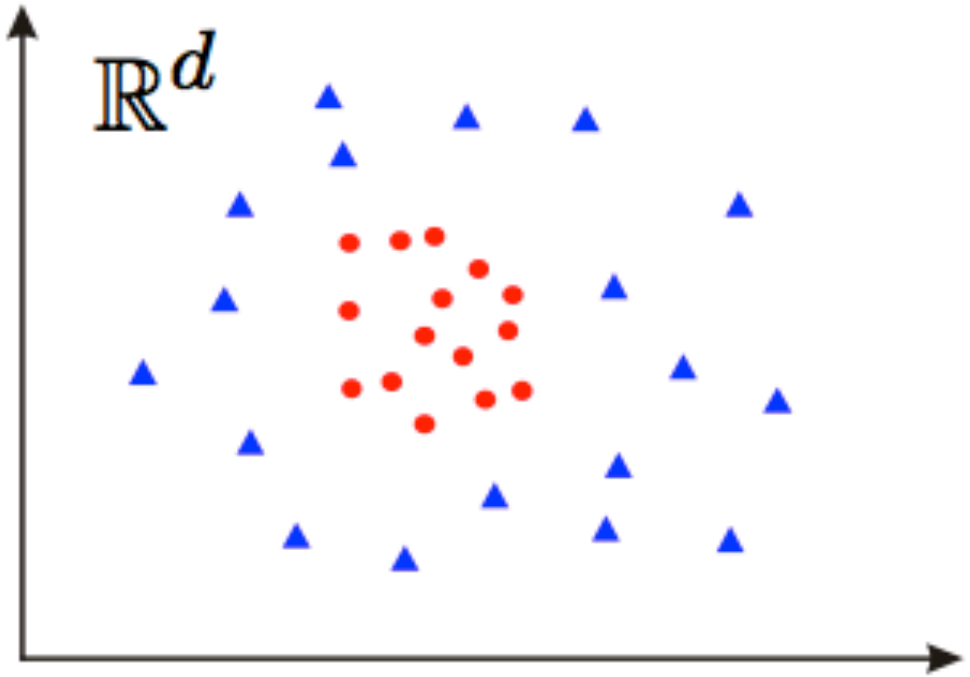
$$\mathbf{w}^{\mathrm{T}} \mathbf{x} + b = 0$$

$$D(H, \mathbf{x}) = \frac{\mathbf{w}^{\mathrm{T}} \mathbf{x} + b}{\|\mathbf{w}\|}$$

distance from point to hyperplane

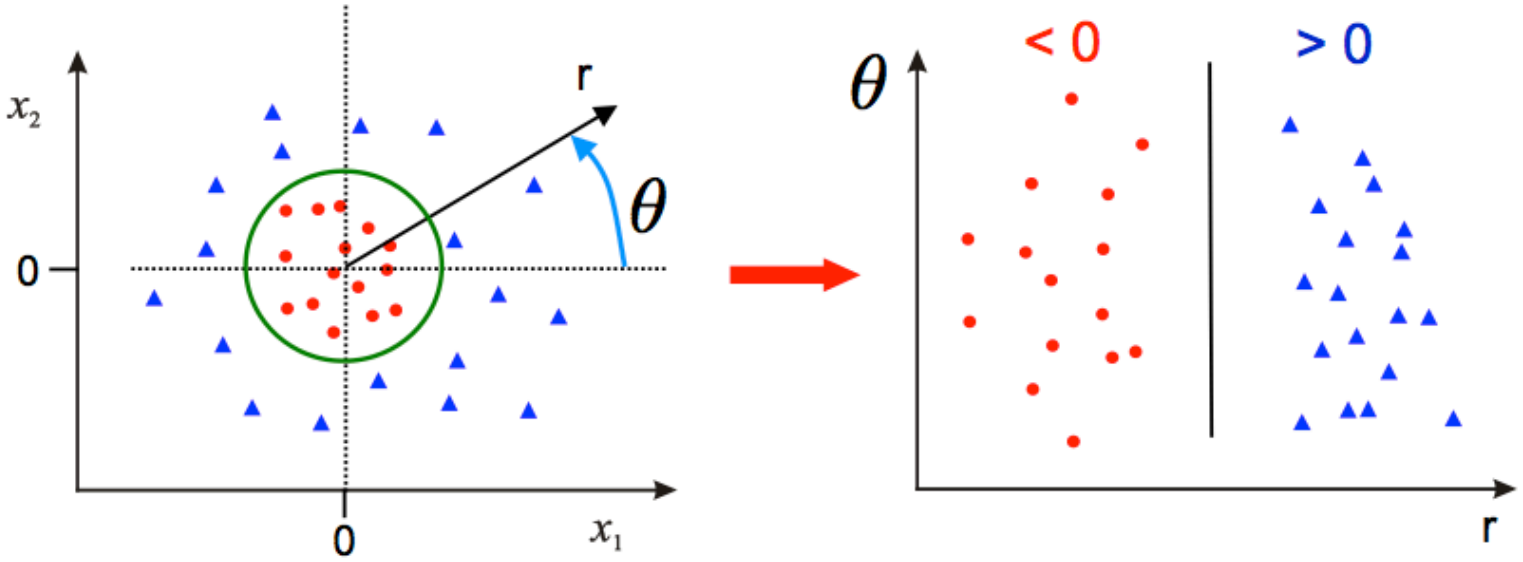
Questions

- What if the features are not 2d?
- **What if the data is not linearly separable?**



Nonlinear SVMs

Solution 1: use polar coordinates



- Data is linearly separable in polar coordinates
- Acts non-linearly in original space

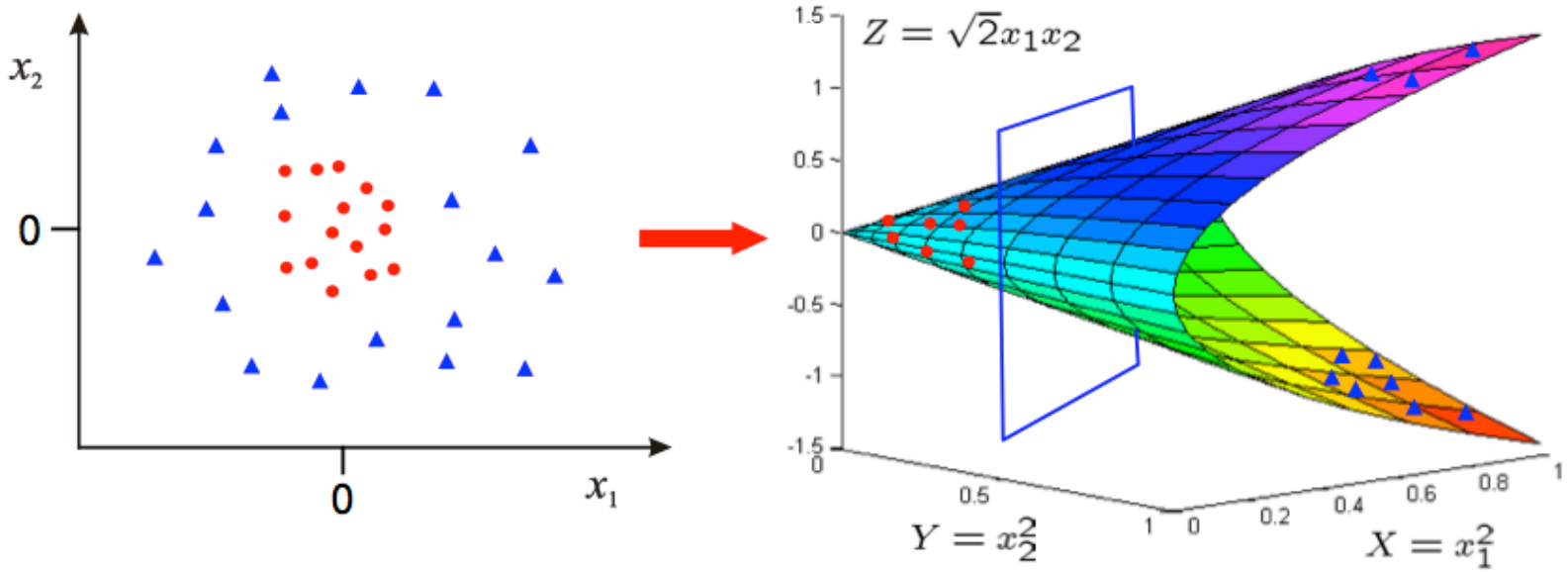
$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \theta \end{pmatrix} \qquad \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

