Support Vector Machines

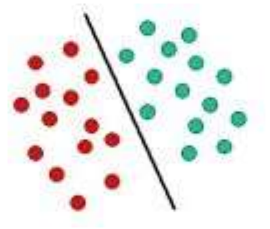
1856 १८५६
उत्तिष्ठत जाग्रत प्राप्य वरान निबोधत
INDIAN INSTITUTE OF ENGINEERING SCIENCE AND TECHNOLOGY, SHIBPUR
भारतीय अभियांत्रिकी विज्ञान एवं प्रौद्योगिकी संस्थान, शिवपुर

Support Vector Machines (**SVM**)

- SVM were introduced by Vladimir Vapnik (Vapnik, 1995).
- The main objective in SVM is to find the hyperplane which separates the d-dimensional data points perfectly into two classes.
- However, since example data is often not linearly separable, SVM's introduce the notion of a “kernel induced feature space" which casts the data points (input space) into a higher dimensional feature space where the data is separable.
- SVM's higher-dimensional space doesn't need to be dealt with directly which eliminates overfitting.

Support Vector Machines - Linear classifier

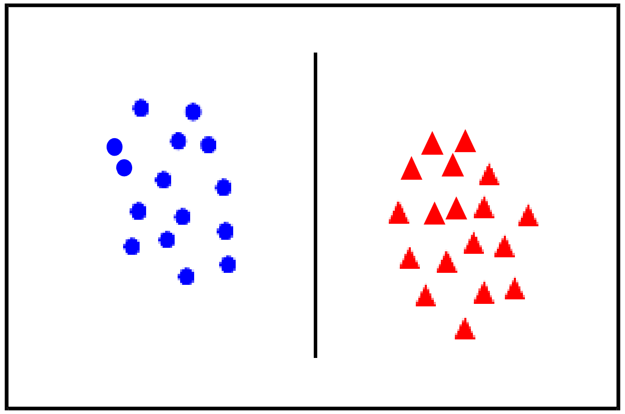
- Classification tasksare based on drawing separating lines to distinguish between objects of different class labels are known as hyperplane classifiers.
- A decision plane is one that separates between a set of objects having different class labels.

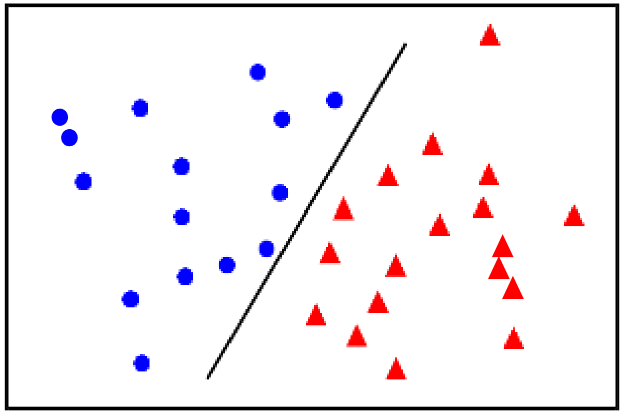


- Any new object falling to the right is labeled, i.e., classified, as GREEN (or classified as RED should it fall to the left of the separating line).
- The objects closest to the hyperplane is called support vectors

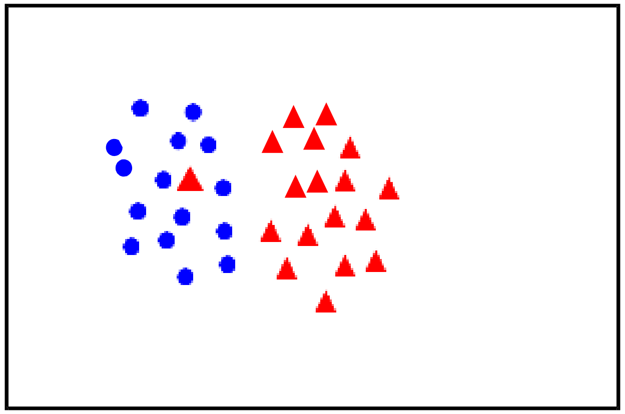
Linear separability

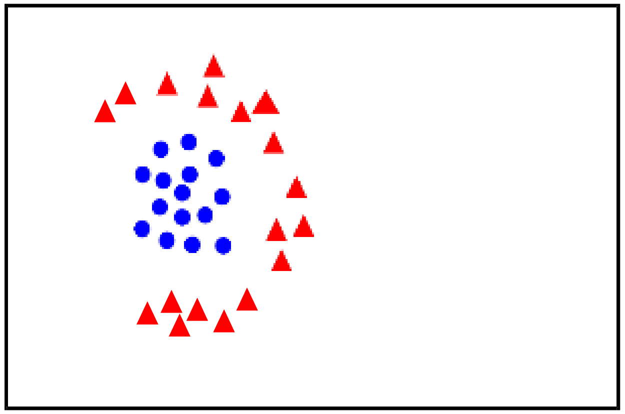
linearly separabl e



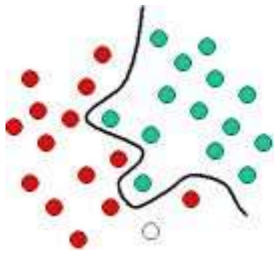


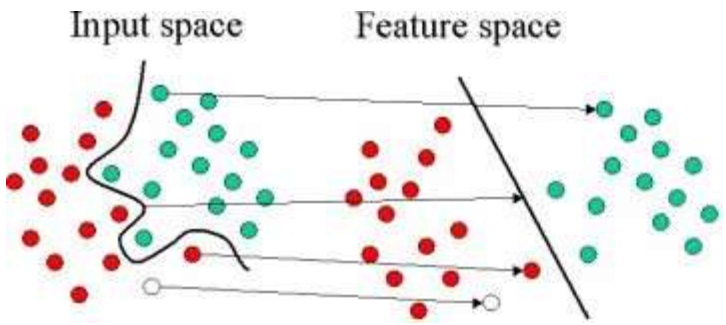
not linearly separabl e





Input Space to Feature Space





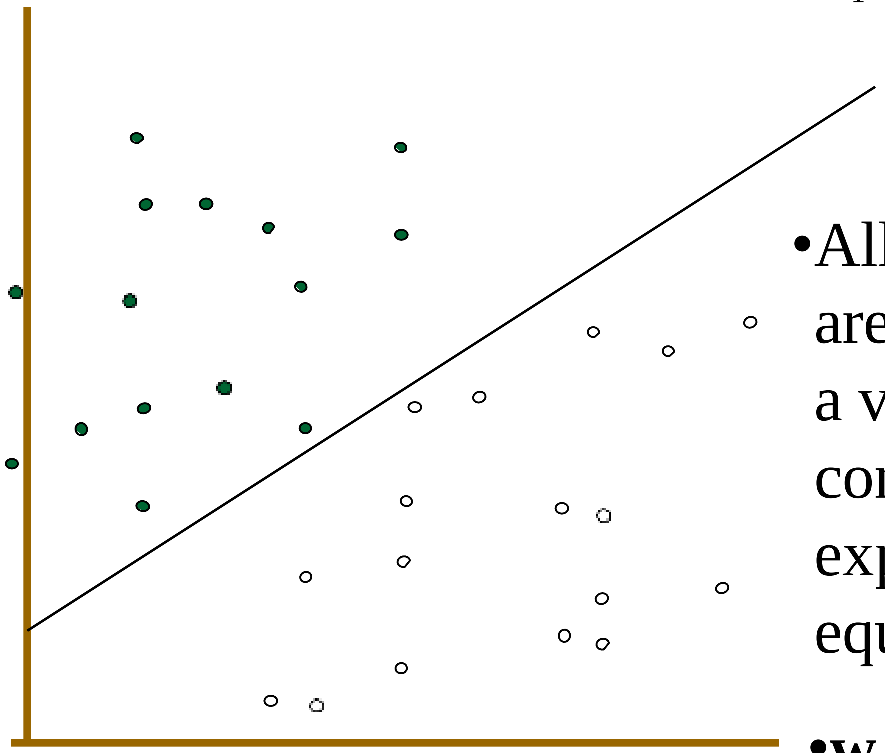
- The original objects are transformed, using a set of mathematical functions, known as kernels.
- Instead of constructing the complex curve, we find an optimal line that can separate the objects.

Linear Classifiers

We are given l training examples $\{x_i, y_i\}$; i = 1.. l , where each example has d inputs ($x_i \in \mathbf{R}^d$), and a class label with one of two values ($y_i \in \{-1, 1\}$.

● denotes +1

○ denotes -1



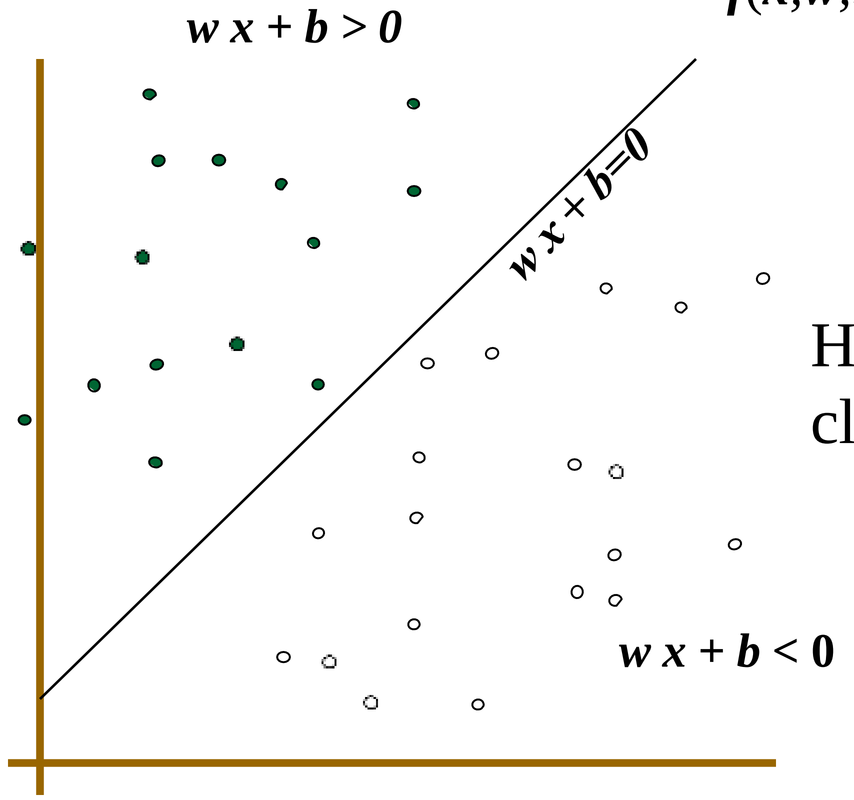
- All hyperplanes in $\mathbf{R}^d$ are parameterized by a vector (**w**) and a constant (b), expressed using the equation **w** . **x** + b = 0
- **w** is the vector orthogonal to the hyperplane

