



# ISTA 421/521

## Introduction to Machine Learning

### Lecture 21: Support Vector Machines

**Clay Morrison**

clayton@sista.arizona.edu

Gould-Simpson 819

Phone 621-6609

6 November 2014



# Support Vector Machines

(SVMs)



# Support Vector Machines (SVMs)

- Considered one of the best off-the-shelf classifiers for most problems – state of the art.
- **BUT**, “No free lunch”: not guaranteed the best
  - Wolpert & Macready 1997
  - “...any two optimization algorithms are equivalent when their performance is averaged across all possible problems.” (from 2005)
- SVMs are particularly useful in applications where the number of attributes is much larger than the number of training objects
  - Number of parameters is based on the number of training objects, not the number of attributes!



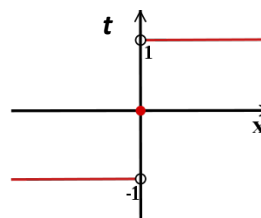
# Support Vector Machines (SVMs)

- Standard SVM uses linear decision boundary given by:  $\mathbf{w}^T \mathbf{x}_{\text{new}} + b$
- SVM **decision function** for test point:

$$t_{\text{new}} = \text{sign}(\mathbf{w}^T \mathbf{x}_{\text{new}} + b)$$

labels are  $\{1, -1\}$  rather than  $\{0, 1\}$

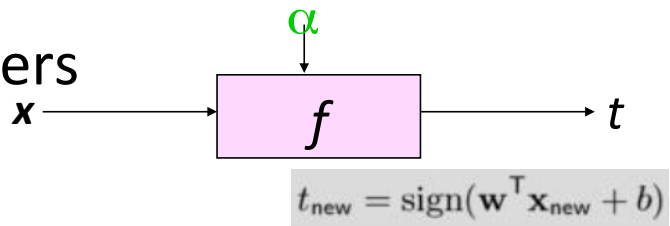
$$\text{sgn}(x) := \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$



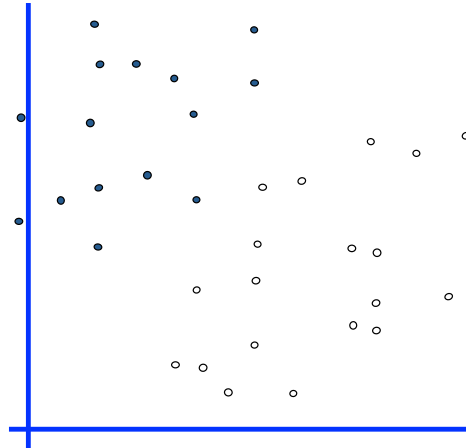
- **Goal**: find  $\mathbf{w}$  and  $b$  based on training data
- **Criteria**: Maximize the **margin**



# Linear Classifiers

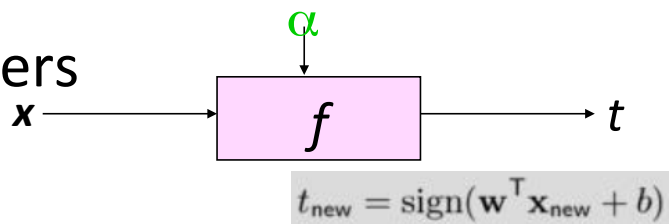


- denotes +1
- denotes -1

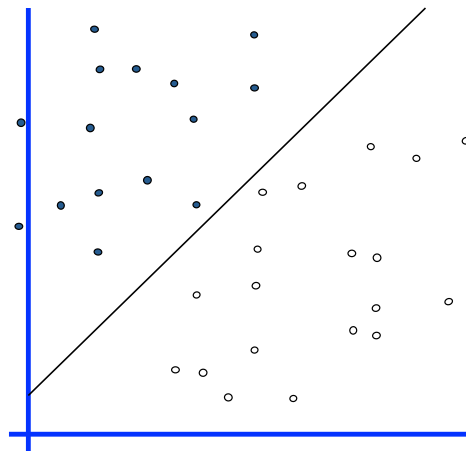


How would you classify this data?