Slide from Andrew Zisserman

Nonlinear SVMs

Solution 2: map data to higher dimension

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \qquad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

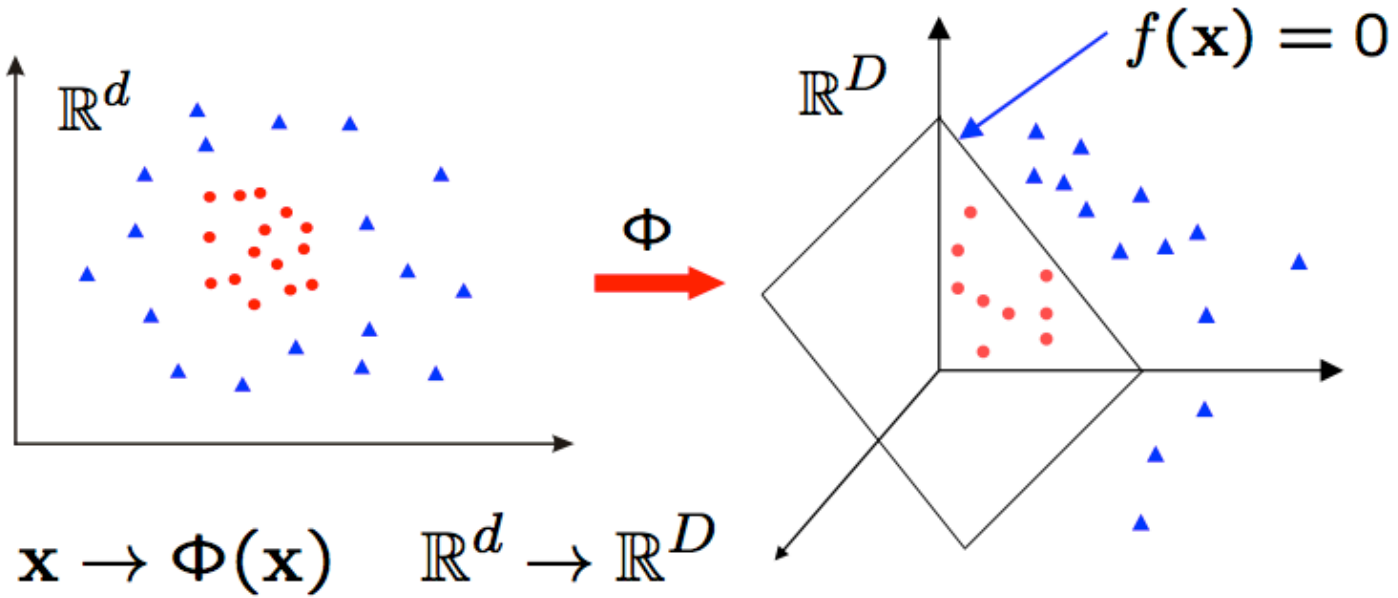


- Data is linearly separable in 3D
- This means that the problem can still be solved by a linear classifier

Slide from Andrew Zisserman

Nonlinear SVMs

SVM classifiers in a transformed feature space



$\Phi : \mathbf{x} \rightarrow \Phi(\mathbf{x}) \quad \mathbb{R}^d \rightarrow \mathbb{R}^D$

Learn classifier linear in $\mathbf{w}$ for $\mathbb{R}^D$:

$$f(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x}) + b$$

$\Phi(\mathbf{x})$ is a feature map

Slide from Andrew Zisserman

The Kernel Trick

- Recall we transformed linear regression into nonlinear regression using a feature vector $\Phi(x)$ and, ultimately, the "kernel trick."

- We also use the kernel trick here to transform a linear classifier into nonlinear one.

Example Kernel

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \quad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$\begin{aligned} \Phi(\mathbf{x})^\top \Phi(\mathbf{z}) &= \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right) \begin{pmatrix} z_1^2 \\ z_2^2 \\ \sqrt{2}z_1z_2 \end{pmatrix} \\ &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 \\ &= (x_1z_1 + x_2z_2)^2 \\ &= (\mathbf{x}^\top \mathbf{z})^2 \end{aligned}$$

Kernel Trick

- Classifier can be learnt and applied without explicitly computing $\Phi(\mathbf{x})$
- All that is required is the kernel $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z})^2$
- Complexity of learning depends on N (typically it is $O(N^3)$) not on D

Slide from Andrew Zisserman

Example Kernels

- Linear kernels $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$
- Polynomial kernels $k(\mathbf{x}, \mathbf{x}') = \left(1 + \mathbf{x}^\top \mathbf{x}'\right)^d$ for any $d > 0$
  - Contains all polynomials terms up to degree d
- Gaussian kernels $k(\mathbf{x}, \mathbf{x}') = \exp\left(-||\mathbf{x} - \mathbf{x}'||^2 / 2\sigma^2\right)$ for $\sigma > 0$
  - Infinite dimensional feature space

Slide from Andrew Zisserman

Nonlinear SVMs

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

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$$

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

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

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

Questions

- What if the features are not 2d?
- What if the data is not linearly separable?
- **What to do for more than two classes?**

Multi-class SVMs

- Achieve multi-class classifier by combining a number of binary classifiers

- **One vs. all**
  - Training: learn an SVM for each class vs. the rest
  - Testing: apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value
- **One vs. one**
  - Training: learn an SVM for each pair of classes
  - Testing: each learned SVM "votes" for a class to assign to the test example

Software for SVMs

There exist many implementations of maximum margin classification with and without support for arbitrary kernel functions:

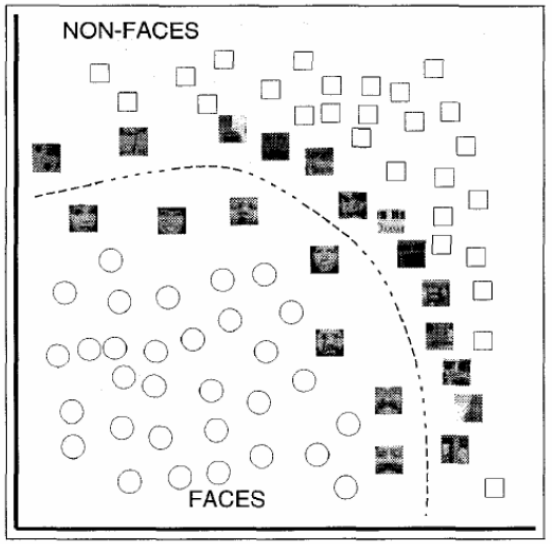
- General purpose SVMs packages: `libSVM`[1] `SVMlight`[2],
  - source code, applications, wrappers for Matlab, Python, R...
  - standard kernels are included, user kernels can be defined
  - sparse data format (sometime annoying for vision data)
- Very fast implementations for linear kernels: SVMlin[3], SGD[4]
- Larger toolboxes with wrappers for `SVMlight` or `libSVM`: MLPy, Orange, PyML, Shogun, Spider, Torch, YALE, Weka...

---

[1] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`
[2] `http://svmlight.joachims.org/`
[3] `http://people.cs.uchicago.edu/~vikass/svmlin.html`
[4] `http://leon.bottou.org/projects/sgd`

SVMs for recognition

1. Define a vector representation for each example.
2. Select a kernel function.
3. Compute pairwise kernel values between labeled examples
4. Given this “kernel matrix” to SVM optimization software to identify support vectors & weights.
5. To classify a new example: compute kernel values between new input and support vectors, apply weights, check sign of output.