- Given such a hyperplane (**w**,b) that separates the data, using function $f(x) = \text{sign}(\mathbf{w}.\ \mathbf{x} + b)$

Linear Classifiers

- ● denotes +1
- ○ denotes -1

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



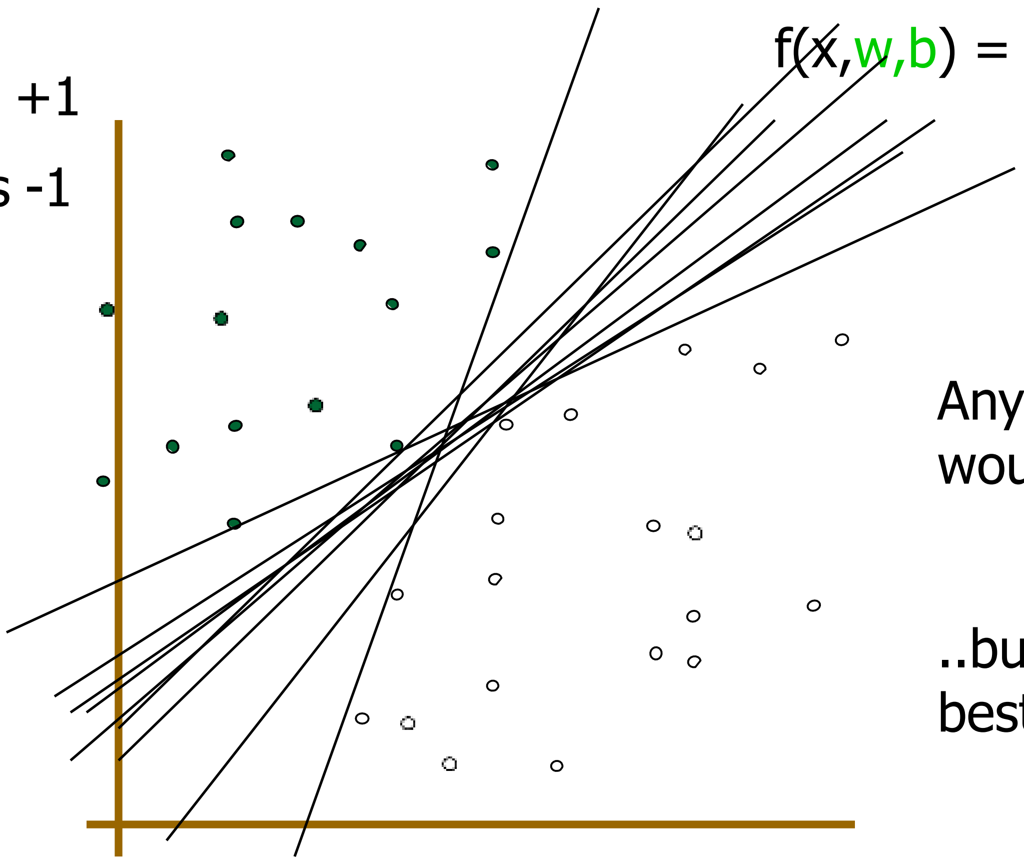
How would you classify this data?

Linear Classifiers

● denotes +1

○ denotes -1

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



Any of these would be fine..

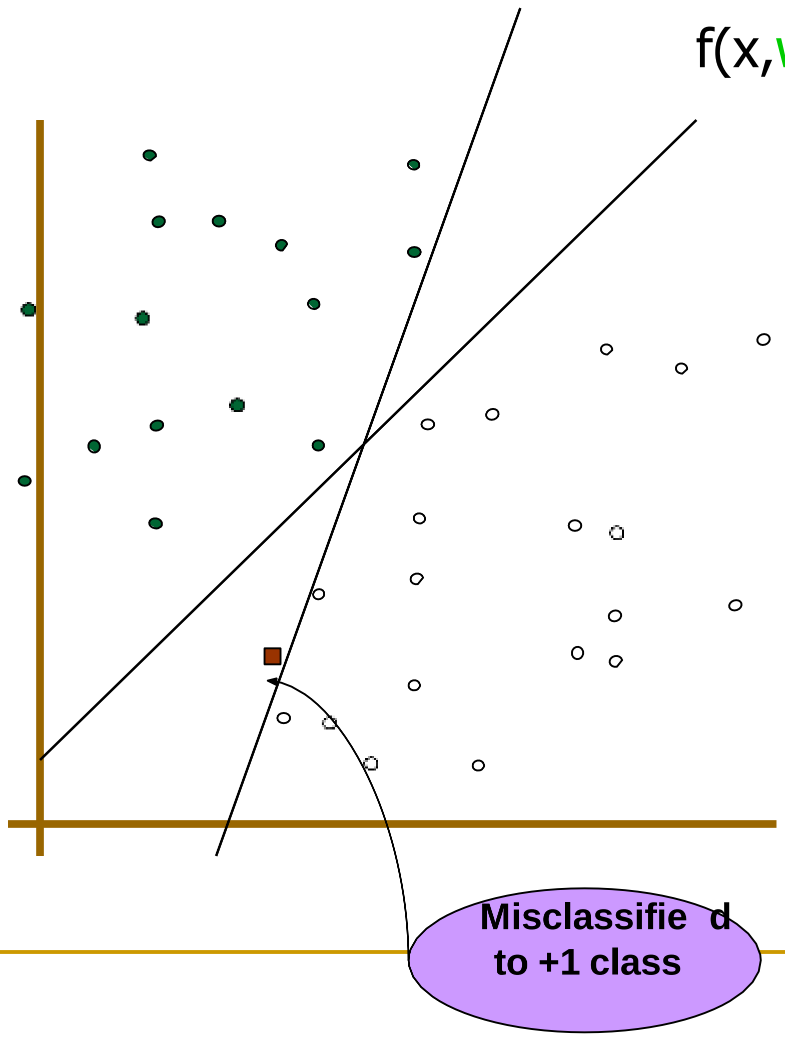
..but which is best?

Linear Classifiers

● denotes +1

○ denotes -1

$f(x,w,b) = \mathrm{sign}(w\ x + b)$



How would you classify this data?

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

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

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

• how can we find this separating hyperplane ?

The Perceptron Algorithm

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

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 }

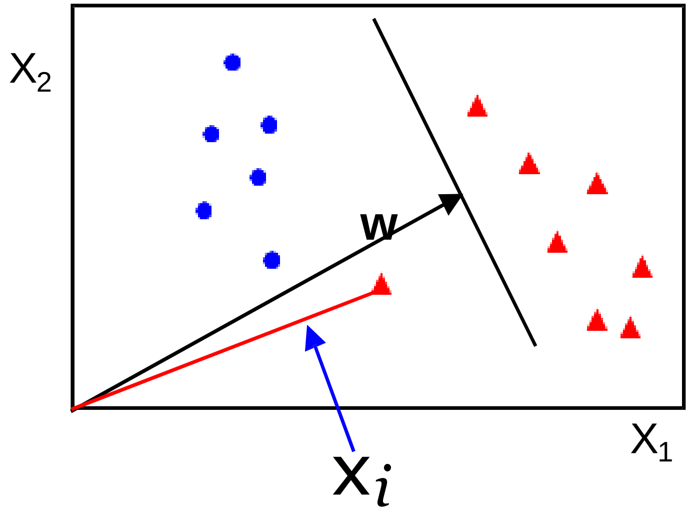
  • if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha \, \mathrm{sign}(f(\mathbf{x}_i)) \, \mathbf{x}_i$