# Linear Classifiers

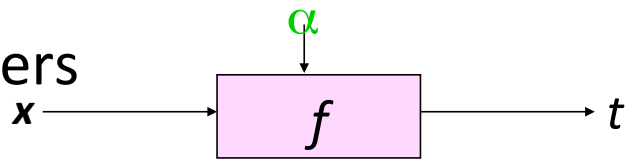


- denotes +1
- denotes -1



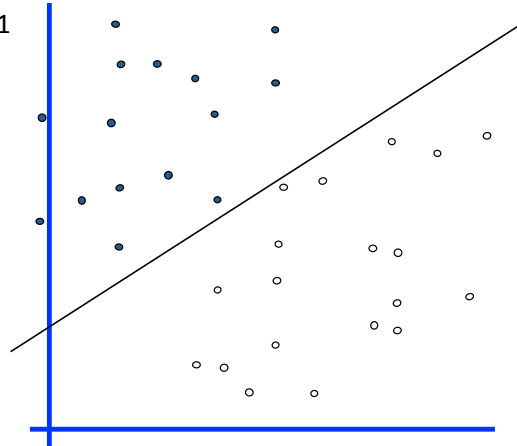
How would you classify this data?

# Linear Classifiers



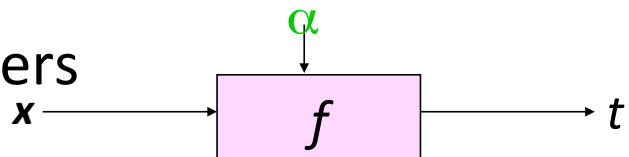
$$t_{\text{new}} = \text{sign}(\mathbf{w}^T \mathbf{x}_{\text{new}} + b)$$

- denotes +1
- denotes -1



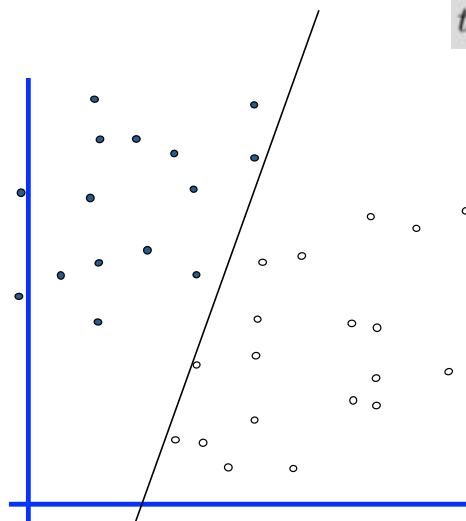
How would you classify this data?