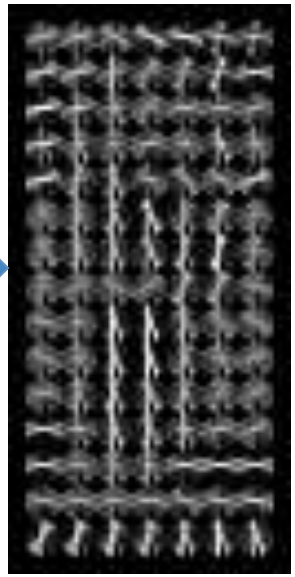
Case Study: Pedestrian Detector

Navneet Dalal and Bill Triggs,
"Histograms of Oriented Gradients for Human Detection," CVPR 2005

Dalal and Triggs CVPR'05

- Detect upright pedestrians
- Histogram of oriented gradient feature vector
- Linear SVM classifier; sliding window detector

64X128



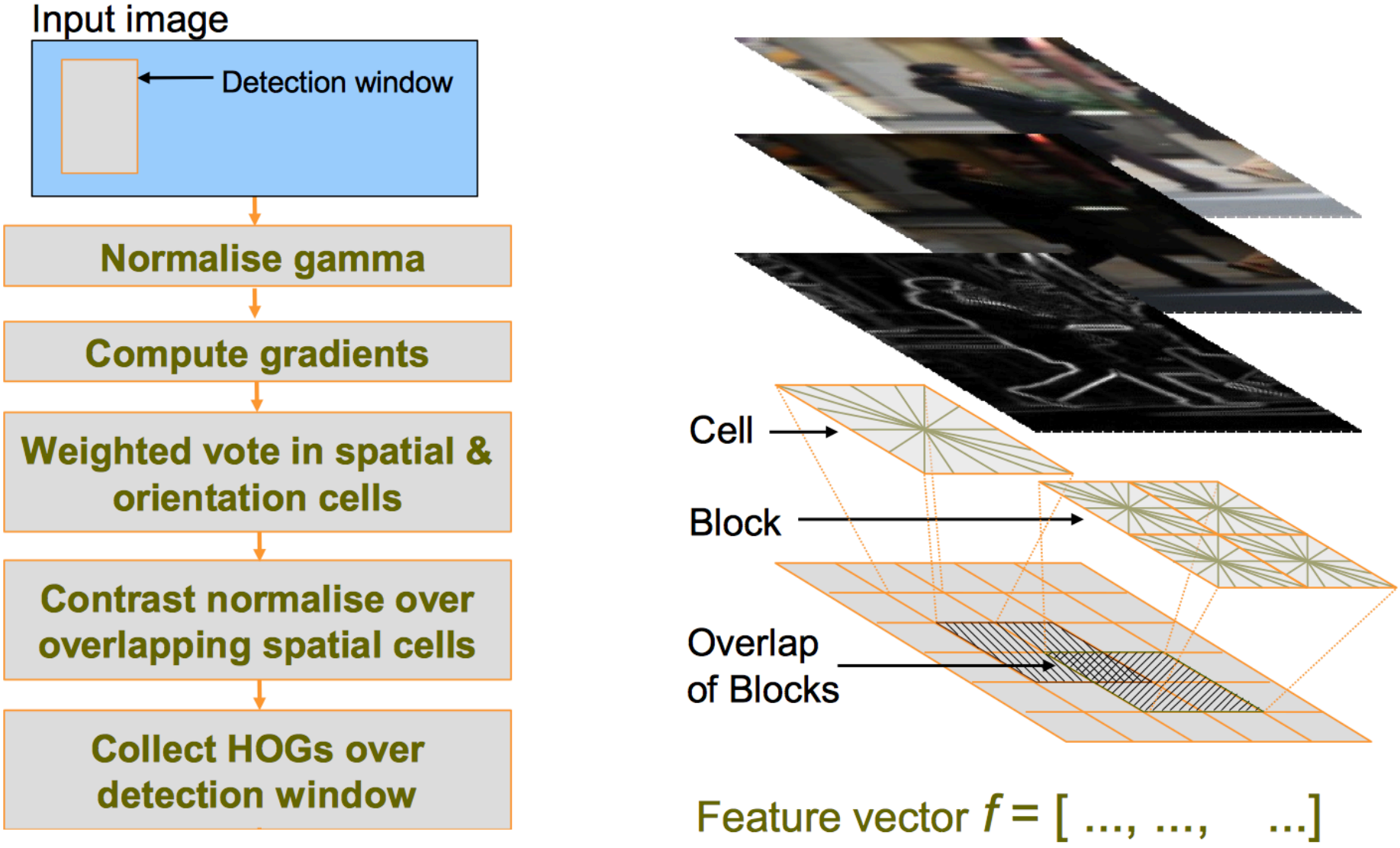
HoG descriptor

$\mathbf{x}_i \in \mathbb{R}^d, \text{ with } d = 1024$

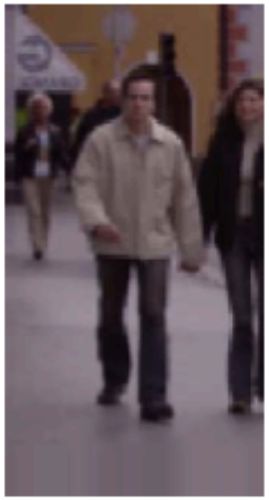
Sliding window classifier

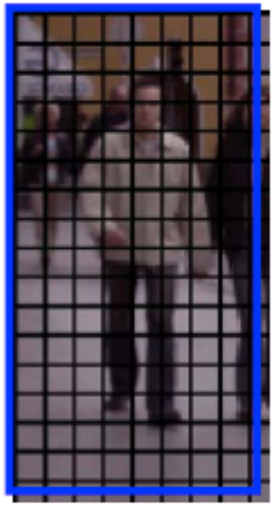
$$f(x) = \mathbf{w}^\top \mathbf{x} + b$$

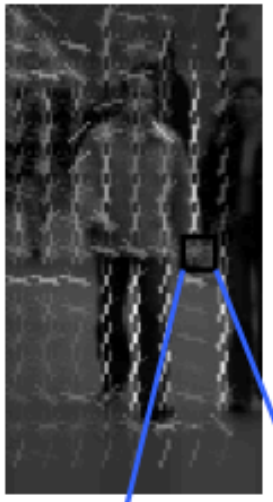
HoG Feature Extraction



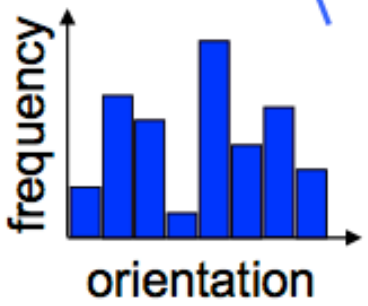
HoG Feature Extraction: Cells







- tile window into 8 x 8 pixel cells
- each cell represented by HOG



Each cell contains a histogram of gradient orientations, weighted by gradient magnitude

HoG Feature Extraction: Blocks

"Each scalar cell response contributes several components to the final descriptor vector, each normalized with respect to a different block. This may seem redundant but good normalization is critical and including overlap significantly improves the performance." Dalal&Triggs CVPR'05