• Until all the data is correctly classified

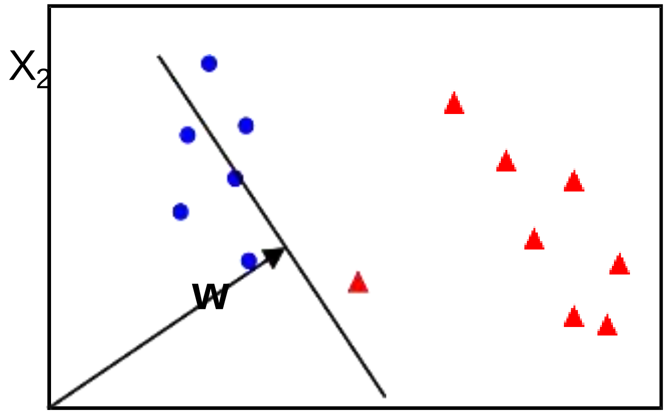
For example in 2D

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

before update
update

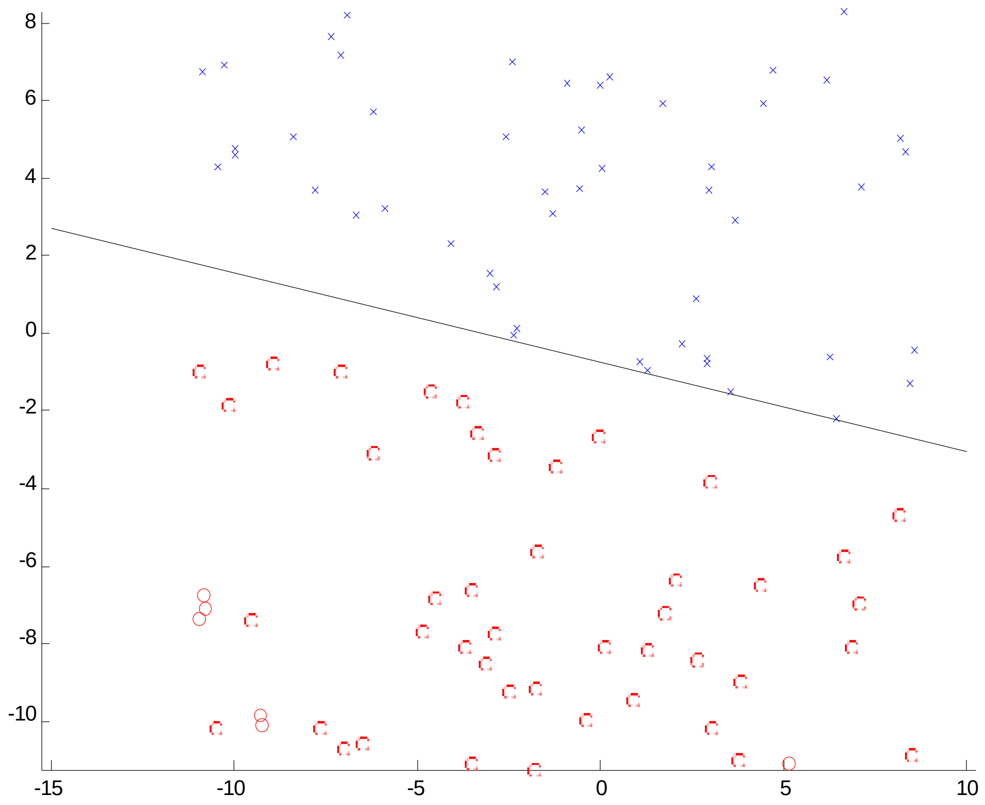


after



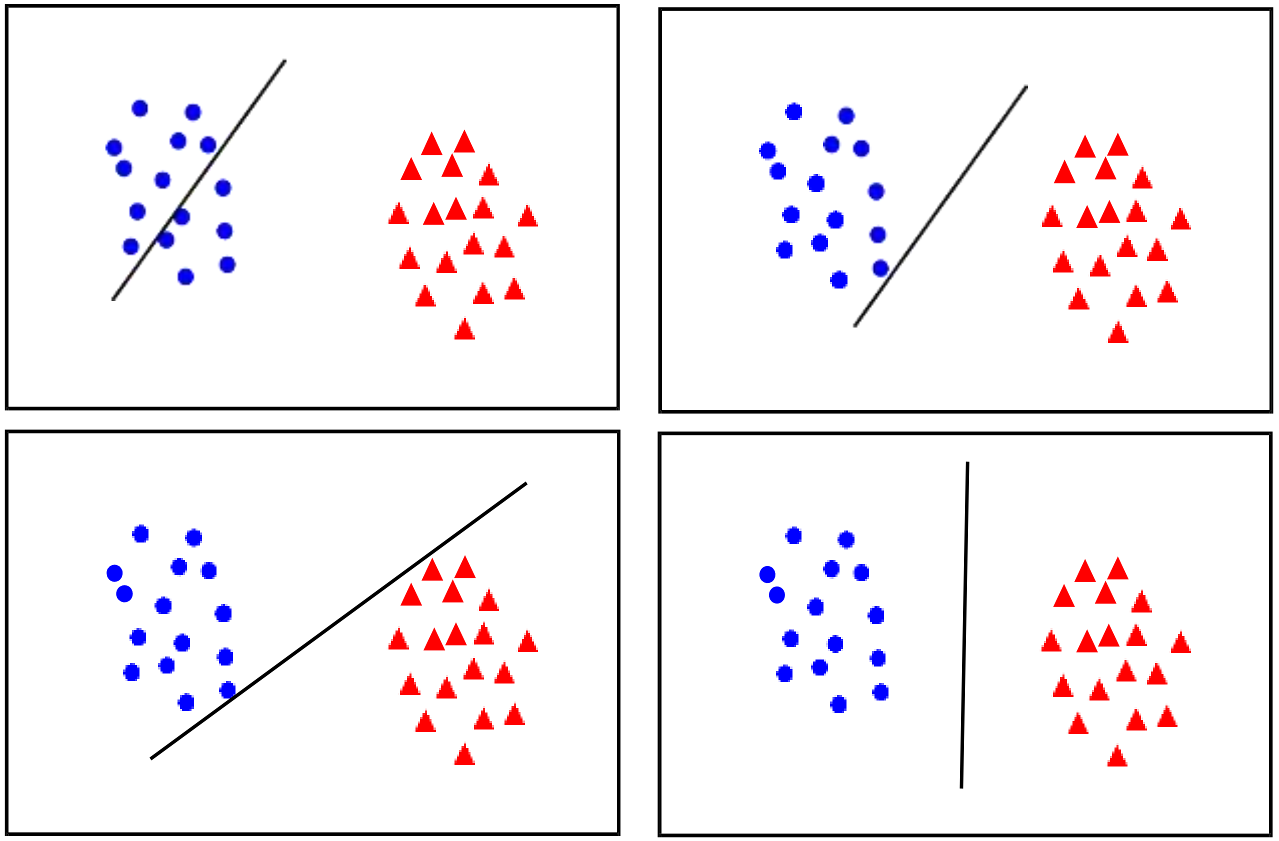
$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \, \mathbf{x}_i$$

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

Hyperplane Classifier

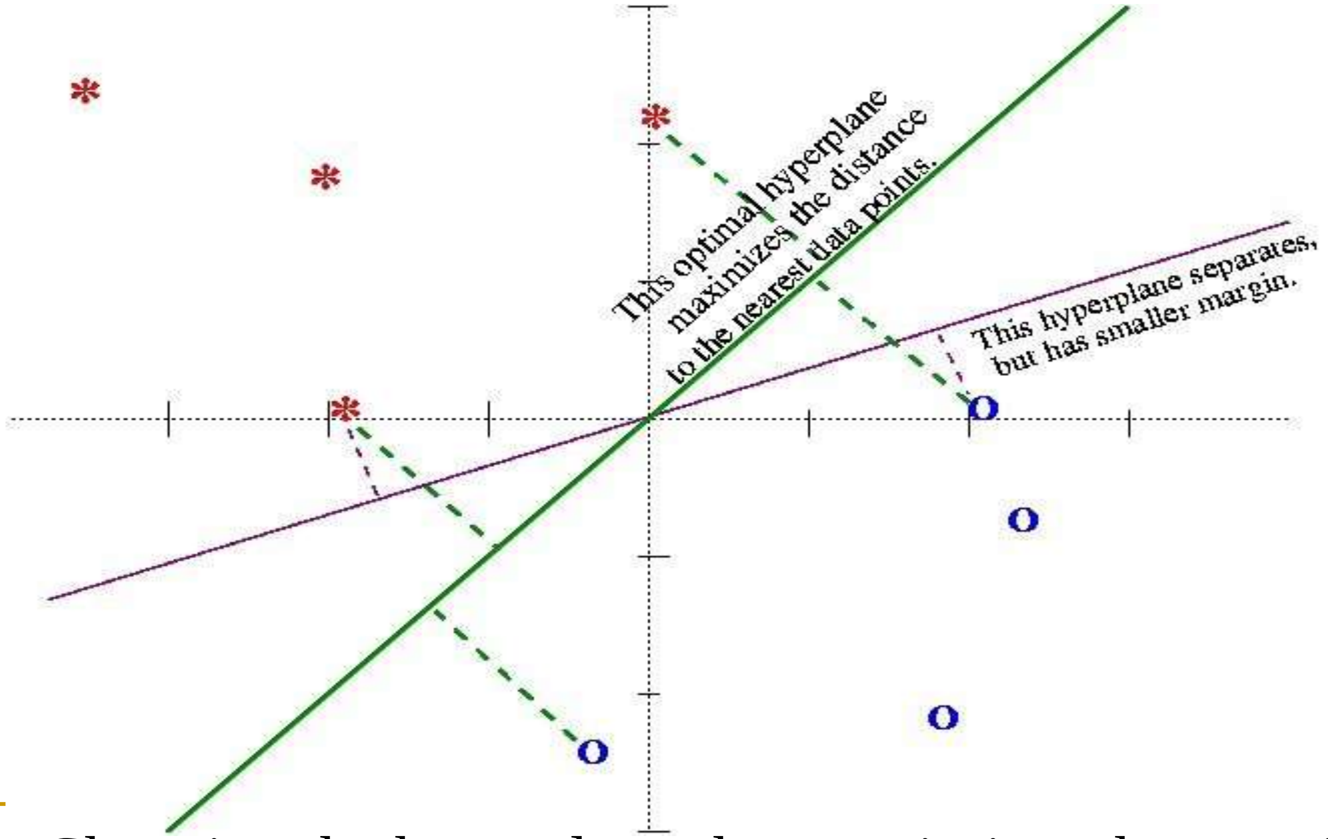
- A given hyperplane represented by ($\mathbf{w}$,b) is equally expressed by all pairs $\{\lambda\mathbf{w}, \lambda b\}$ for $\lambda \in R^+$.
- We define the hyperplane which separates the data from the hyperplane by a "distance" so that at least one example on both sides has a distance of exactly 1.
- That is, we consider those that satisfy:
- $\mathbf{w} \,.\, \mathbf{x_i} + b \geq 1$ when $y_i = +1$
- $\mathbf{w} \,.\, \mathbf{x_i} + b \leq 1$ when $y_i = -1$

$$y_i\,(\mathbf{w} \,.\, \mathbf{x_i} + b) \geq 1 \qquad \forall i$$

- To obtain the geometric distance from the hyperplane to a data point, we normalize by the magnitude of **w**.

- We want the hyperplane that maximizes the geometric distance to the closest data points.

$$d((\mathbf{w}, b), \mathbf{x}_i) = [y_i(\mathbf{w}.\mathbf{x}_i + b)] / \|\mathbf{w}\| \geq 1 / \|\mathbf{w}\|$$


This optimal hyperplane maximizes the distance to the nearest data points.
This hyperplane separates, but has smaller margin.