# Linear Classifiers



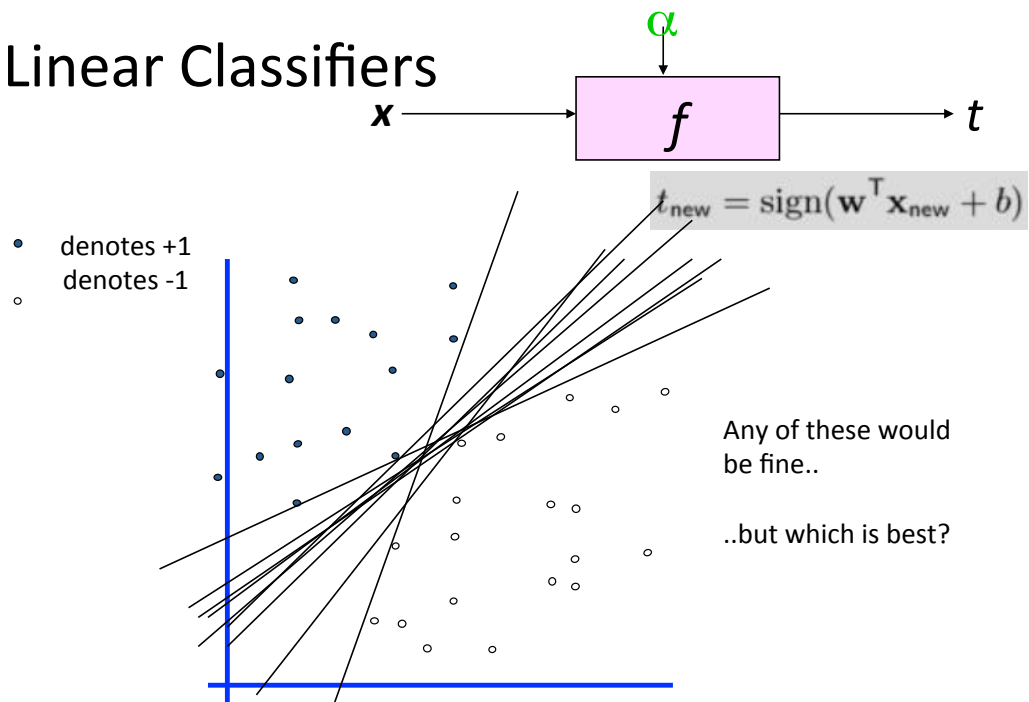
$$t_{\text{new}} = \text{sign}(\mathbf{w}^T \mathbf{x}_{\text{new}} + b)$$

- denotes +1
- denotes -1

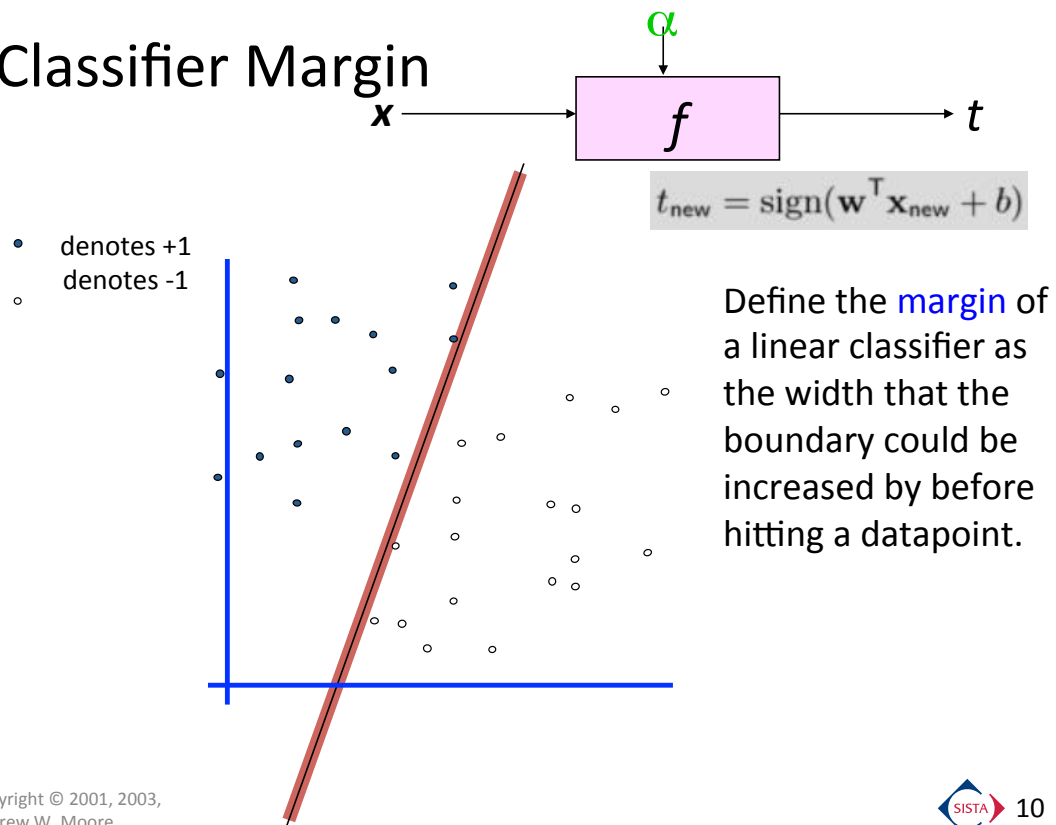


How would you classify this data?