HoG Design Choices

Parameters

- Gradient scale
- Orientation bins
- Block overlap area

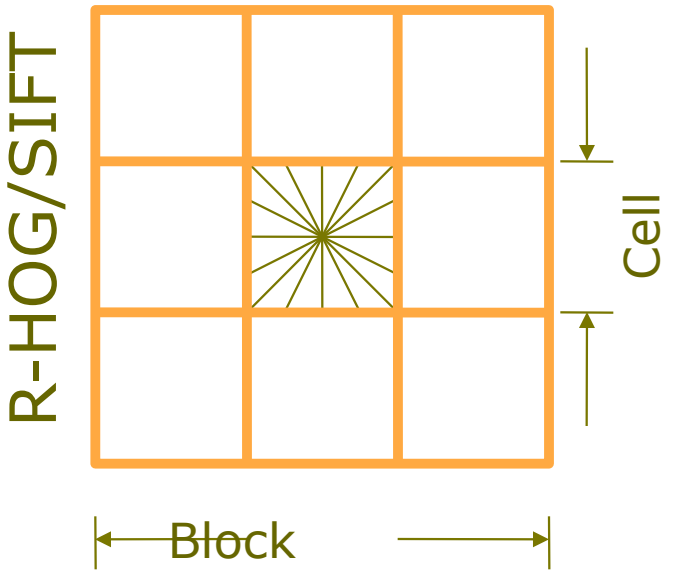
Other choices

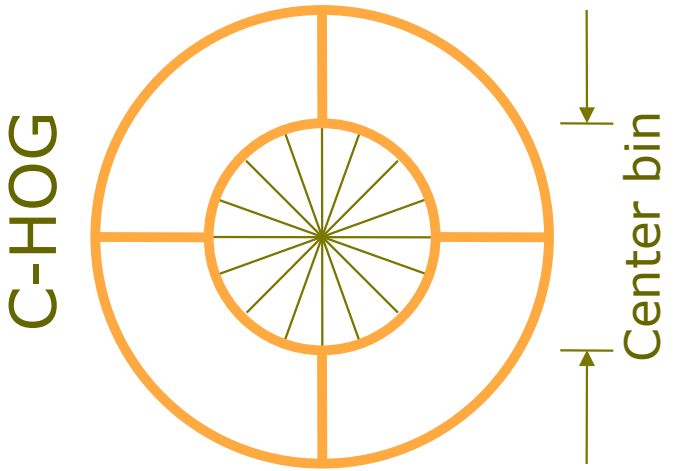
- RGB or Lab, Color/gray
- Block normalization

L2-hys, $v \leftarrow v / \sqrt{\|v\|_2^2 + \varepsilon}$

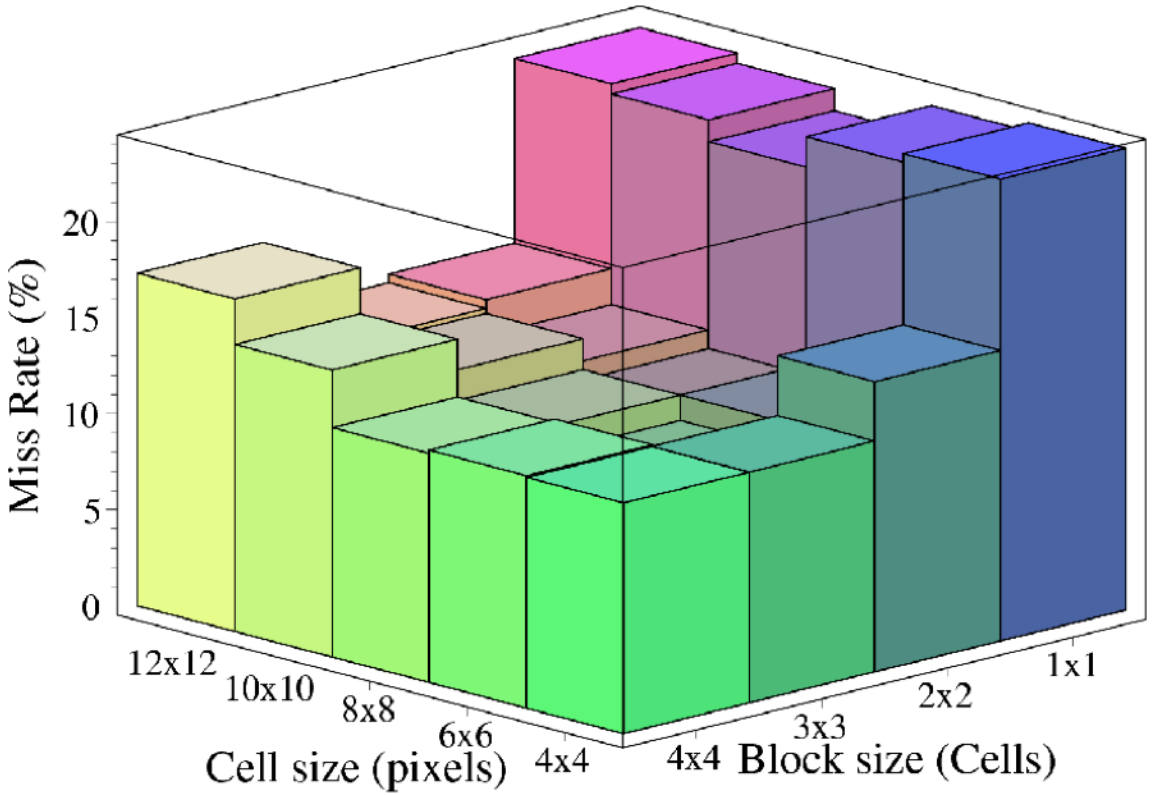
or

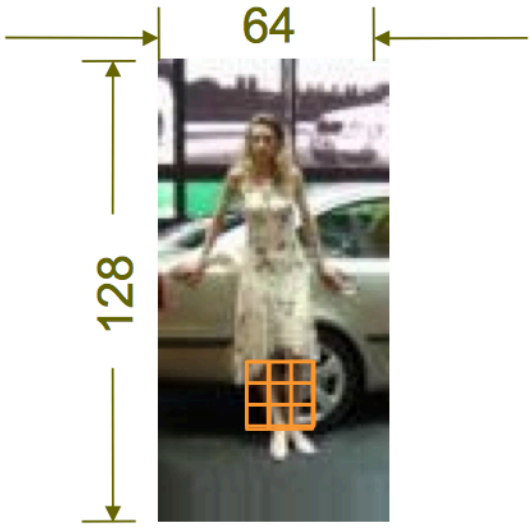
L1-sqrt, $v \leftarrow \sqrt{v / (\|v\|_1 + \varepsilon)}$





Parameter / design choices were guided by extensive experimentation to determine empirical effects on detector performance (e.g. miss rate)