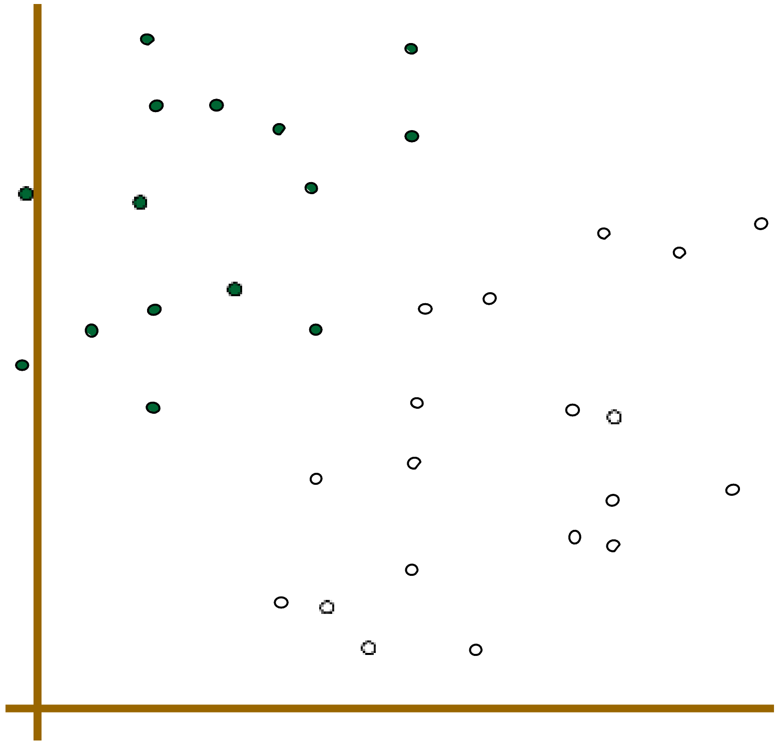
Choosing the hyperplane that maximizes the margin

Classifier Margin

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

● denotes +1

○ denotes -1



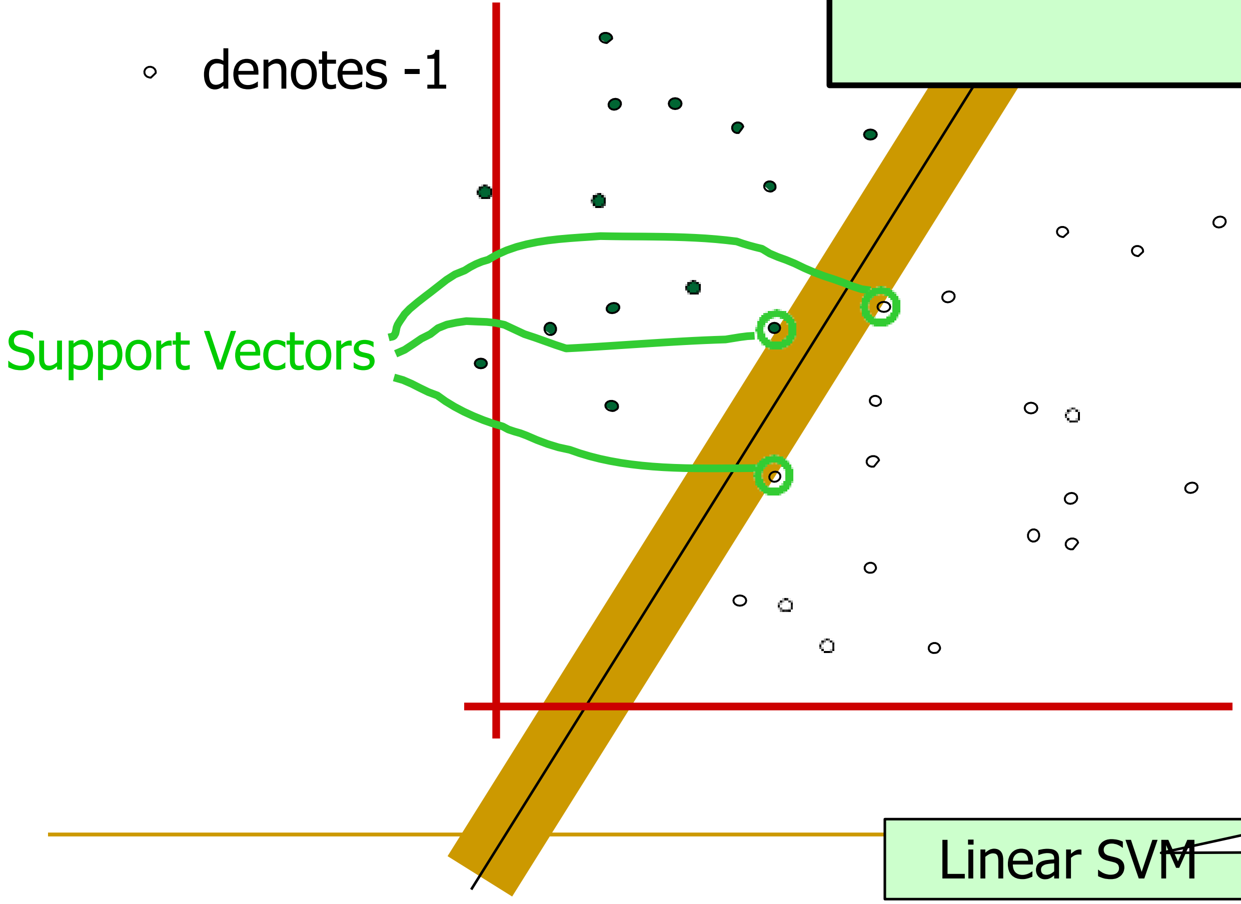
Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin

1. Maximizing the margin
2. support vectors are important

● denotes +1

○ denotes -1

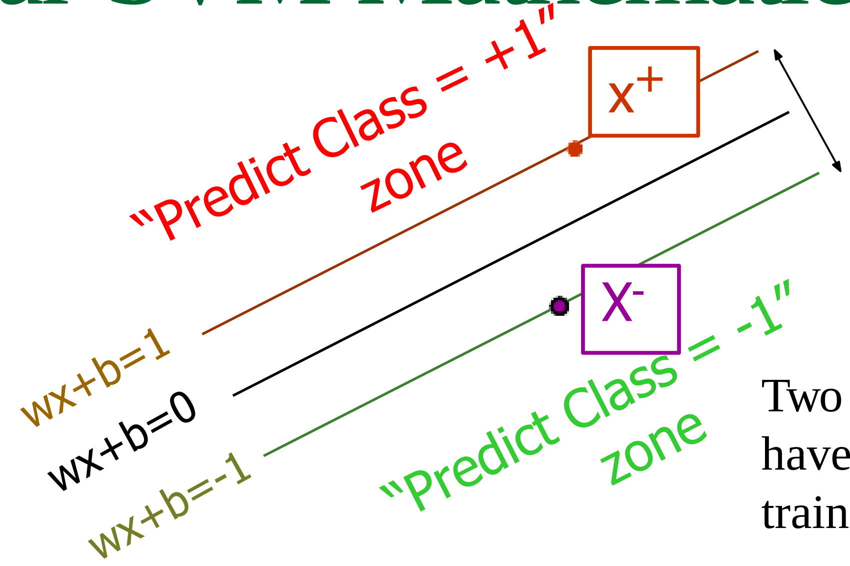


The maximum margin linear classifier

This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Linear SVM Mathematically



M=Margin Width

Two hyperplanes are parallel (they have the same normal) and that no training points fall between them.

What we know:

- $\boldsymbol{w} \, . \, \boldsymbol{x}^+ + b = +1$
- $\boldsymbol{w} \, . \, \boldsymbol{x}^- + b = -1$
- $\boldsymbol{w} \, . \, (\boldsymbol{x}^+ - \boldsymbol{x}^-) = 2$

$$M = \frac{(x^+ - x^-) \cdot w}{\|w\|} = \frac{2}{\|w\|}$$

Linear SVM Mathematically

- Goal: 1) Correctly classify all training data

$$\left.\begin{array}{ll} wx_i + b \geq 1 & \text{if } y_i = +1 \\ wx_i + b \leq 1 & \text{if } y_i = -1 \end{array}\right\}$$

$$y_i(wx_i + b) \geq 1 \quad \text{for all } i$$

2) Maximize the Margir $M = \frac{2}{\|w\|}$ or same as minimize $\frac{1}{2} w^t w$

- We can formulate a constrained optimization Problem and solve for **w** and b

- Minimize $\Phi(w) = \frac{1}{2} w^t w$

  subject to $\forall i \quad y_i(wx_i + b) \geq 1$

Lagrange Multipliers

- Consider a problem: $min_x\ f(x)$ subject to $h(x) = 0$
- We define the Lagrangian $L(x, \alpha) = f(x) - \alpha\, h(x)$
- α is called “Lagrange multiplier”
- Solve: $\min_x \max_\alpha L(x, \alpha)$ subject to $\alpha \geq 0$

Original Problem:

Find **w** and b such that

$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized;

and for all i $\{(\mathbf{x_i}, y_i)\}$: $y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

Construct the Lagrangian Function for optimization

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i [y^{(i)}(w^T x^{(i)} + b) - 1]$$ S. T. $\alpha_i \geq 0;\ \forall_i$

Our goal is to: $max_{\alpha \geq 0}\, min_{w,b}\, \mathcal{L}(w,b,\alpha)$ OR $min_{w,b}\, max_{\alpha \geq 0}\, \mathcal{L}(w,b,\alpha)$

$$\frac{\partial}{\partial w}\mathcal{L}(w,b,\alpha) = w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} = 0$$

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$$

The derivative with respect to b

$$\frac{\partial}{\partial b} C(w,b,\alpha) = \sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

Substituting we get: $C(w,b,\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} - b \sum_{i=1}^{m} \alpha_i y^{(i)}$

$$= \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)}$$

$\max_{\boldsymbol{\alpha}}$: $\sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)} . x^{(j)})$

Subject to $0 \le \alpha_i \le C \;\; for \; \forall i = 1 \ldots m$ and $\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$

$\boldsymbol{\alpha}$ is the vector of *m* non-negative Lagrange multipliers to be determined, and C is a constant

Optimal hyperplane : $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i x_i$

- The vector **w** is just a linear combination of the training examples.