# Linear Classifiers

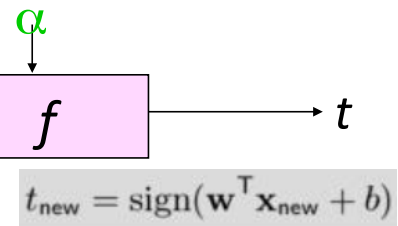
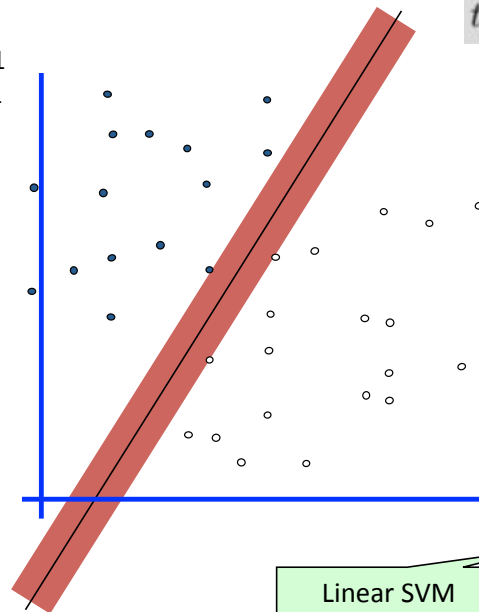


# Classifier Margin



# Maximum Margin

- denotes +1
- denotes -1



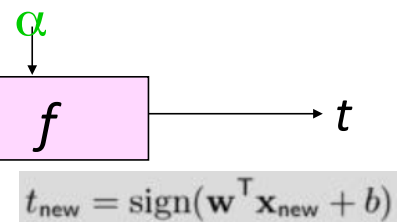
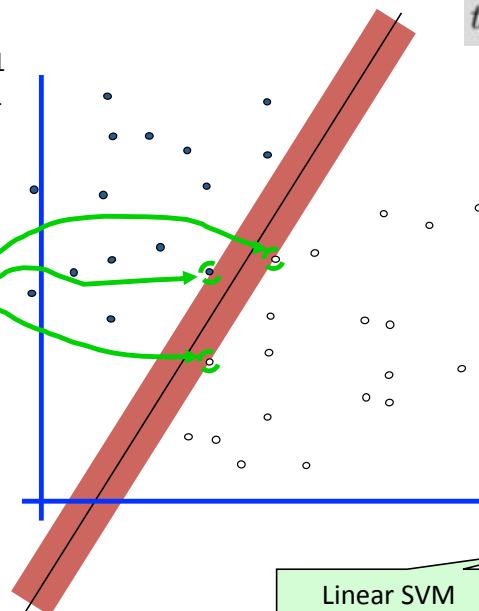
The **maximum margin linear classifier** is the linear classifier with the, um, maximum margin. This is the simplest kind of SVM (Called an LSVM)

