# Lecture 2: The SVM classifier

C4B Machine Learning — Hilary 2011 — A. Zisserman

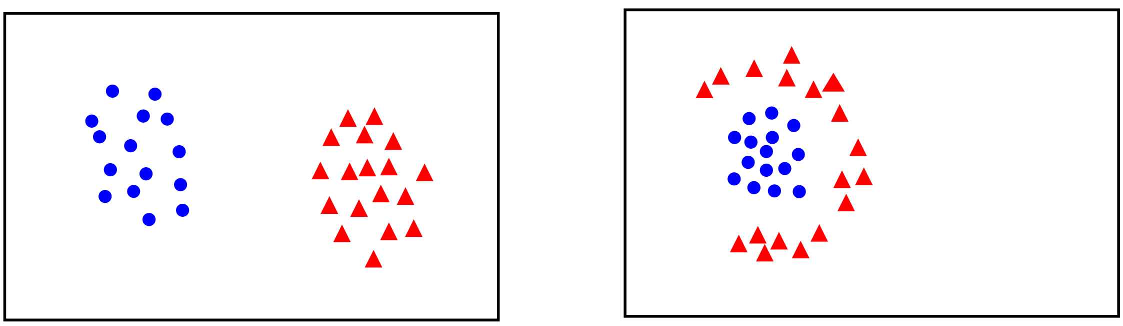
- Review of linear classifiers
  - Linear separability
  - Perceptron
- Support Vector Machine (SVM) classifier
  - Wide margin
  - Cost function
  - Slack variables
  - Loss functions revisited

## Binary Classification

Given training data $(\mathbf{x}_i, y_i)$ for $i = 1 \ldots N$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, learn a classifier $f(\mathbf{x})$ such that

$$f(\mathbf{x}_i) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

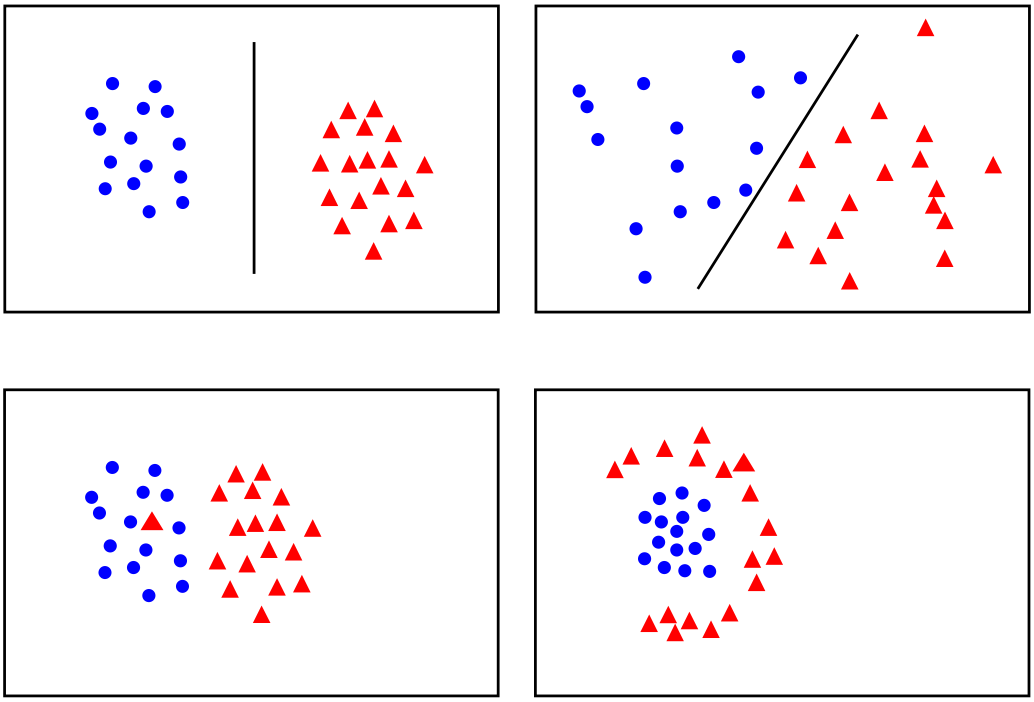
i.e. $y_i f(\mathbf{x}_i) > 0$ for a correct classification.



# Linear separability

linearly separable

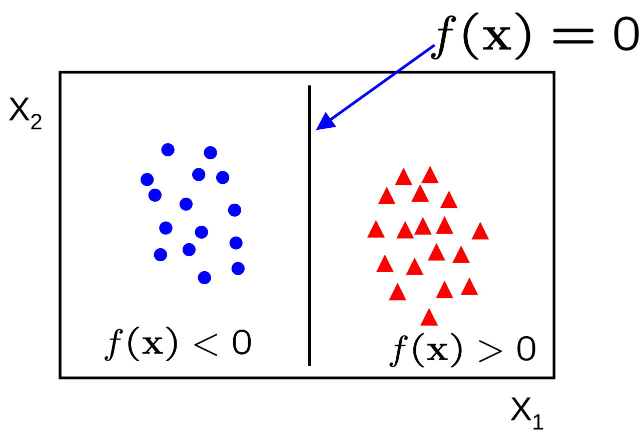
not linearly separable



# Linear classifiers

A linear classifier has the form

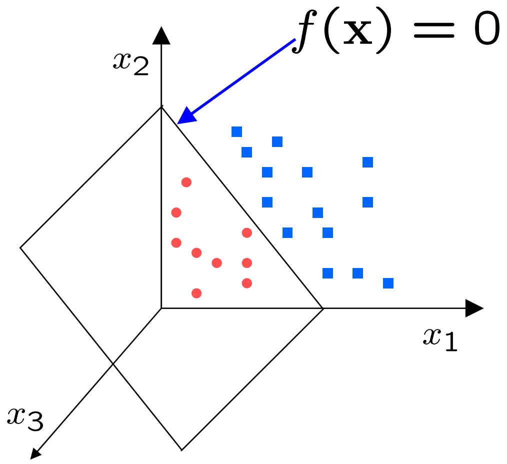
$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$




- in 2D the discriminant is a line
- $\mathbf{w}$ is the normal to the plane, and b the bias
- $\mathbf{w}$ is known as the weight vector

# Linear classifiers

A linear classifier has the form




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

- in 3D the discriminant is a plane, and in nD it is a hyperplane

For a K-NN classifier it was necessary to `carry' the training data

For a linear classifier, the training data is used to learn **w** and then discarded

Only **w** is needed for classifying new data

# Reminder: The Perceptron Classifier

Given linearly separable data $\mathbf{x}_i$ labelled into two categories $y_i = \{-1,1\}$ , find a weight vector **w** such that the discriminant function

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

separates the categories for i = 1, .., N

- how can we find this separating hyperplane ?