$$C(w,b,\alpha) = \frac{1}{2}\mathbf{w}.\mathbf{w} - \sum_i \alpha_i \left[\left(\mathbf{w}.\mathbf{x}_i + b\right) y_i - 1\right]$$

$$\alpha_i \geq 0, \ \forall_i$$

$$y_i (\vec{w} \cdot \vec{x}_i + b) = 1 \quad \text{(1)}$$

$$y_i y_i (\vec{w} \cdot \vec{x}_i + b) = y_i \quad \text{(2)}$$

$$(\vec{w} \cdot \vec{x}_i + b) = y_i \quad \text{(3)}$$

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

$b = y_k - w^T \mathbf{x}_k$ for any $\mathbf{x}_k$ such that $\alpha_k \neq 0$

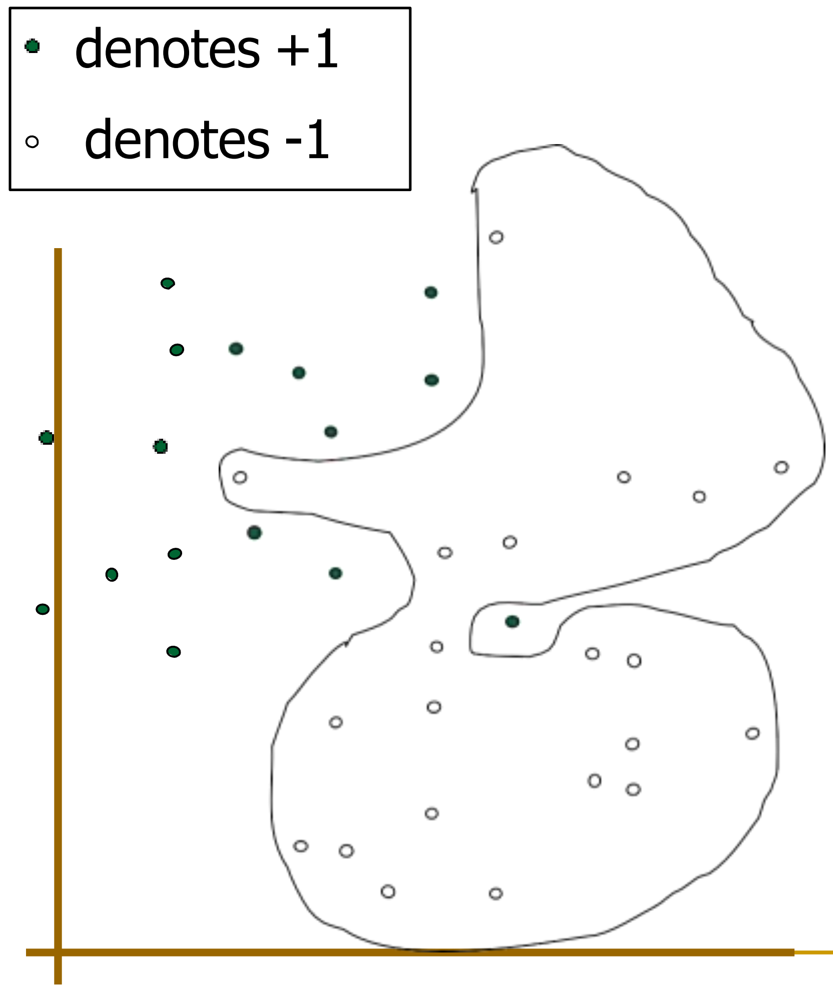
$$\begin{aligned} w^T x + b &= \left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \right)^T x + b \\ &= \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b. \end{aligned}$$

- If we've found the α_i's, in order to make a prediction, we have to calculate a quantity that depends only on the inner product between x and the points in the training set.
-
  - Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a **support vector.**
  - Then the classifying function will have the form:

$$\boxed{f(\mathbf{x}) = \Sigma \alpha_i y_i \mathbf{x_i}^T \mathbf{x} + b}$$

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

Dataset with noise



- **Hard Margin:** So far we require all data points be classified correctly.

  - No training error

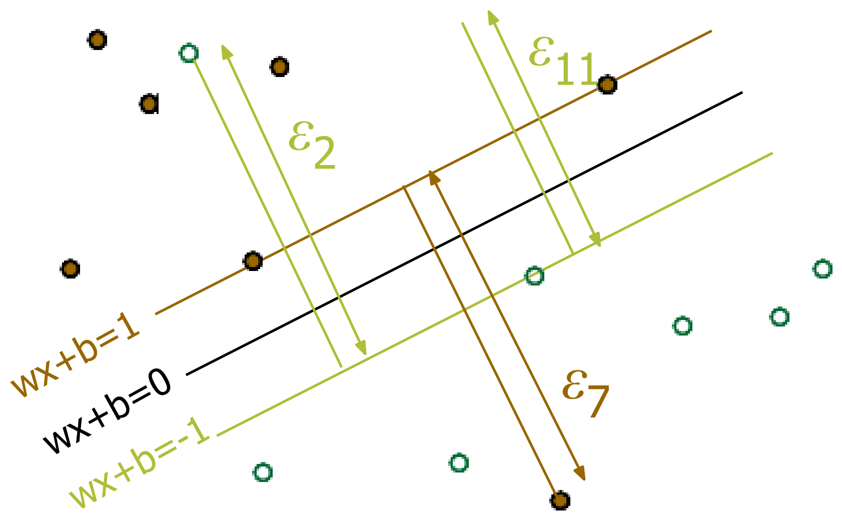
- What if the training set is noisy?

  - Solution 1: use very powerful kernels

OVERFITTING!

Soft Margin Classification

***Slack variables ξi* can be added to allow misclassification of difficult or noisy examples.**



What should our quadratic optimization criterion be?

Minimize

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + \lambda \sum_{k=1}^{R} \varepsilon_k$$

Hard Margin v.s. Soft Margin

- **The old formulation:**

Find **w** and *b* such that
$\mathbf{\Phi}(\mathbf{w}) = ½\, \mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\, (\mathbf{w^T x_i} + b) \geq 1$

- **The new formulation incorporating slack variables:**

Find **w** and *b* such that
$\mathbf{\Phi}(\mathbf{w}) = ½\, \mathbf{w}^T\mathbf{w} + \lambda \Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\, (\mathbf{w^T x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- **Parameter λ can be viewed as a way to control overfitting.**

Linear SVMs: Overview

- The classifier is a *separating hyperplane.*
- Most "important" training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution training points appear only inside dot products:

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

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

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

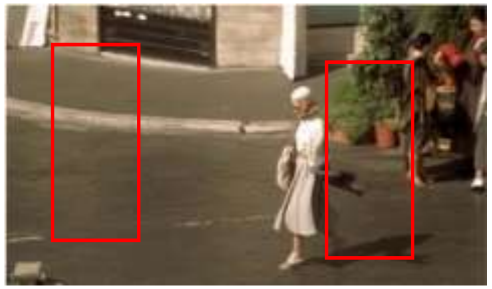
•does an image window contain a person or not?