Linear SVM

# Maximum Margin

- denotes +1
- denotes -1

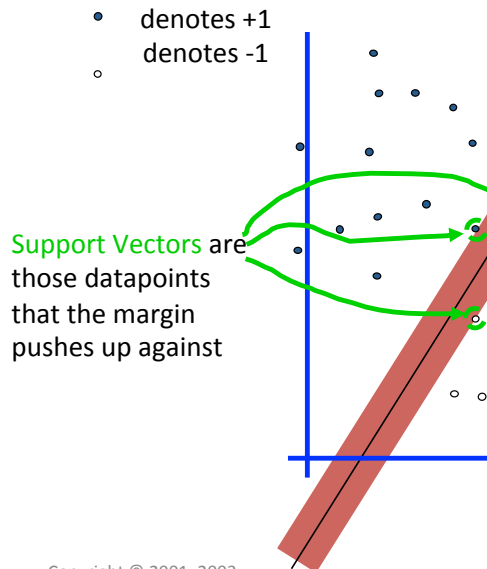
Support Vectors are those datapoints that the margin pushes up against



The **maximum margin linear classifier** is the linear classifier with the, um, maximum margin. This is the simplest kind of SVM (Called an LSVM)

Linear SVM

# Why Maximum Margin?



Copyright © 2001, 2003,  
Andrew W. Moore

1. Intuitively this feels safest.
2. If we've made a small error in the location of the boundary (it's been jolted in its perpendicular direction) this gives us least chance of causing a misclassification.
3. LOOCV is easy since the model is immune to removal of any non-support-vector datapoints.
4. There's some theory (using VC dimension) that is related to (but not the same as) the proposition that this is a good thing.
5. Empirically it works very very well.

## Recall: Relation of $\mathbf{a}^T \mathbf{b}$ to Geometry

- $\mathbf{a}^T \mathbf{b}$  is special (also  $\mathbf{a} \cdot \mathbf{b}$ ), called the *dot product*

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

(This is actually a species of a more general operation called an *inner product*)

- Plays a role in defining
  - Euclidean vector length (norm)

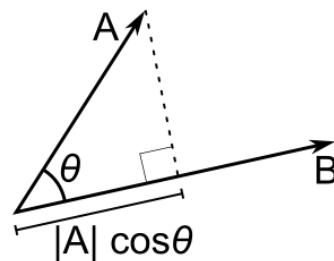
$$\|\mathbf{x}\| := \sqrt{\mathbf{x} \cdot \mathbf{x}}.$$

- Angles

$$\mathbf{a} \cdot \mathbf{a} = \|\mathbf{a}\|^2$$

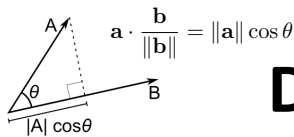
$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \theta$$

$$\theta = \arccos \left( \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|} \right).$$



The dot (inner) product is therefore a measure of the length of vector  $\mathbf{a}$  when we *project* it onto  $\mathbf{b}$  (and normalize by the length (norm) of  $\mathbf{b}$ ):

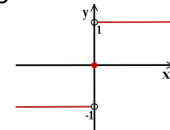
$$\mathbf{a} \cdot \frac{\mathbf{b}}{\|\mathbf{b}\|} = \frac{1}{\|\mathbf{b}\|} \|\mathbf{a}\| \|\mathbf{b}\| \cos \theta = \|\mathbf{a}\| \cos \theta$$



## Defining the Margin

Our “decision function”, which we will also refer to as  $D(x)$ :

$$t_{\text{new}} = \text{sign}(\mathbf{w}^T \mathbf{x}_{\text{new}} + b)$$



We note that the argument in  $D(x)$  is invariant under a rescaling:  $\mathbf{w} \rightarrow \lambda \mathbf{w}$ ,  $b \rightarrow \lambda b$ .

We will implicitly fix a scale with:

$$\mathbf{w} \cdot \mathbf{x}_1 + b = 1$$

$$\mathbf{w} \cdot \mathbf{x}_2 + b = -1$$

for the support vectors (canonical hyperplanes).

Combine both constraints by subtracting one from the other, to get the following:

$$\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = 2$$

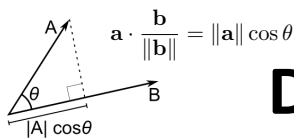
$\mathbf{w}$  is in the direction **perpendicular** to the boundary.