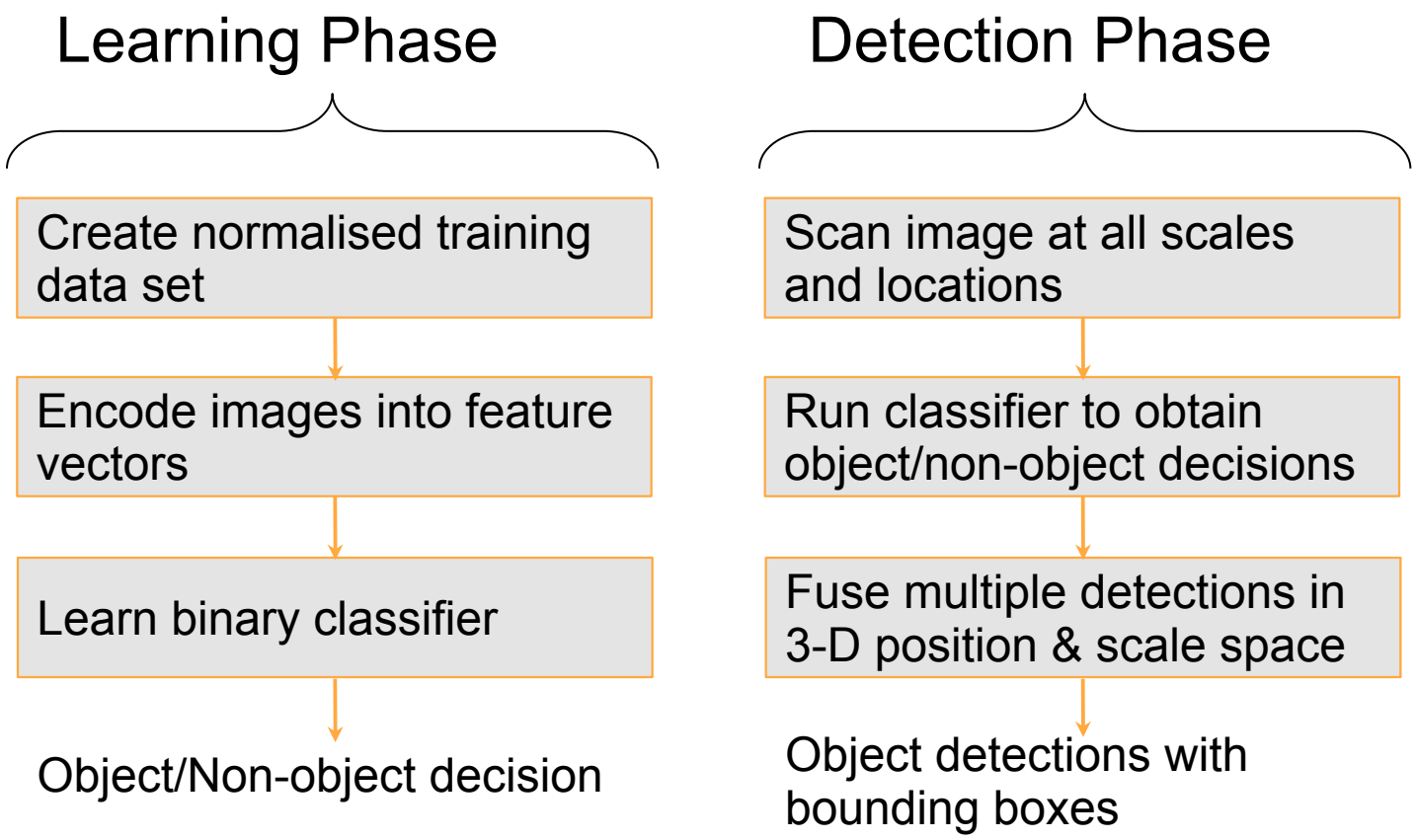




Dalal&Triggs Detector

- Default detector configuration:
  - RGB colour space with no gamma correction ;
  - [−1, 0, 1] gradient filter with no smoothing ;
  - linear gradient voting into 9 orientation bins in 0◦–180◦;
  - 16×16 pixel blocks of four 8×8 pixel cells;
  - Gaussian spatial window with σ = 8 pixel;
  - *L2-Hys* (Lowe-style clipped L2 norm) block normalization;
  - block spacing stride of 8 pixels (hence 4-fold coverage of each cell) ;
  - 64×128 detection window ;
  - linear SVM classifier.

Detector Architecture

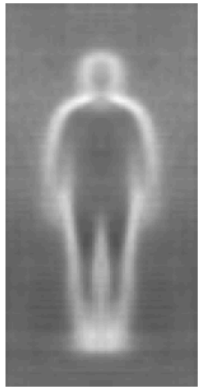
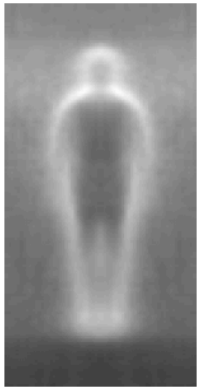


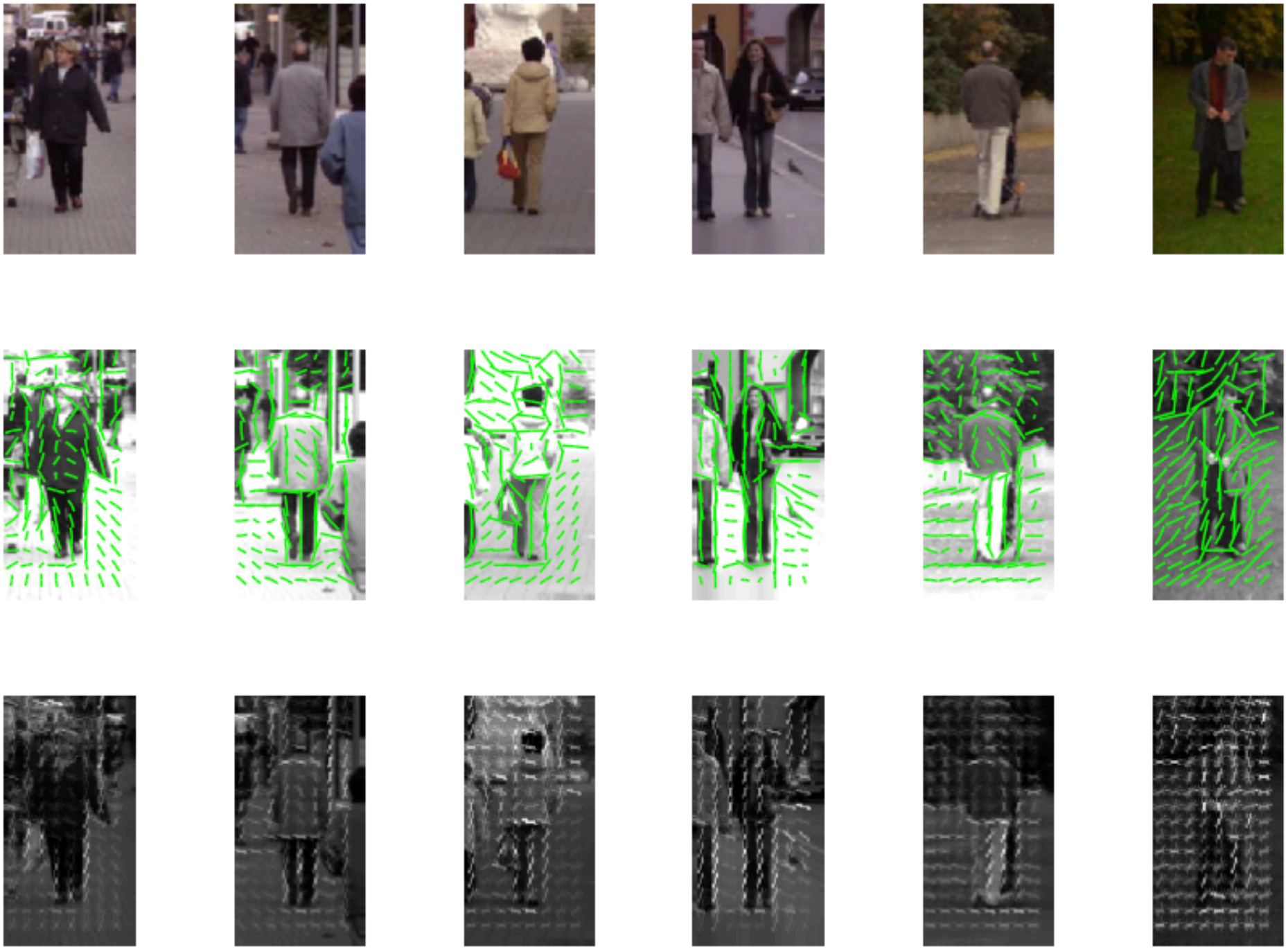
Positive and negative examples

+ thousands more…

+ millions more…

Evaluation Data Sets

MIT pedestrian database		INRIA person database	
 		 	
Train	507 positive windows Negative data unavailable	Train	1208 positive windows 1218 negative images
Test	200 positive windows Negative data unavailable	Test	566 positive windows 453 negative images
Overall 709 annotations+ reflections		Overall 1774 annotations+ reflections	



Person detection with HoG & linear SVM

Soft (C=0.01) linear SVM trained with SVMLight.