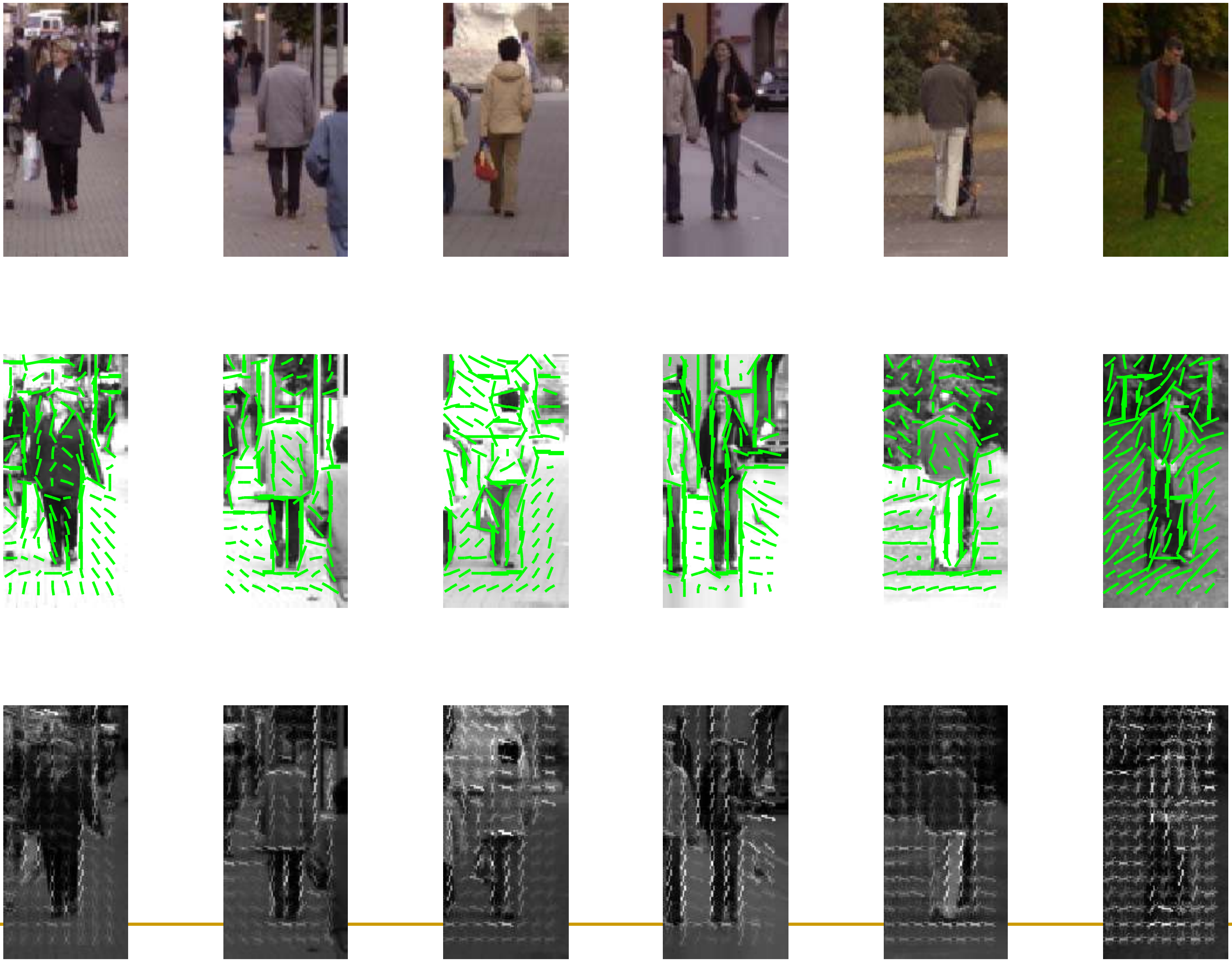
Method: the HOG detector

Training data and features

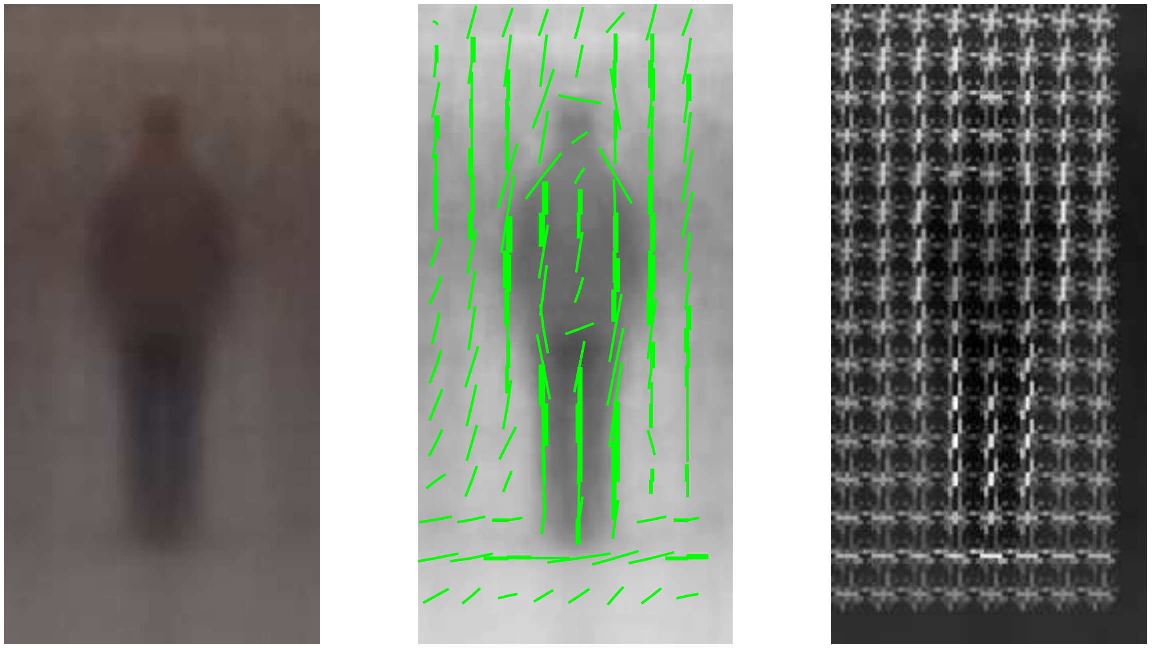
- Positive data – 1208 positive window examples

- Negative data – 1218 negative window examples (initially)





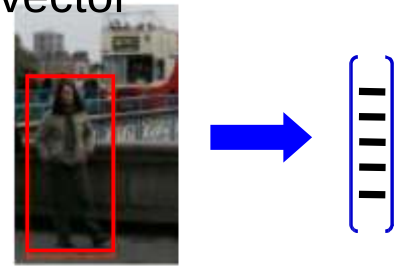
Averaged positive examples



Algorithm

Training (Learning)

- Represent each example window by a HOG feature vector



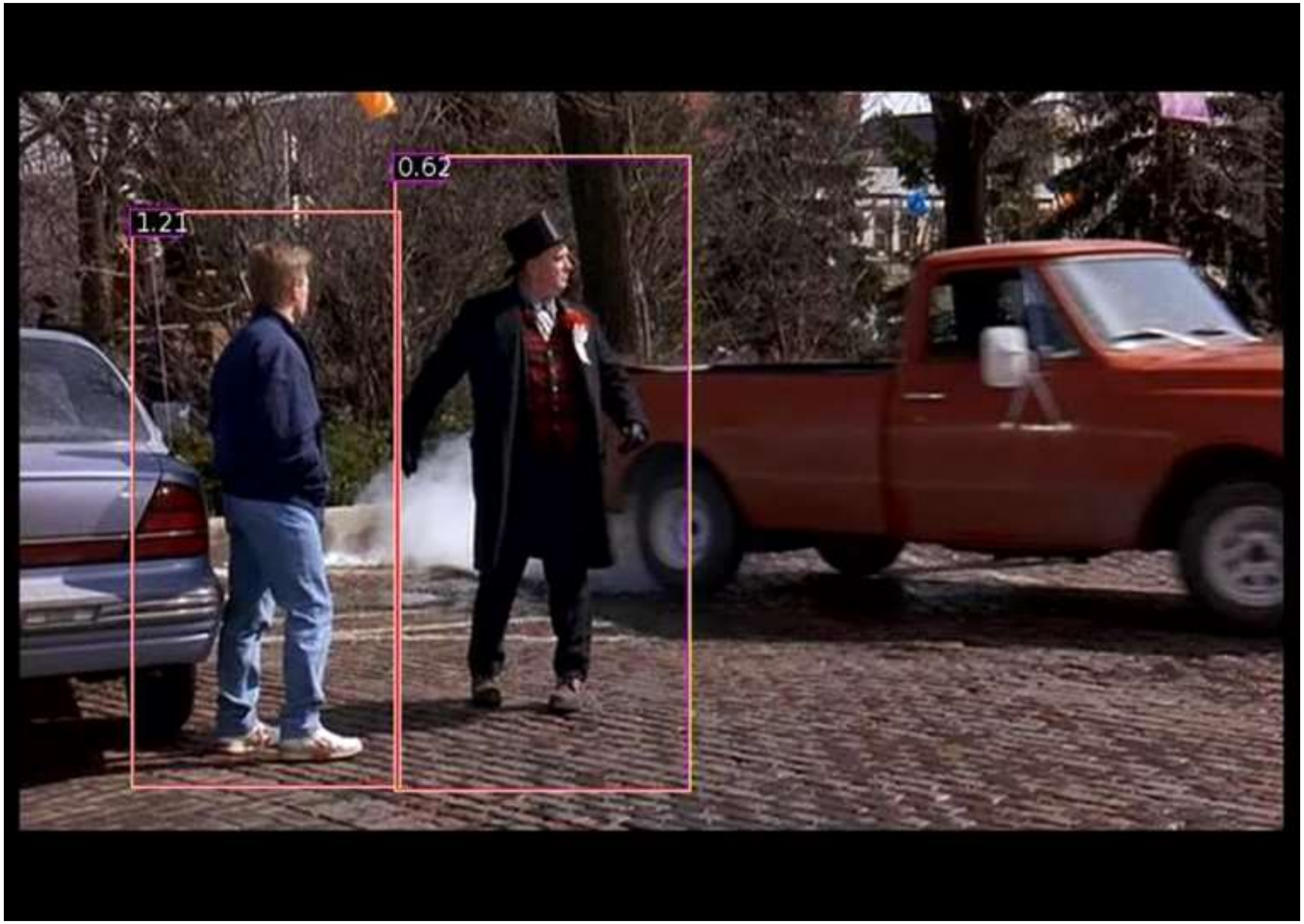
$$\mathbf{x}_i \in \mathbb{R}^d, \text{ with } d = 1024$$

- Train a SVM classifier

Testing (Detection)

- Sliding window classifier

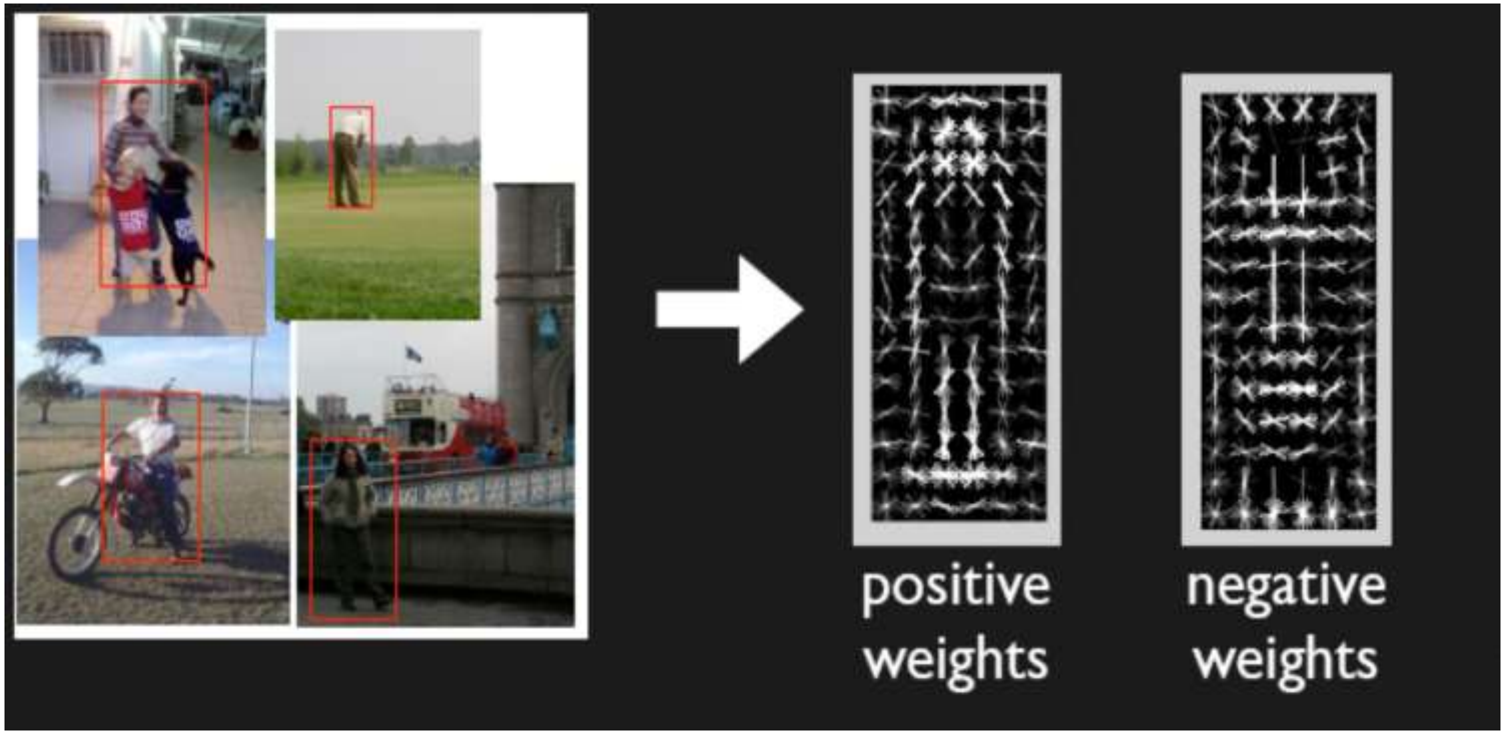
$$f(x) = \mathbf{w}^\top \mathbf{x} + b$$