## The Perceptron Algorithm

Write classifier as $f(\mathbf{x}_i) = \tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}_i + w_0 = \mathbf{w}^\top \mathbf{x}_i$

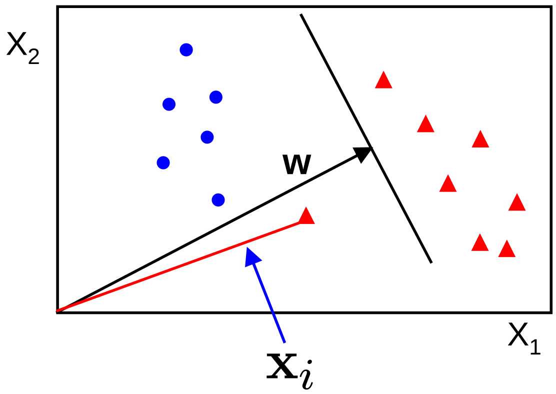
where $\mathbf{w} = (\tilde{\mathbf{w}}, w_0), \mathbf{x}_i = (\tilde{\mathbf{x}}_i, 1)$

- Initialize **w** = 0
- Cycle though the data points { $\mathbf{x}_i$, $y_i$ }
  - if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha\, \text{sign}(f(\mathbf{x}_i))\, \mathbf{x}_i$
- Until all the data is correctly classified

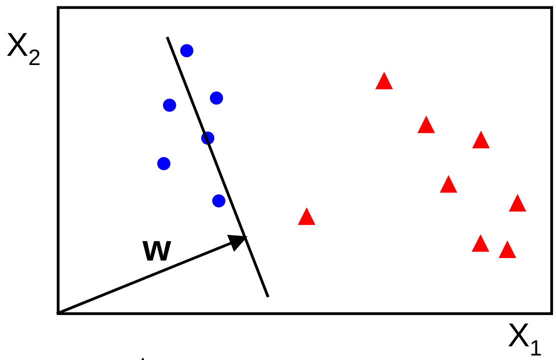
## For example in 2D

- Initialize $\mathbf{w} = 0$
- Cycle though the data points $\{ \mathbf{x}_i, y_i \}$
  - if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha \, \text{sign}(f(\mathbf{x}_i)) \, \mathbf{x}_i$
- Until all the data is correctly classified

before update



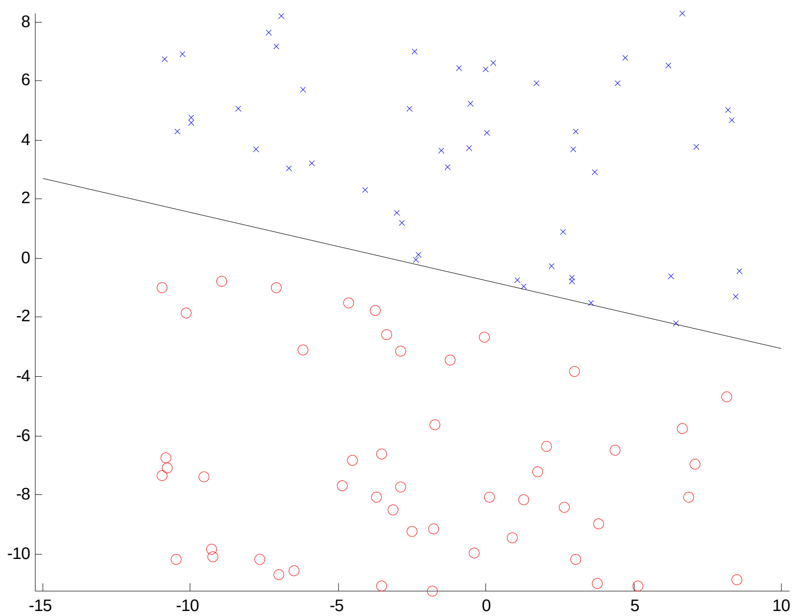

after update




$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \, \mathbf{x}_i$$

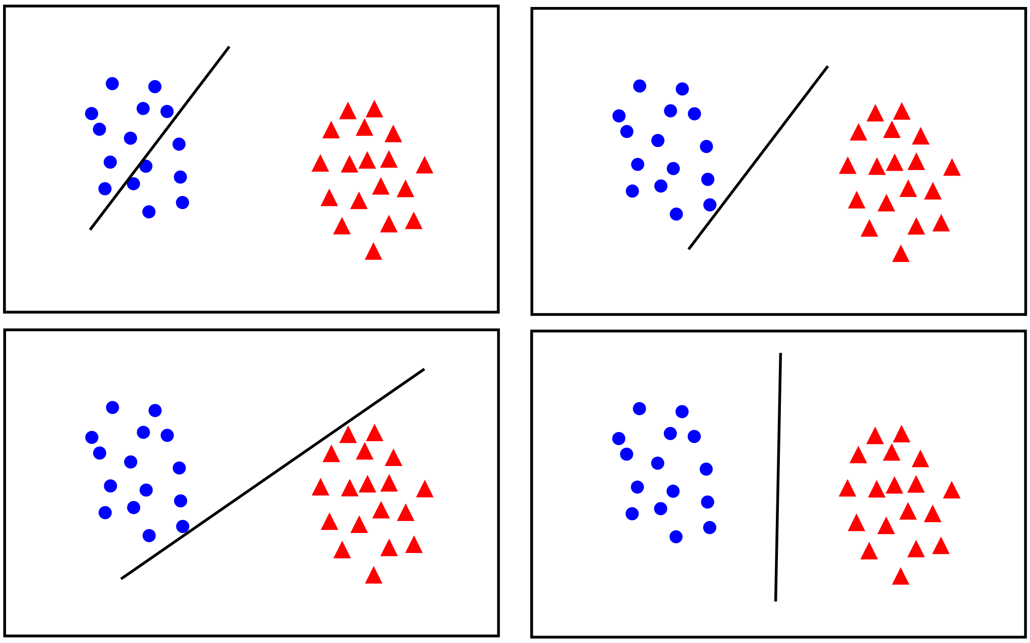
NB after convergence $\mathbf{w} = \sum_i^N \alpha_i \mathbf{x}_i$