[Dalal and Triggs, CVPR 2005]

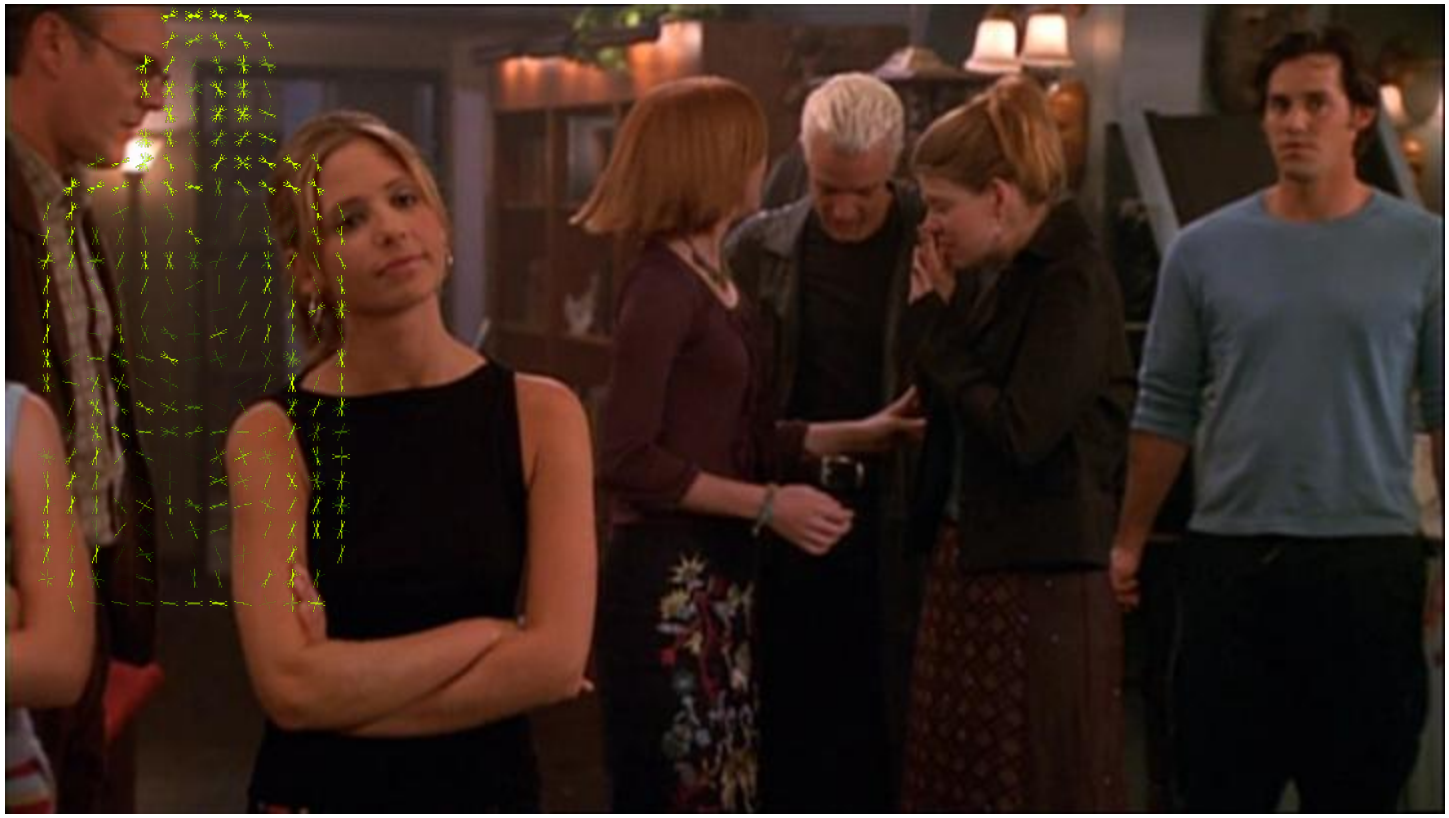
To detect people at all locations and scales:

- Sliding window using learnt HOG template
- Post-processing using non-maxima suppression

To detect people at all locations and scales:

- Sliding window using learnt HOG template
- Post-processing using non-maxima suppression

To detect people at all locations and scales:

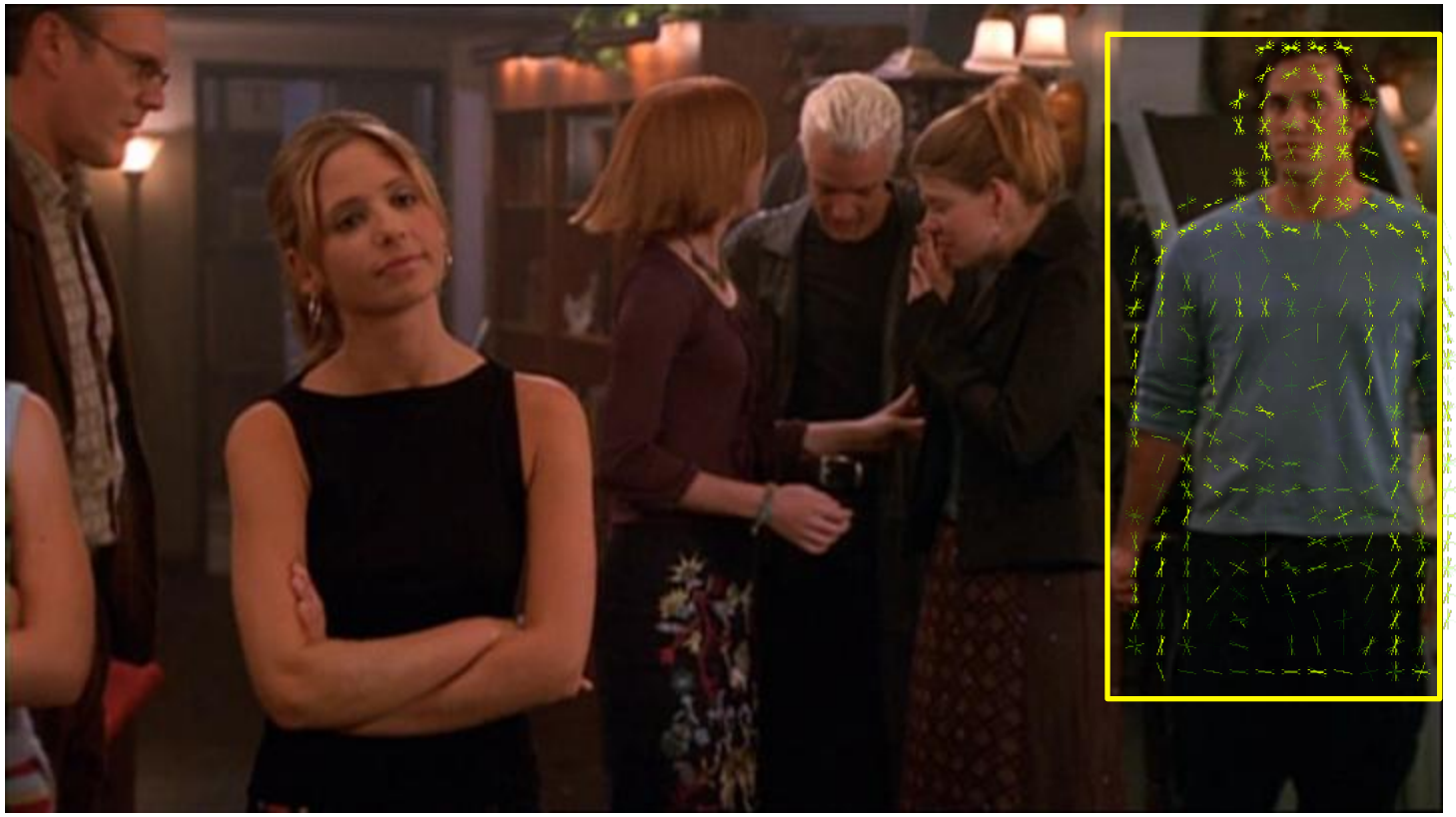


- Sliding window using learnt HOG template
- Post-processing using non-maxima suppression

To detect people at all locations and scales:

- Sliding window using learnt HOG template
- Post-processing using non-maxima suppression

To detect people at all locations and scales:

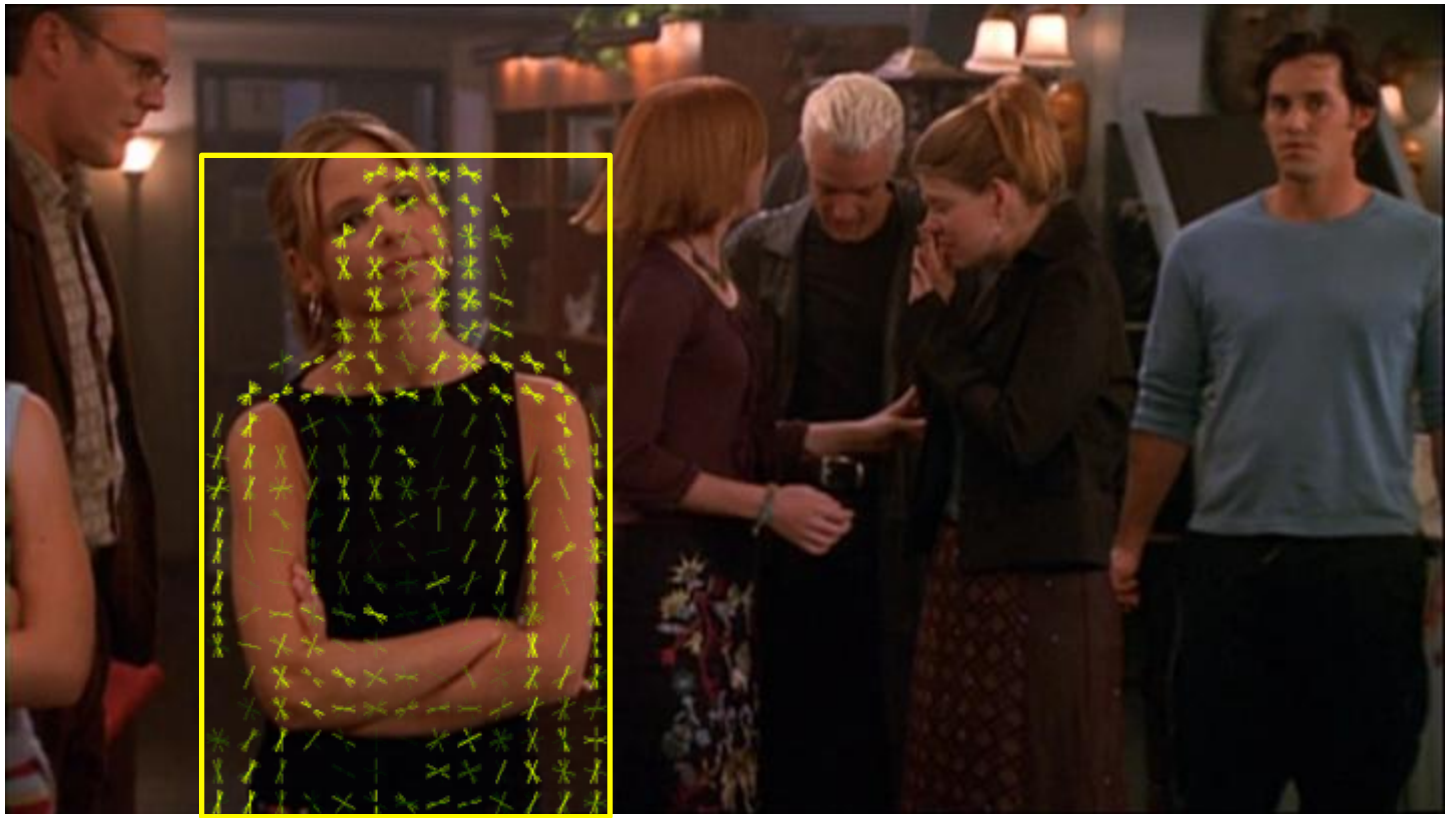


- Sliding window using learnt HOG template
- Post-processing using non-maxima suppression

To detect people at all locations and scales:

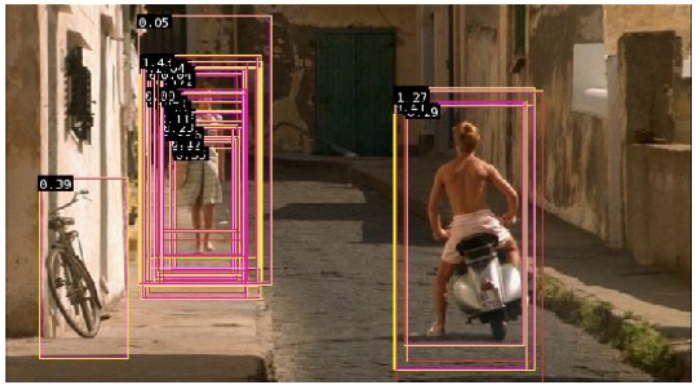
- Sliding window using learnt HOG template
- Post-processing using non-maxima suppression

To detect people at all locations and scales:



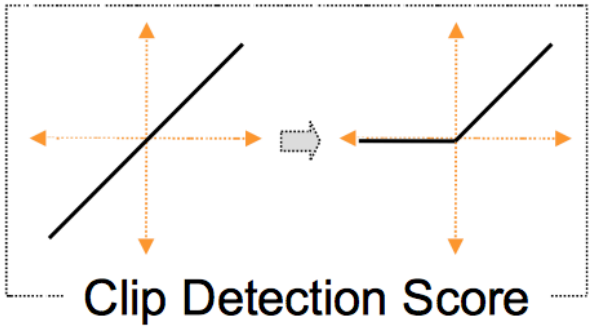
- Sliding window using learnt HOG template
- Post-processing using non-maxima suppression

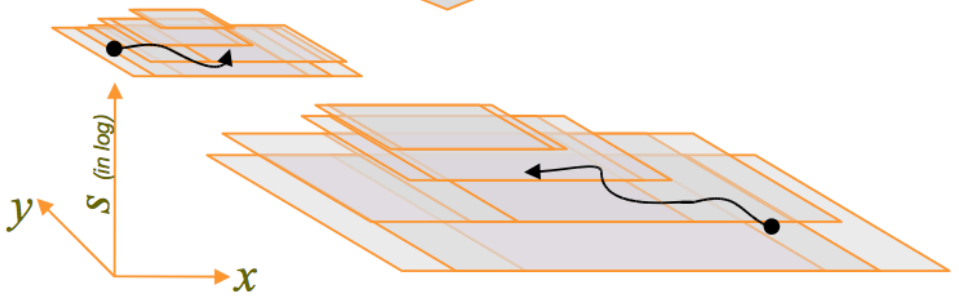
Non-maximum Suppression across Scales



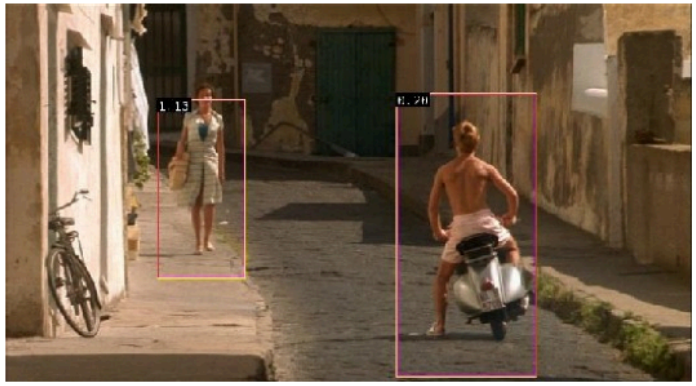
Multi-scale dense scan of detection window