Dalal and Triggs, CVPR 2005

Learned model

$$f(\mathrm{x}) = \mathrm{w}^{>}\mathrm{x} + b$$



Slide from Deva Ramanan

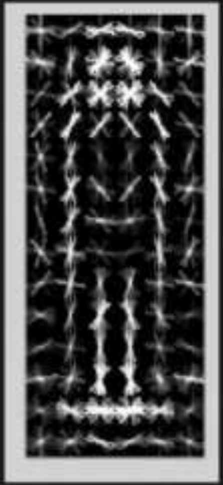
What do negative weights mean?

$$wx > 0$$

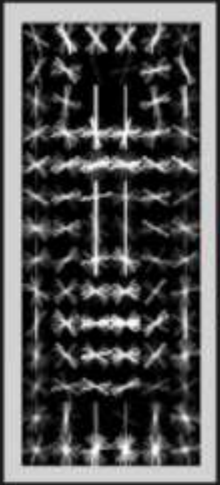
$$(w_+ - w_-)x > 0$$

$$w_+ > w_- x$$

pedestrian model



>



pedestrian background model

Complete system should compete pedestrian/pillar/doorway models

Discriminative models come equipped with own bg

(avoid firing on doorways by penalizing vertical edges)

Slide from Deva Ramanan

Illustration : Linear SVM

- Consider the case of a binary classification starting with a training data of 8 tuples as shown in Table 1.
- Using quadratic programming, we can solve the KKT constraints to obtain the Lagrange multipliers λ_i for each training tuple, which is shown in Table 1.
- Note that only the first two tuples are support vectors in this case.
- Let $W = (w_1, w_2)$ and b denote the parameter to be determined now. We can solve for w_1 and w_2 as follows:

$$w_1 = \sum_i \lambda_i . y_i . x_{i1} = 65.52 \times 1 \times 0.38 + 65.52 \times -1 \times 0.49 = -6.64$$

$$w_2 = \sum_i \lambda_i . y_i . x_{i2} = 65.52 \times 1 \times 0.47 + 65.52 \times -1 \times 0.61 = -9.32$$

Illustration : Linear SVM

Table 1: Training Data

x_1	x_2	y	λ
0.38	0.47	+	65.52
0.49	0.61	-	65.52
0.92	0.41	-	0
0.74	0.89	-	0
0.18	0.58	+	0
0.41	0.35	+	0
0.93	0.81	-	0
0.21	0.10	+	0

Illustration : Linear SVM

Figure 6: Linear SVM example.

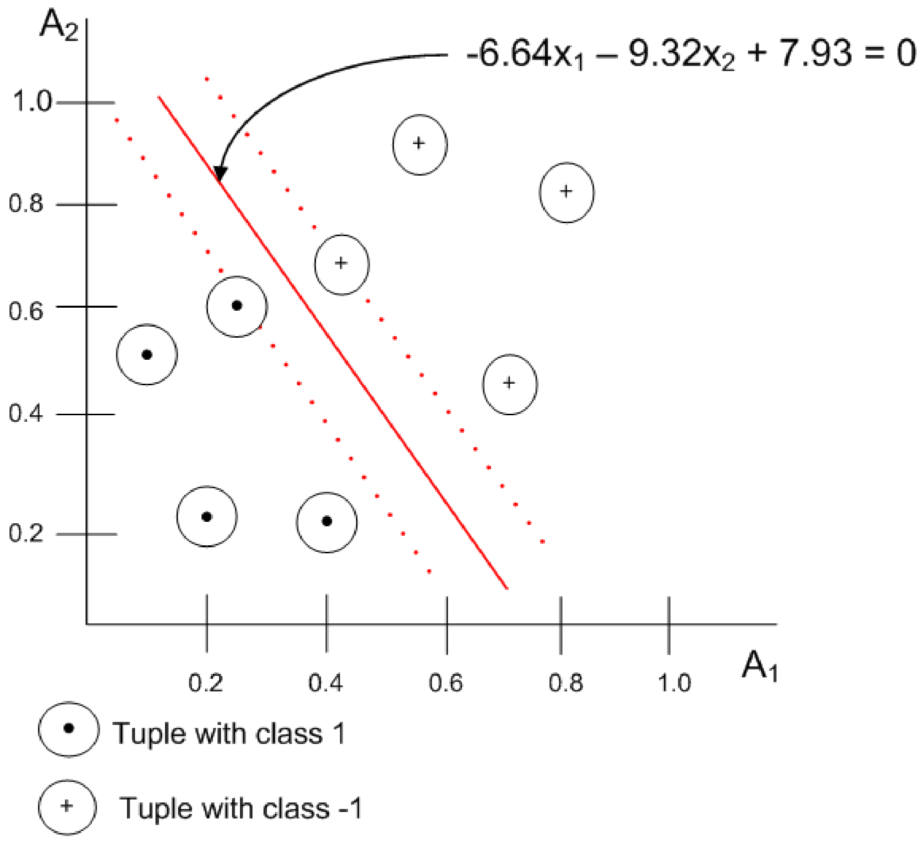


Illustration : Linear SVM

- The parameter b can be calculated for each support vector as follows

  $b_1 = 1 - W.x_1$ // for support vector x_1

  $= 1 - (-6.64) \times 0.38 - (-9.32) \times 0.47$ //using dot product

  $= 7.93$

  $b_2 = 1 - W.x_2$ // for support vector x_2

  $= 1 - (-6.64) \times 0.48 - (-9.32) \times 0.611$ //using dot product

  $= 7.93$

- Averaging these values of b_1 and b_2, we get $b = 7.93$.

Illustration : Linear SVM

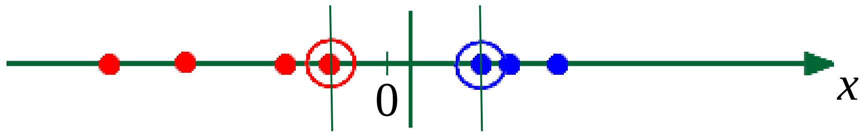
- Thus, the MMH is $-6.64x_1 - 9.32x_2 + 7.93 = 0$ (also see Fig. 6).
- Suppose, test data is $X = (0.5, 0.5)$. Therefore,

  $\delta(X) = W.X + b$
  $= -6.64 \times 0.5 - 9.32 \times 0.5 + 7.93$
  $= -0.05$
  $= -ve$

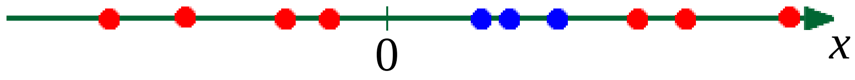
- This implies that the test data falls on or below the MMH and SVM classifies that X belongs to class label -.

Non-linear SVMs

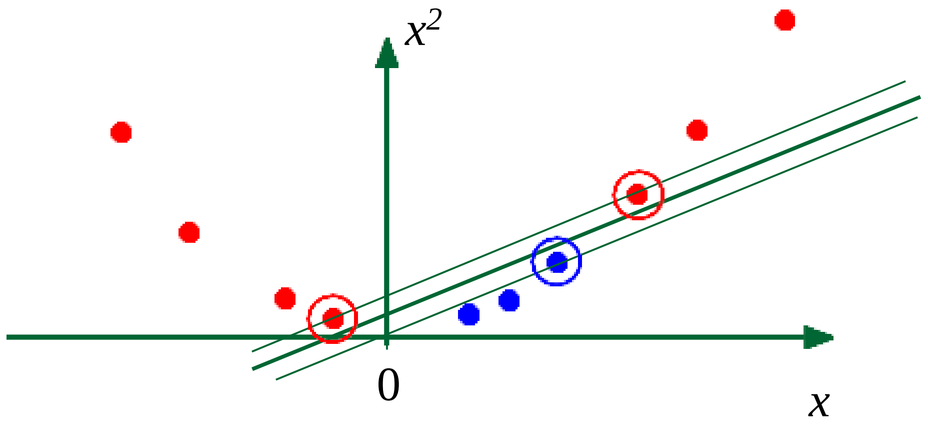
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?



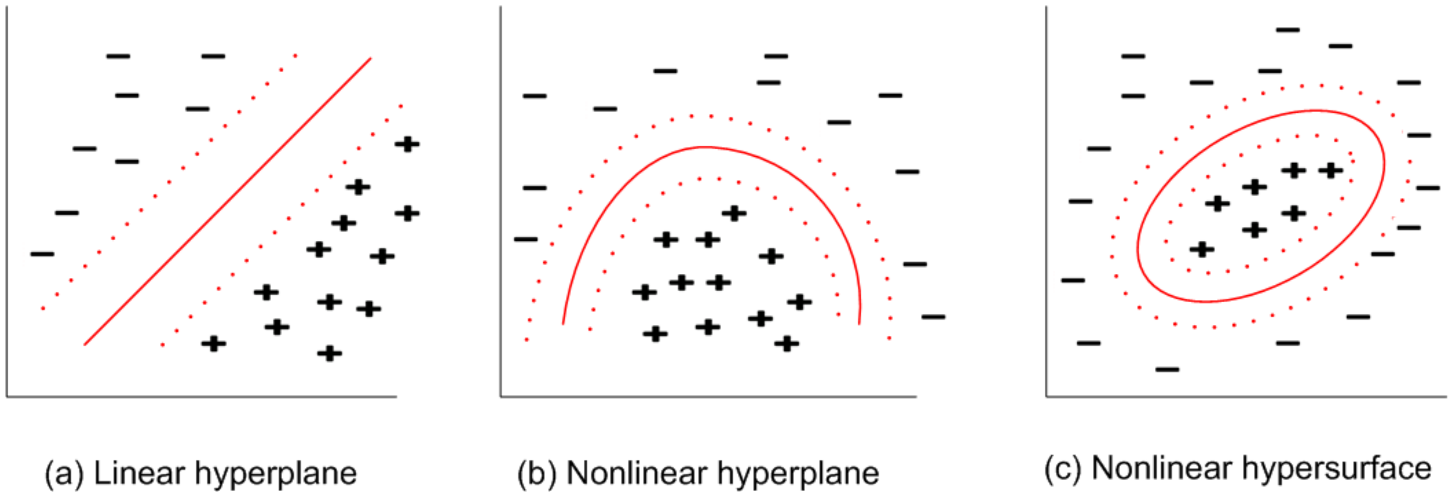
- How about… mapping data to a higher-dimensional space:



Non-Linear SVM

- For understanding this, .
- Note that a linear hyperplane is expressed as a linear equation in terms of *n*-dimensional component, whereas a non-linear hypersurface is a non-linear expression.