## Perceptron example



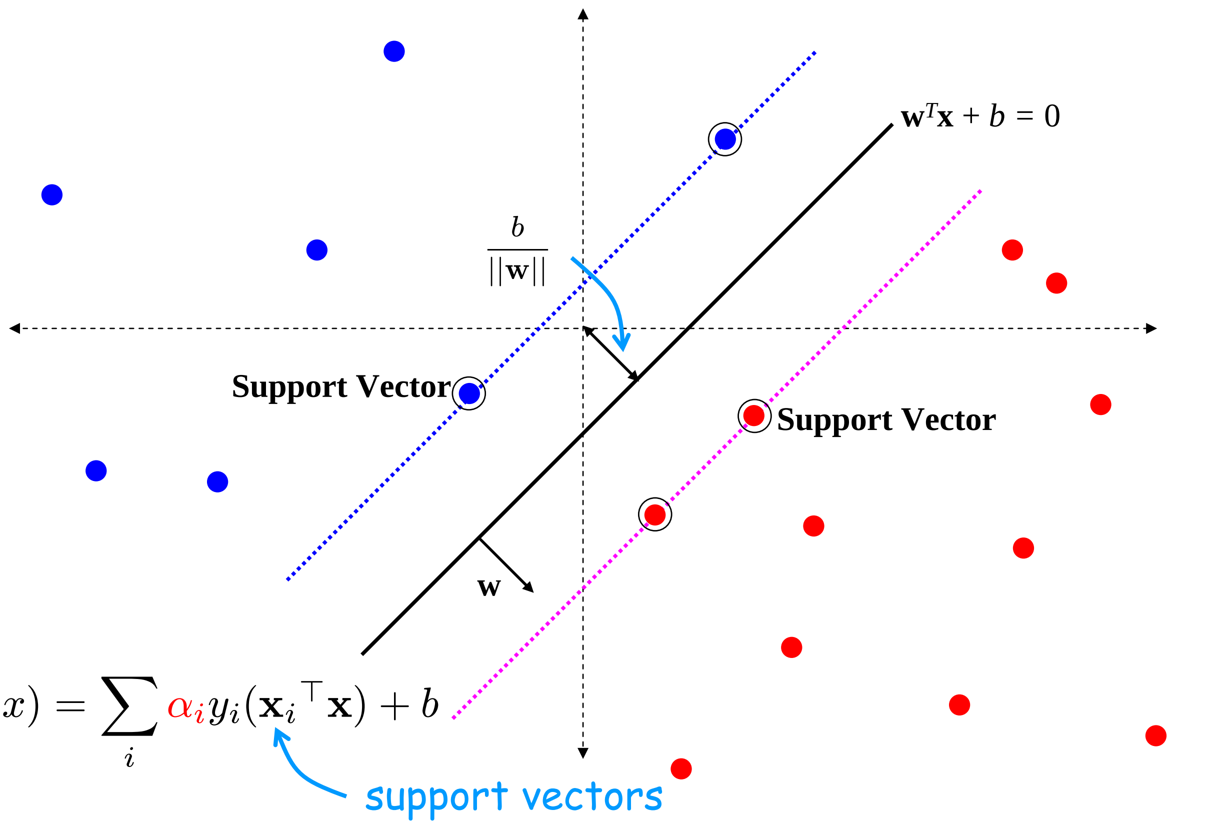

- if the data is linearly separable, then the algorithm will converge
- convergence can be slow …
- separating line close to training data
- we would prefer a larger margin for generalization

# What is the best w?



- maximum margin solution: most stable under perturbations of the inputs

# Support Vector Machine




$$f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$$

## SVM – sketch derivation

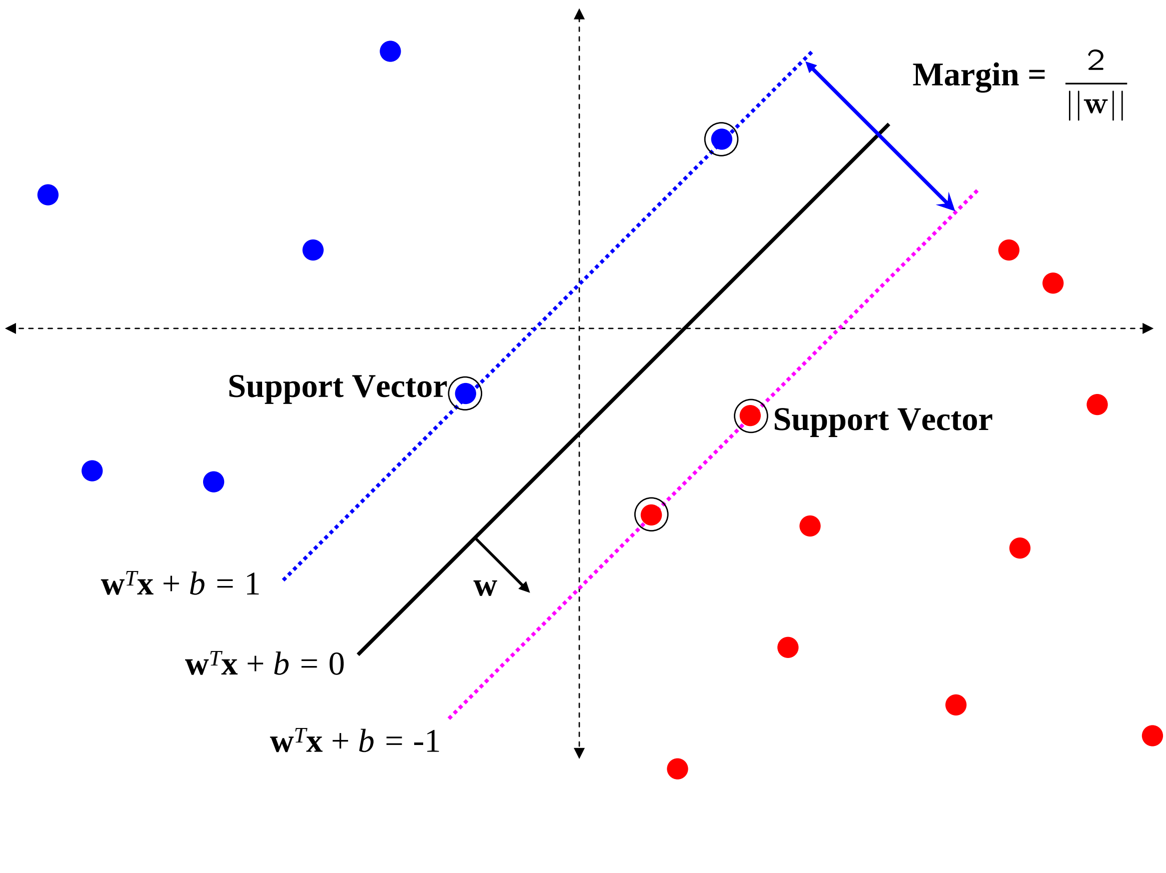
- Since $\mathbf{w}^\top \mathbf{x} + b = 0$ and $c(\mathbf{w}^\top \mathbf{x} + b) = 0$ define the same plane, we have the freedom to choose the normalization of $\mathbf{w}$

- Choose normalization such that $\mathbf{w}^\top \mathbf{x}_+ + b = +1$ and $\mathbf{w}^\top \mathbf{x}_- + b = -1$ for the positive and negative support vectors respectively

- Then the margin is given by

$$\frac{\mathbf{w}^\top \left(\mathbf{x}_+ - \mathbf{x}_-\right)}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$

# Support Vector Machine



## SVM – Optimization

- Learning the SVM can be formulated as an optimization:

$$\max_{\mathbf{w}} \frac{2}{||\mathbf{w}||} \text{ subject to } \mathbf{w}^\top \mathbf{x}_i + b \begin{array}{ll} \geq 1 & \text{if } y_i = +1 \\ \leq -1 & \text{if } y_i = -1 \end{array} \text{ for } i = 1 \ldots N$$