Bias





Threshold



Final detections

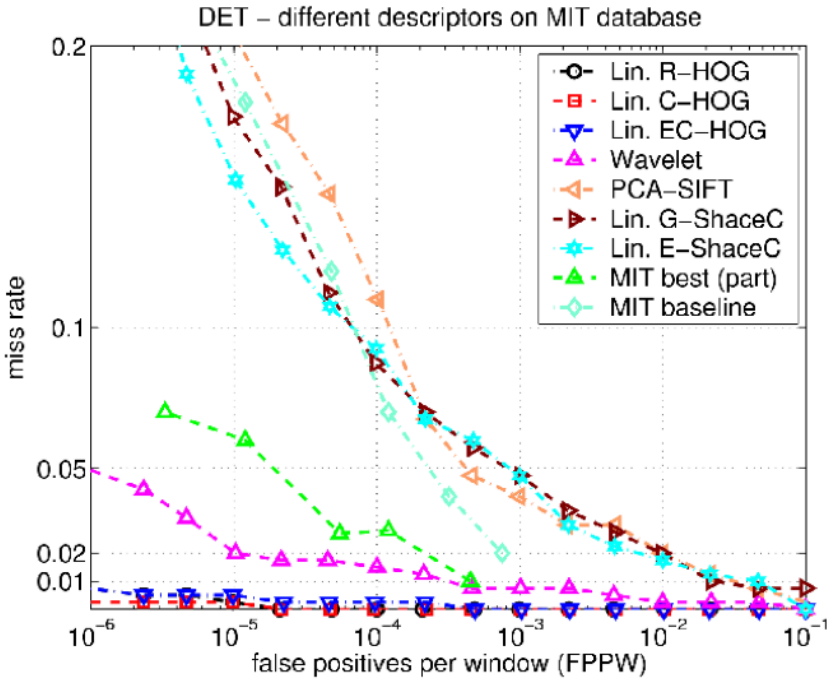
$$\mathbf{H}_i = [\exp(s_i)\sigma_x, \exp(s_i)\sigma_y, \sigma_s]$$

$$f(\mathbf{x}) = \sum_i^n w_i \exp\left(-\left\|(\mathbf{x}-\mathbf{x}_i)/\mathbf{H}_i^{-1}\right\|^2/2\right)$$

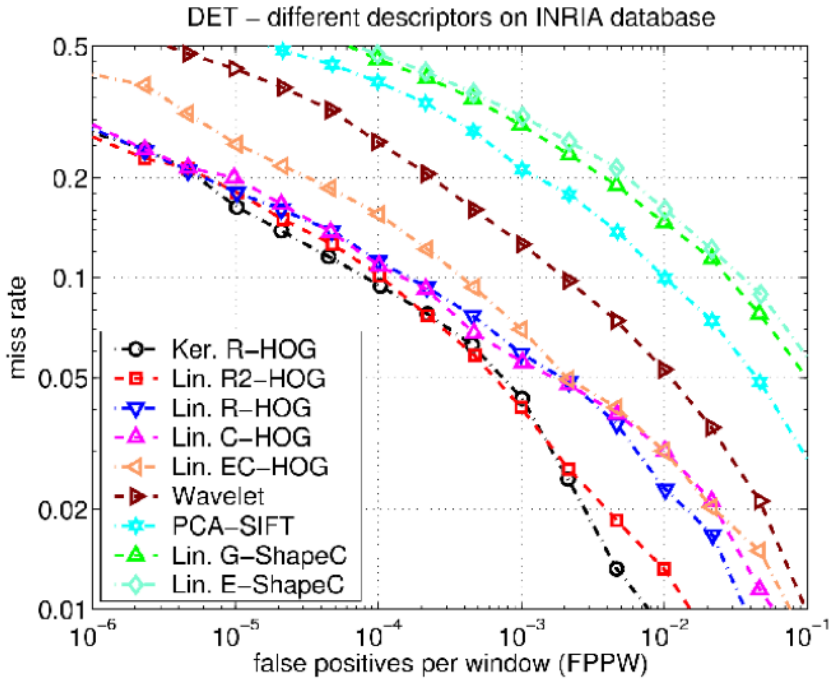
Apply robust mode detection, like mean shift

Overall Performance

MIT pedestrian database



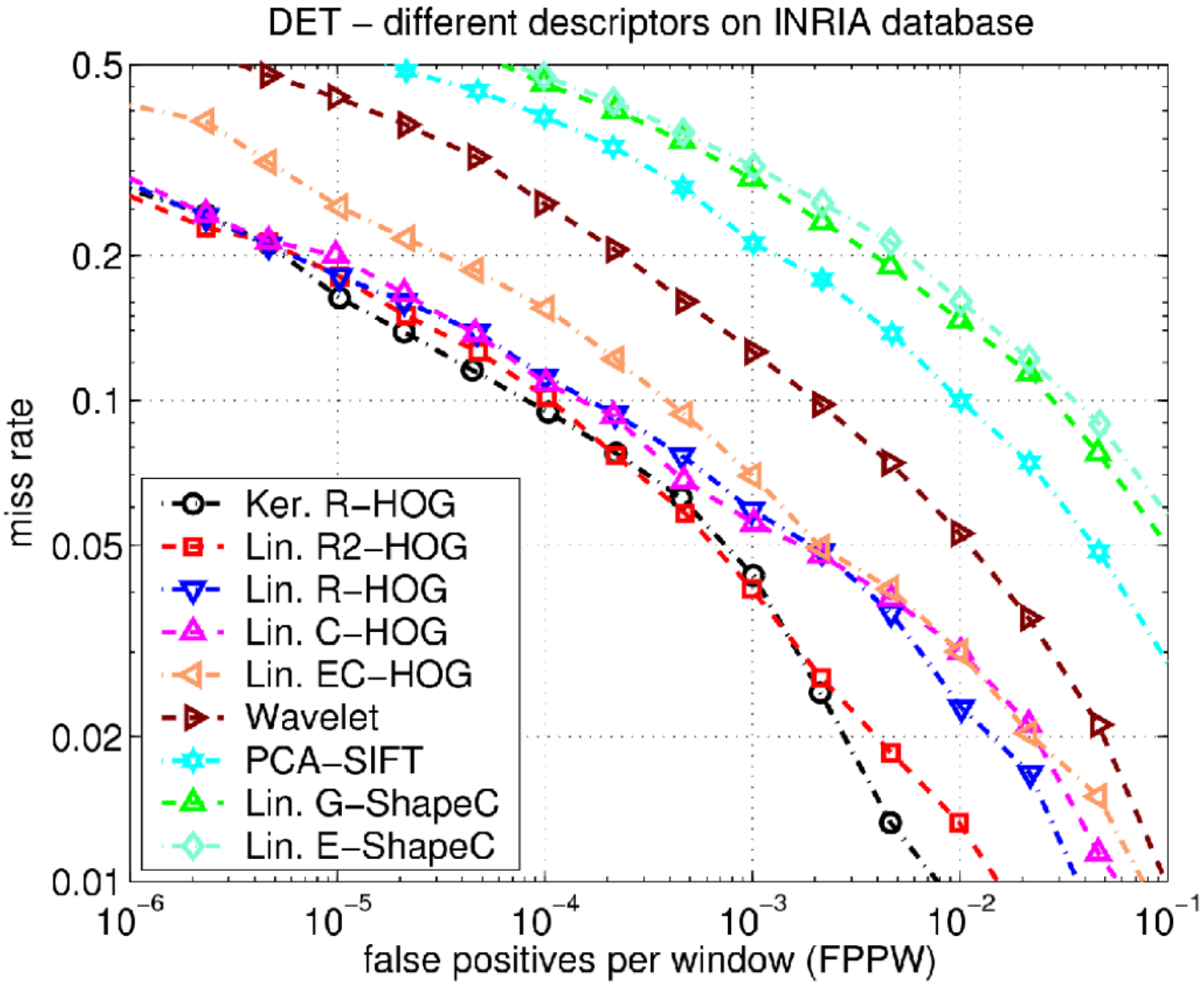
INRIA person database



R/C-HOG give near perfect separation on MIT database

Have 1-2 order lower false positives than other descriptors

Performance on INRIA Database



Dalal and Triggs Summary

- HoG feature representation
- Linear SVM classifier; sliding window detector
- Non-maximum suppression across scale
- Use of detector performance metrics
  to guide turning of system parameters
- Detection rate 90% at 10^{-4} FP per window
- Slower than Viola-Jones detector