Figure 13: 2D view of few class separabilities.



Non-Linear SVM

- A hyperplane is expressed as

$$linear: \; w_1x_1 + w_2x_2 + w_3x_3 + c = 0 \tag{30}$$

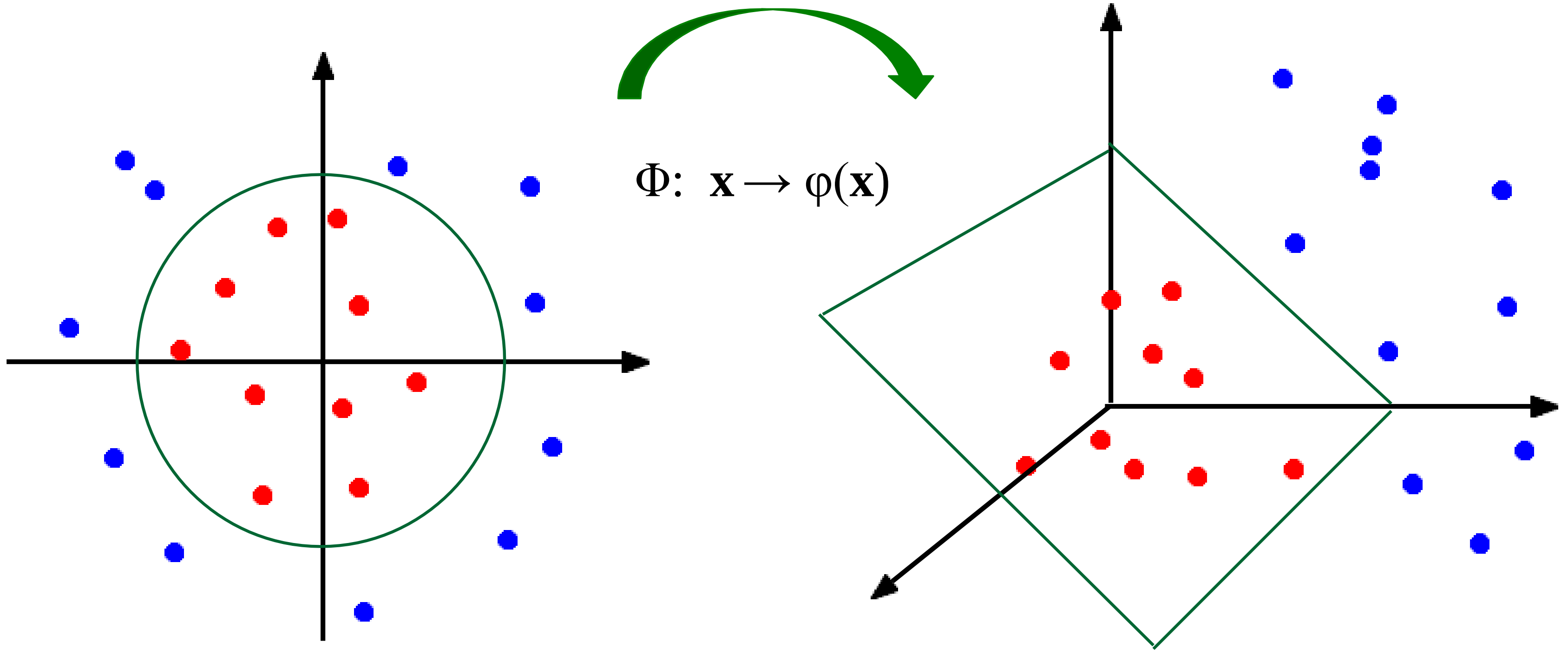
- Whereas a non-linear hypersurface is expressed as.

$$Nonlinear: \; w_1x_1^2 + w_2x_2^2 + w_3x_1x_2 + w_4x_3^2 + w_5x_1x_3 + c = 0 \tag{31}$$

- The task therefore takes a turn to find a nonlinear decision boundaries, that is, nonlinear hypersurface in input space comprising with linearly not separable data.
- This task indeed neither hard nor so complex, and fortunately can be accomplished extending the formulation of linear SVM, we have already learned.

Non-linear SVMs: Feature spaces

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



Mapping the Inputs to other dimensions - the use of Kernels

- Finding the optimal curve to fit the data is difficult.
- There is a way to “pre-process" the data in such a way that the problem is transformed into one of finding a simple hyperplane.
- We define a mapping $z = \varphi(x)$ that transforms the d-dimensional input vector x into a (usually higher) d*-dimensional vector z.
- We hope to choose a $\varphi()$ so that the new training data $\{\varphi(x_i), y_i\}$ is separable by a hyperplane.
- How do we go about choosing $\varphi()$?

Efficient dot-product of polynomials

Polynomials of degree exactly *d*

d=1

$$\phi(u).\phi(v) = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} . \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = u_1 v_1 + u_2 v_2 = u.v$$

d=2

$$\phi(u).\phi(v) = \begin{pmatrix} u_1^2 \\ u_1 u_2 \\ u_2 u_1 \\ u_2^2 \end{pmatrix} . \begin{pmatrix} v_1^2 \\ v_1 v_2 \\ v_2 v_1 \\ v_2^2 \end{pmatrix} \begin{aligned} &= u_1^2 v_1^2 + 2 u_1 v_1 u_2 v_2 + u_2^2 v_2^2 \\ &= (u_1 v_1 + u_2 v_2)^2 \\ &= (u.v)^2 \end{aligned}$$

For any *d* **:**

$$\phi(u).\phi(v) = (u.v)^d$$

- Taking a dot product and exponentiating gives same results as mapping into high dimensional space and then taking dot produce

The "Kernel Trick"

- **The linear classifier relies on dot product between vectors $K(x_i,x_j)=x_i^T x_j$**
- **If every data point is mapped into high-dimensional space via some transformation Φ: $x \rightarrow \varphi(x)$, the dot product becomes:**

$$K(x_i,x_j)= \varphi(x_i)^T\varphi(x_j)$$

- **A *kernel function* is some function that corresponds to an inner product in some expanded feature space.**
- **Example:**

2-dimensional vectors $x=[x_1\ \ x_2]$; let $K(x_i,x_j)=(1 + x_i^Tx_j)^2$,

Need to show that $K(x_i,x_j)= \varphi(x_i)^T\varphi(x_j)$:

$$K(x_i,x_j)=(1 + x_i^Tx_j)^2,$$
$$= 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2} + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$
$$= [1\ \ x_{i1}^2\ \ \sqrt{2}\, x_{i1}x_{i2}\ \ \ x_{i2}^2\ \ \sqrt{2}x_{i1}\ \ \sqrt{2}x_{i2}]^T [1\ \ x_{j1}^2\ \ \sqrt{2}\, x_{j1}x_{j2}\ \ \ x_{j2}^2\ \ \sqrt{2}x_{j1}\ \ \sqrt{2}x_{j2}]$$
$$= \varphi(x_i)^T\varphi(x_j), \quad \text{where } \varphi(x) = [1\ \ x_1^2\ \ \sqrt{2}\, x_1x_2\ \ \ x_2^2\ \ \sqrt{2}x_1\ \ \sqrt{2}x_2]$$

Non-linear SVMs Mathematically

- **Dual problem formulation:**

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(\mathrm{x}_i, \mathrm{x}_j)$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

- **The solution is:**

$$\boxed{f(\mathrm{x}) = \Sigma\alpha_i y_i K(\mathrm{x}_i, \mathrm{x}_j) + b}$$

- **Optimization techniques for finding α_i's remain the same!**

Examples of Kernel Functions

- Linear: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$

- Polynomial of power p: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^p$

- Gaussian (radial-basis function network):

$$K(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2})$$

- Sigmoid: $K(\mathbf{x}_i,\mathbf{x}_j)= \tanh(\beta_0\mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j + \beta_1)$

Nonlinear SVM - Overview

- SVM locates a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the dot product in the feature space.

Properties of SVM

- **Flexibility in choosing a similarity function**
- **Sparseness of solution when dealing with large data sets**

  -only support vectors are used to specify the separating hyperplane
- **Ability to handle large feature spaces**

  -complexity does not depend on the dimensionality of the feature space
- **Overfitting can be controlled by soft margin approach**
- **Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution**
- **Feature Selection**

Weakness of SVM

- **It is sensitive to noise**

  -A relatively small number of mislabeled examples can dramatically decrease the performance

- **It only considers two classes**

  - how to do multi-class classification with SVM?

  - Answer:

  1) with output arity m, learn m SVM's

  - **SVM 1 learns "Output==1" vs "Output != 1"**
  - **SVM 2 learns "Output==2" vs "Output != 2"**
  - **:**
  - **SVM m learns "Output==m" vs "Output != m"**

  2)To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

Some Issues

- **Choice of kernel**
  - Gaussian or polynomial kernel is default
  - if ineffective, more elaborate kernels are needed
  -domain experts can give assistance in formulating appropriate similarity measures

- **Choice of kernel parameters**
  - e.g. σ in Gaussian kernel
  - σ is the distance between closest points with different classifications
  - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.

- **Optimization criterion** – Hard margin v.s. Soft margin
  - a lengthy series of experiments in which various parameters are tested

Additional Resources

- **An excellent tutorial on VC-dimension and Support Vector Machines:**

  C.J.C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):955-974, 1998.

- **The VC/SRM/SVM Bible:**

  Statistical Learning Theory by Vladimir Vapnik, Wiley-Interscience; 1998

http://www.kernel-machines.org/