- Or equivalently

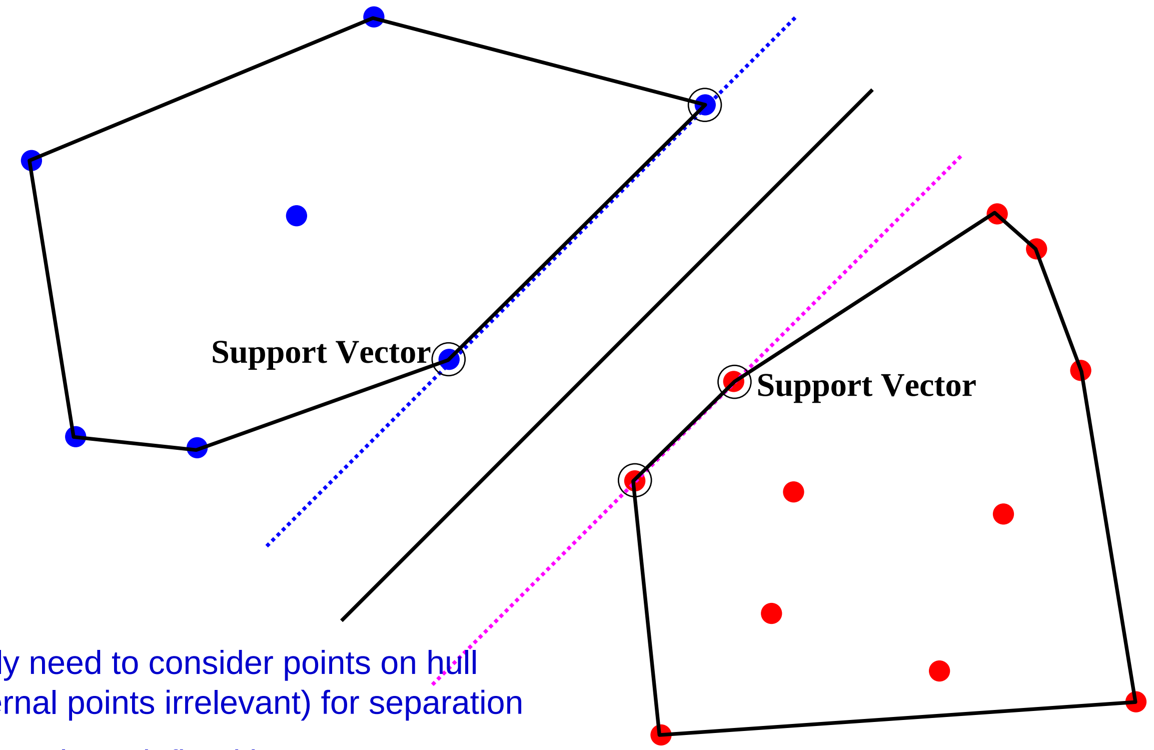
$$\min_{\mathbf{w}} ||\mathbf{w}||^2 \text{ subject to } y_i \left( \mathbf{w}^\top \mathbf{x}_i + b \right) \geq 1 \text{ for } i = 1 \ldots N$$

- This is a quadratic optimization problem subject to linear constraints and there is a unique minimum

## SVM – Geometric Algorithm

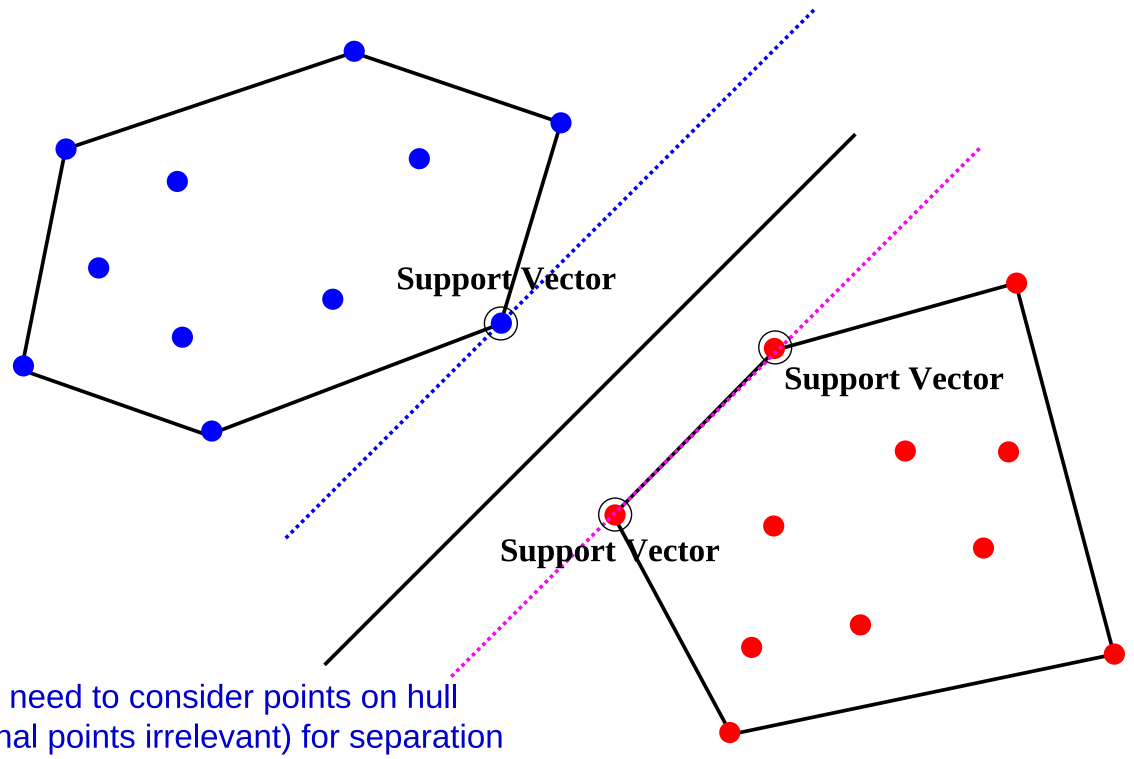
- Compute the convex hull of the positive points, and the convex hull of the negative points
- For each pair of points, one on positive hull and the other on the negative hull, compute the margin
- Choose the largest margin

# Geometric SVM Ex I



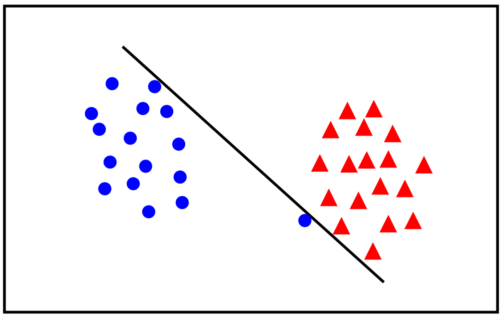

- only need to consider points on hull (internal points irrelevant) for separation
- hyperplane defined by support vectors

# Geometric SVM Ex II

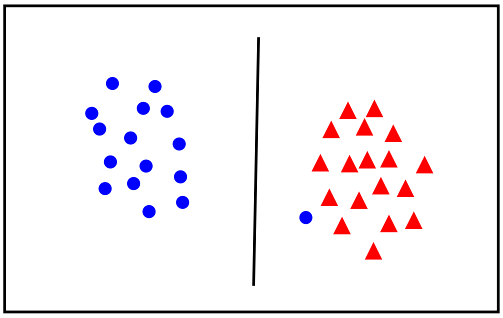



- only need to consider points on hull (internal points irrelevant) for separation
- hyperplane defined by support vectors

## Linear separability again: What is the best w?



• the points can be linearly separated but there is a very narrow margin



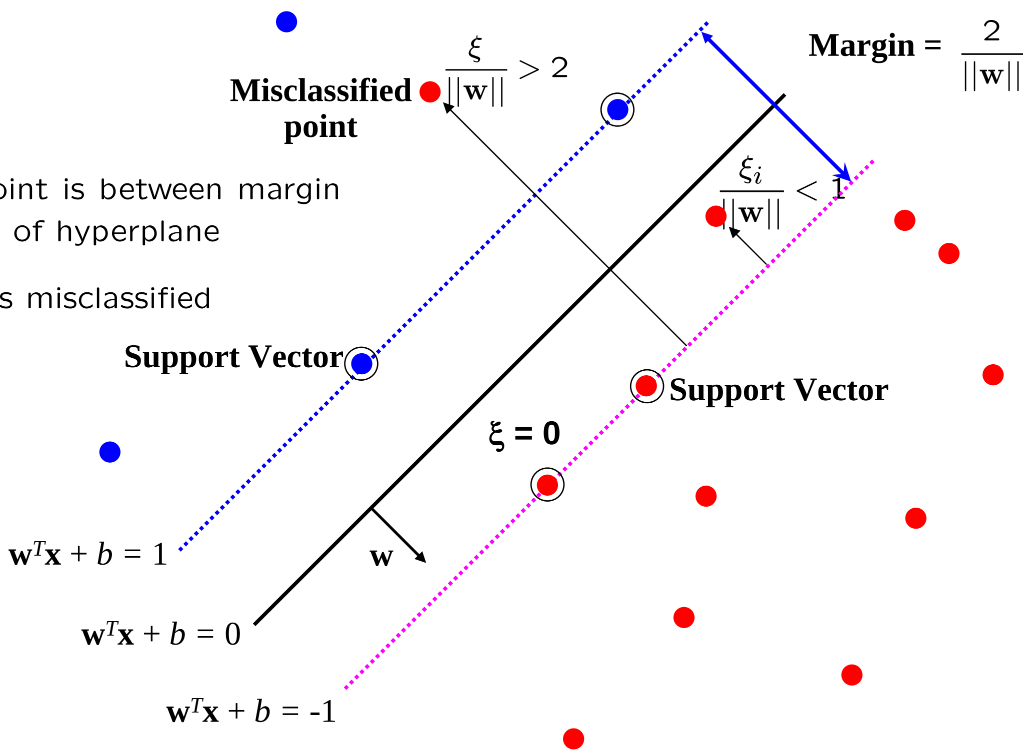
• but possibly the large margin solution is better, even though one constraint is violated

In general there is a trade off between the margin and the number of mistakes on the training data

## Introduce "slack" variables for misclassified points

$\xi_i \geq 0$

- for $0 \leq \xi \leq 1$ point is between margin and correct side of hyperplane
- for $\xi > 1$ point is misclassified



# "Soft" margin solution

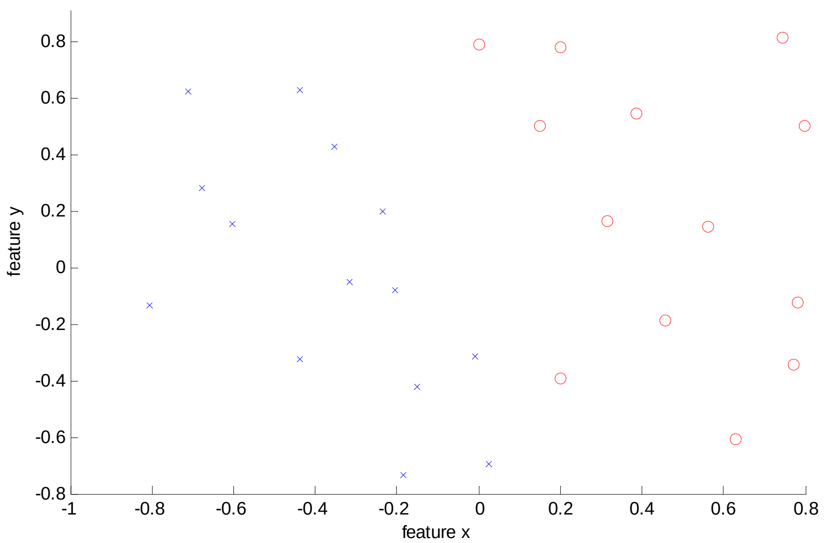
The optimization problem becomes

$$\min_{\mathbf{w}\in\mathbb{R}^d, \xi_i \in \mathbb{R}^+} ||\mathbf{w}||^2 + C \sum_i^N \xi_i$$

subject to

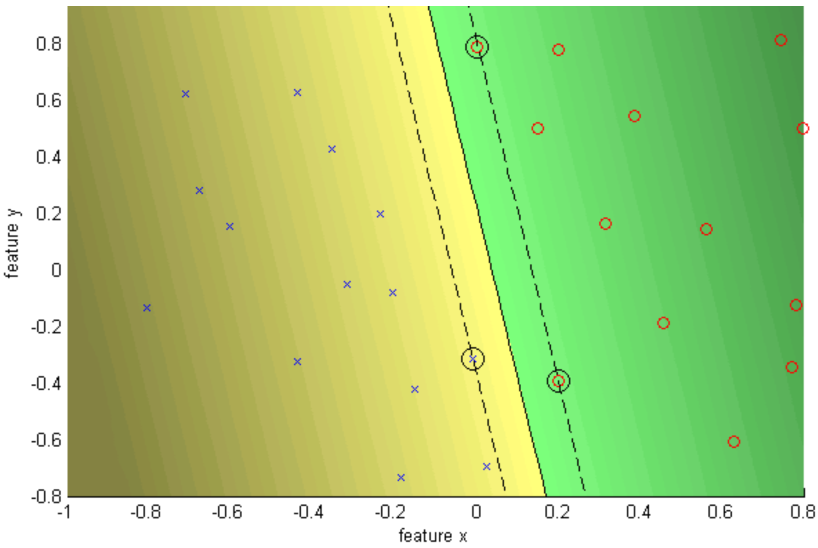
$$y_i \left( \mathbf{w}^\top \mathbf{x}_i + b \right) \geq 1 - \xi_i \text{ for } i = 1 \ldots N$$

- Every constraint can be satisfied if $\xi_i$ is sufficiently large
- $C$ is a regularization parameter:
  - small $C$ allows constraints to be easily ignored $\rightarrow$ large margin
  - large $C$ makes constraints hard to ignore $\rightarrow$ narrow margin
  - $C = \infty$ enforces all constraints: hard margin
- This is still a quadratic optimization problem and there is a unique minimum. Note, there is only one parameter, $C$.




- data is linearly separable
- but only with a narrow margin

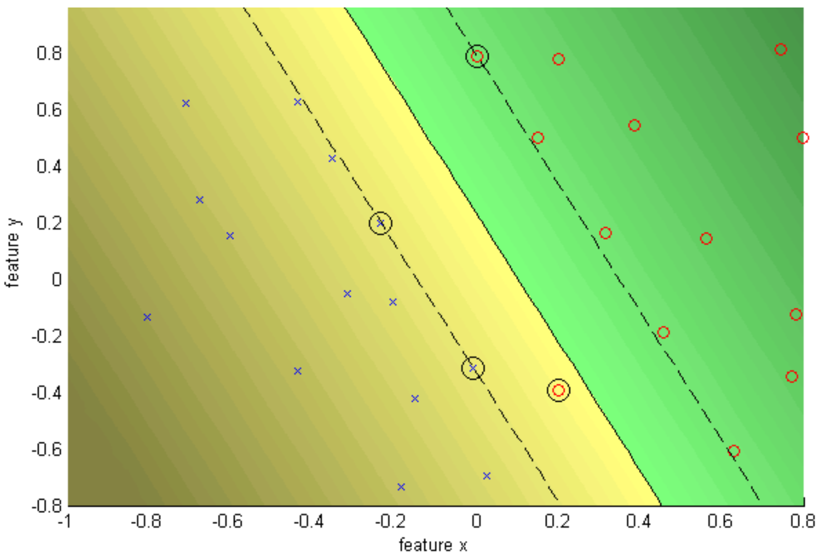
# C = Infinity    hard margin




Comment Window

SVM (L1) by Sequential Minimal Optimizer
Kernel: linear (-), C: Inf
Kernel evaluations: 971
Number of Support Vectors: 3
Margin: 0.0966
Training error: 0.00%

# C = 10    soft margin




Comment Window

SVM (L1) by Sequential Minimal Optimizer
Kernel: linear (-), C: 10.0000
Kernel evaluations: 2645
Number of Support Vectors: 4
Margin: 0.2265
Training error: 3.70%

# Application: Pedestrian detection in Computer Vision

Objective: detect (localize) standing humans in an image

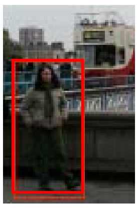
- cf face detection with a sliding window classifier

- reduces object detection to binary classification
- does an image window contain a person or not?

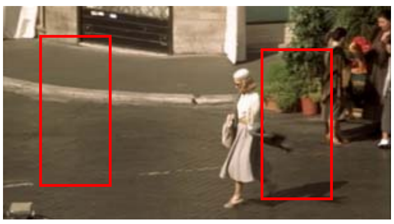
Method: the HOG detector

# Training data and features

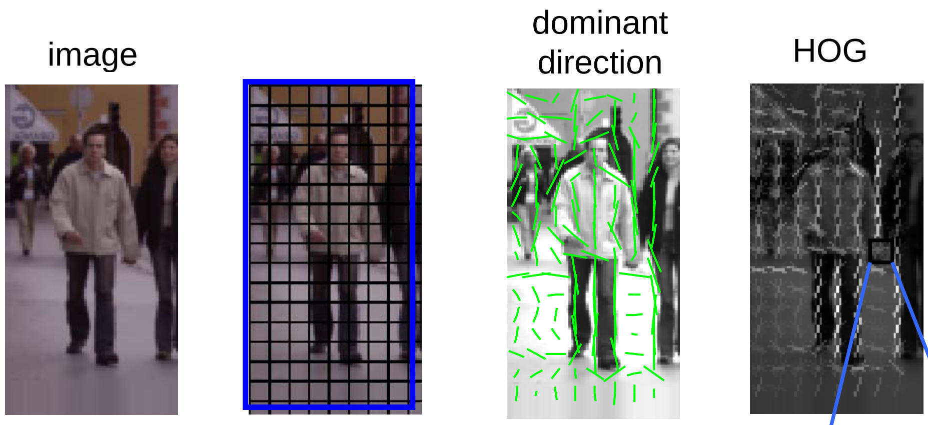
- Positive data – 1208 positive window examples



- Negative data – 1218 negative window examples (initially)

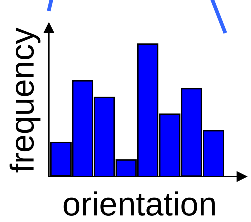


# Feature: histogram of oriented gradients (HOG)

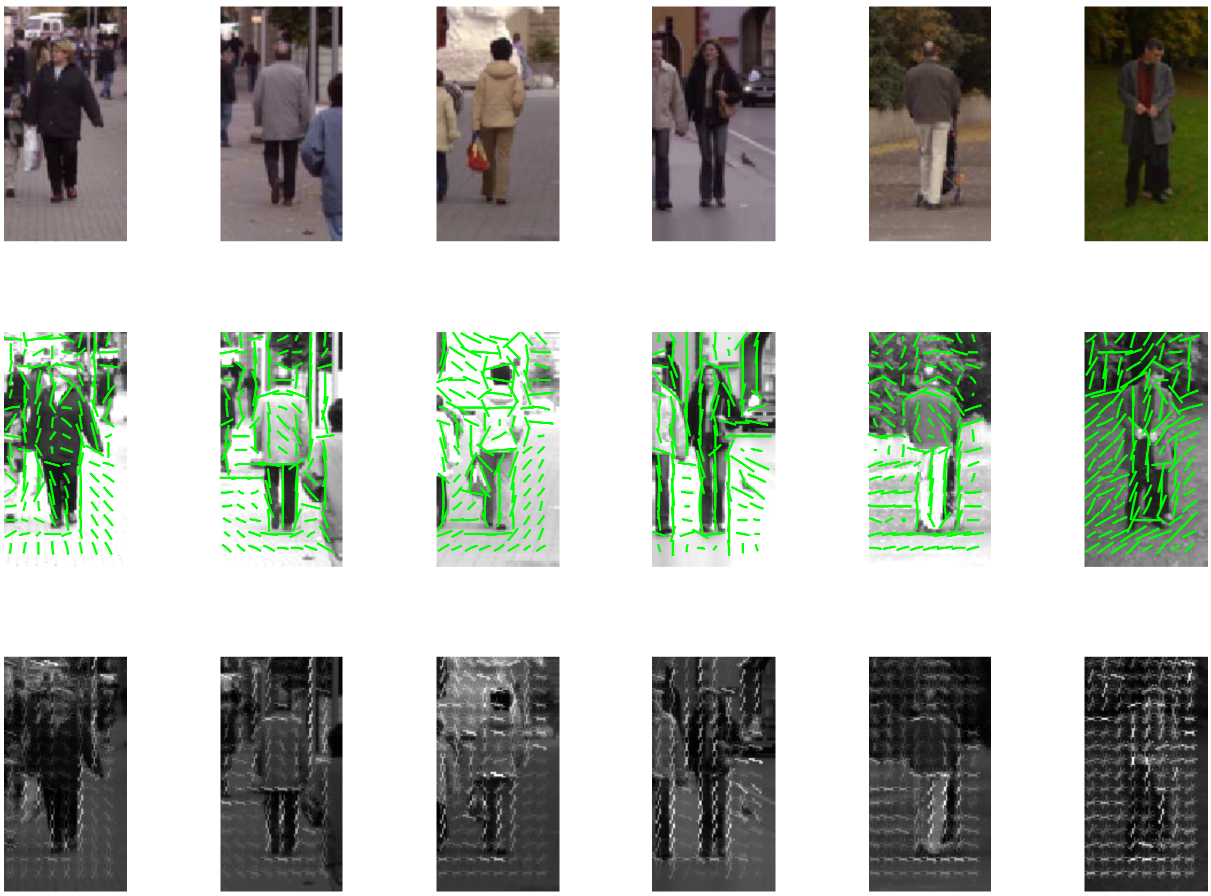



• tile window into 8 x 8 pixel cells

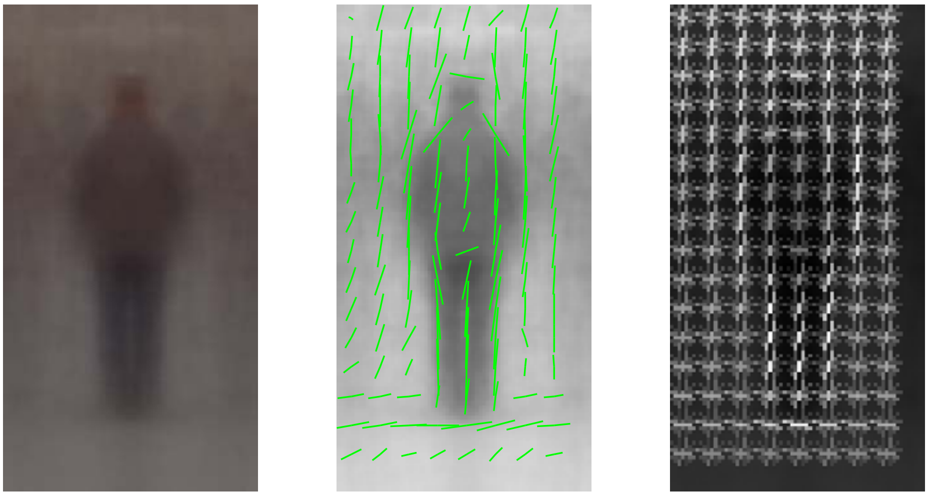
• each cell represented by HOG




Feature vector dimension = 16 x 8 (for tiling) x 8 (orientations) = 1024



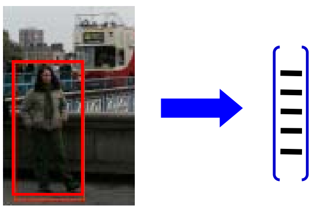
## Averaged examples



# Algorithm

## Training (Learning)

- Represent each example window by a HOG feature vector



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

- Train a SVM classifier

Testing (Detection)

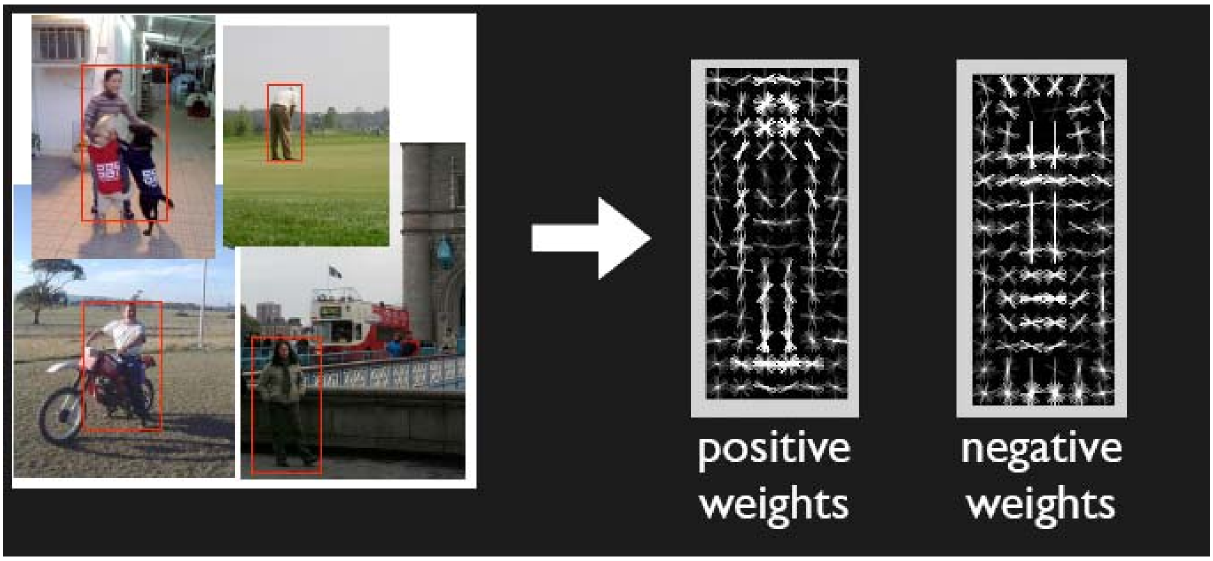
- Sliding window classifier

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

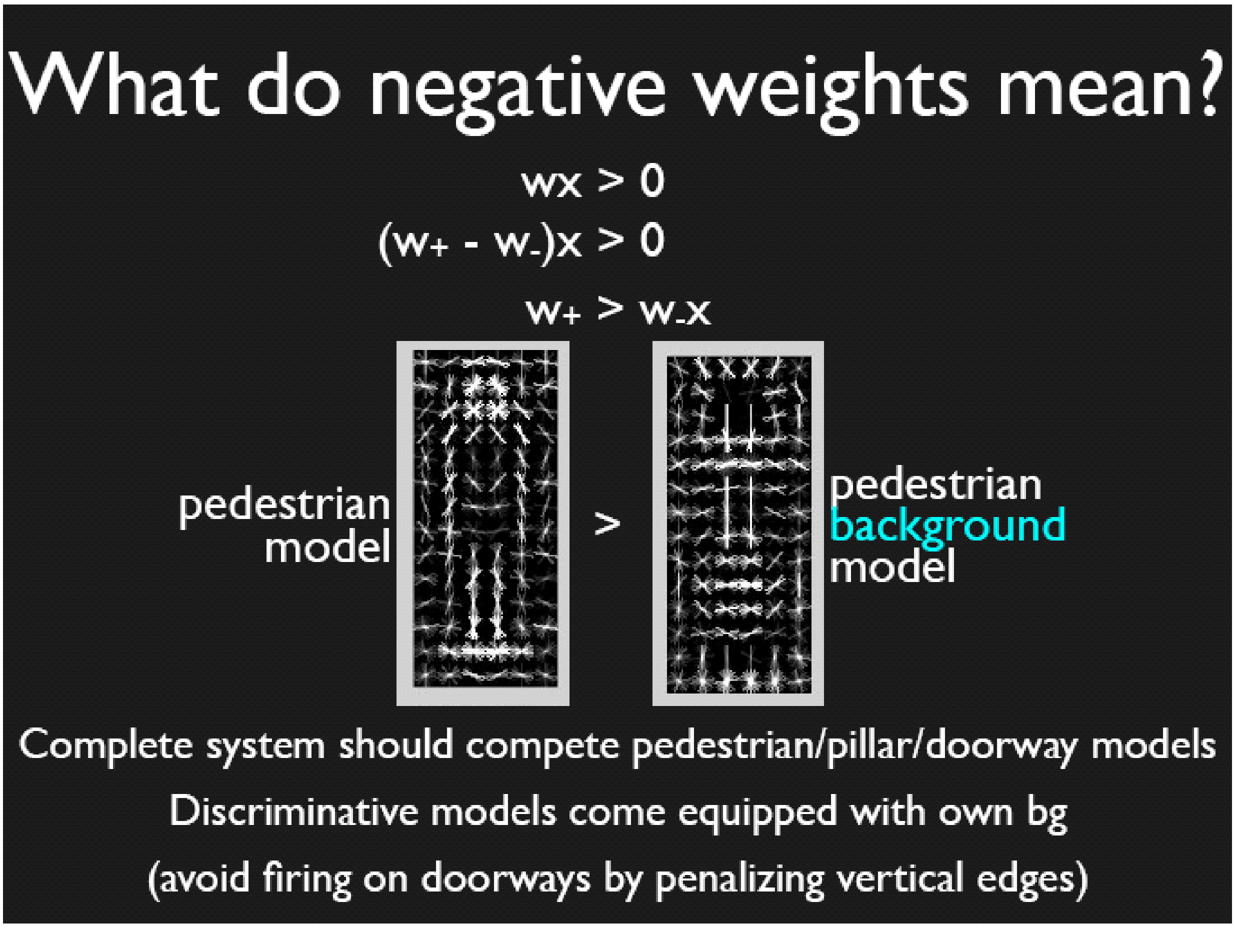

Dalal and Triggs, CVPR 2005

## Learned model

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




Slide from Deva Ramanan




Slide from Deva Ramanan

# Optimization

Learning an SVM has been formulated as a constrained optimization problem over $\mathbf{w}$ and $\xi$

$$\min_{\mathbf{w}\in\mathbb{R}^d, \xi_i \in \mathbb{R}^+} ||\mathbf{w}||^2 + C\sum_i^N \xi_i \text{ subject to } y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1 \dots N$$

The constraint $y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i$, can be written more concisely as

$$y_i f(\mathbf{x}_i) \geq 1 - \xi_i$$

which is equivalent to

$$\xi_i = \max\left(0, 1 - y_i f(\mathbf{x}_i)\right)$$

Hence the learning problem is equivalent to the unconstrained optimization problem

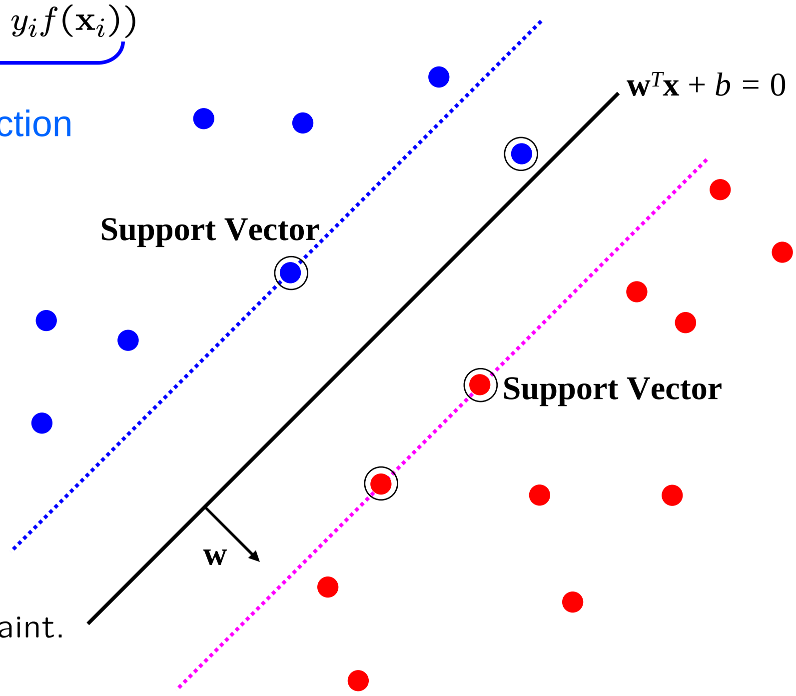
$$\min_{\mathbf{w}\in\mathbb{R}^d} \underbrace{||\mathbf{w}||^2}_{\text{regularization}} + C\sum_i^N \underbrace{\max\left(0, 1 - y_i f(\mathbf{x}_i)\right)}_{\text{loss function}}$$

# Loss function

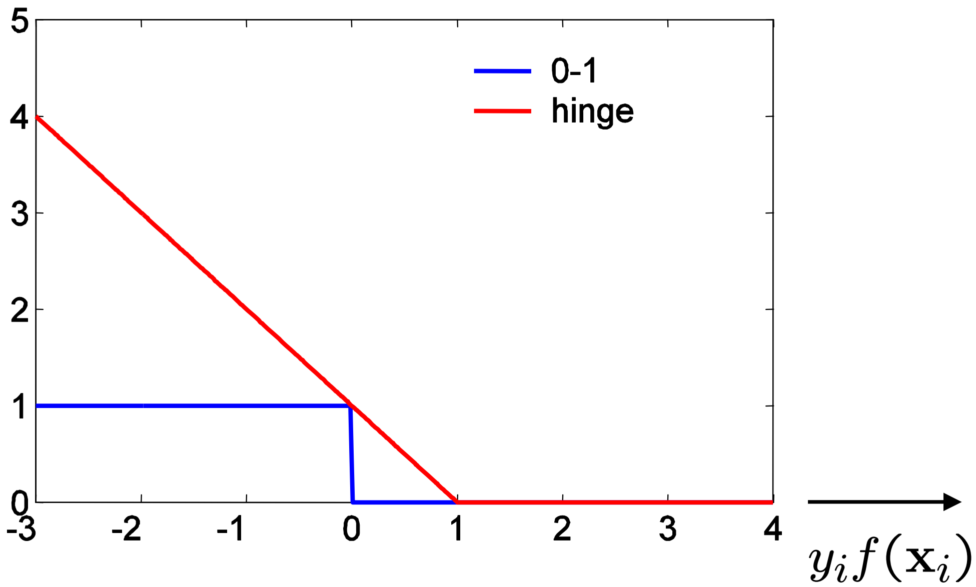
$$\min_{\mathbf{w}\in\mathbb{R}^d} ||\mathbf{w}||^2 + C\sum_i^N \underbrace{\max\left(0, 1 - y_i f(\mathbf{x}_i)\right)}_{\text{loss function}}$$

Points are in three categories:

1. $y_i f(x_i) > 1$
   Point is outside margin.
   No contribution to loss
2. $y_i f(x_i) = 1$
   Point is on margin.
   No contribution to loss.
   As in hard margin case.
3. $y_i f(x_i) < 1$
   Point violates margin constraint.
   Contributes to loss




# Loss functions




- SVM uses "hinge" loss $\max\left(0, 1 - y_i f(\mathbf{x}_i)\right)$
- an approximation to the 0-1 loss

# Background reading and more …

- Next lecture – see that the SVM can be expressed as a sum over the support vectors:

$$f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$$

support vectors

- On web page:
  http://www.robots.ox.ac.uk/~az/lectures/ml

- links to SVM tutorials and video lectures

- MATLAB SVM demo