Normalize it to get the “unit vector”:

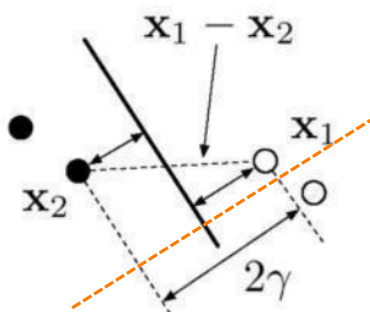
$$\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

The margin (the length of the distance between the support vector canonical hyperplanes) will be given by the **projection** of the vector  $(\mathbf{x}_1 - \mathbf{x}_2)$  onto the normal vector to the hyperplane!

The projection is accomplished by taking the inner product of these two quantities



## Defining the Margin



$\mathbf{w}$  is in the direction **perpendicular** to the boundary.

Normalize it to get the “unit vector”:

$$\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

**Constraints:**

$$\mathbf{w} \cdot \mathbf{x}_1 + b = 1$$

$$\mathbf{w} \cdot \mathbf{x}_2 + b = -1$$

$$\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = 2$$

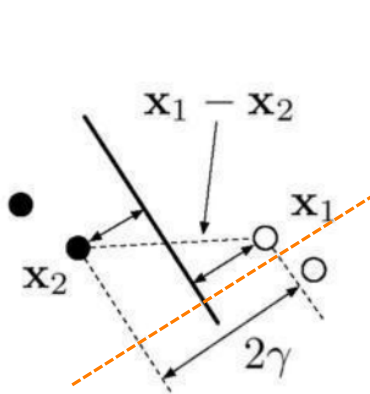
$$\begin{aligned} 2\gamma &= \frac{1}{\|\mathbf{w}\|} \mathbf{w}^T (\mathbf{x}_1 - \mathbf{x}_2) \\ &= \frac{1}{\|\mathbf{w}\|} (\mathbf{w}^T \mathbf{x}_1 - \mathbf{w}^T \mathbf{x}_2) \\ &= \frac{1}{\|\mathbf{w}\|} (\mathbf{w}^T \mathbf{x}_1 + b - \mathbf{w}^T \mathbf{x}_2 - b) \\ &= \frac{1}{\|\mathbf{w}\|} (1 - (-1)) \\ &= \frac{1}{\|\mathbf{w}\|} \cdot 2 \\ \gamma &= \frac{1}{\|\mathbf{w}\|} \end{aligned}$$

The margin (the length of the distance between the support vector canonical hyperplanes) will be given by the **projection** of the vector  $(\mathbf{x}_1 - \mathbf{x}_2)$  onto the normal vector to the hyperplane!

The projection is accomplished by taking the inner product of these two quantities



# Maximizing the Margin



$$\gamma = \frac{1}{\|\mathbf{w}\|}$$

Recall, we set a scale for the closest points to the margin:

$$\mathbf{w} \cdot \mathbf{x}_1 + b = 1 \quad \text{For the s.v. class 1}$$

$$\mathbf{w} \cdot \mathbf{x}_2 + b = -1 \quad \text{For the s.v. class 2}$$

Therefore,  $\mathbf{w}$  must be chosen such that:


$$\mathbf{w} \cdot \mathbf{x}_n + b \geq 1 \quad \text{for all } \mathbf{x}_n \text{ in class 1}$$

$$\mathbf{w} \cdot \mathbf{x}_n + b \leq -1 \quad \text{for all } \mathbf{x}_n \text{ in class 2}$$

Combining both constraints is easy:

$$t_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 \quad \text{For all } \mathbf{x}_n$$

There are a total of  $N$  constraints

Easier to *minimize*  $\frac{1}{2} \|\mathbf{w}\|^2$   17

## Constrained Optimization with Lagrange Multipliers

- Find values of a set of parameters that maximize (or minimize) an objective function, but also satisfy some constraints.
- Create new objective function that includes the original plus an additional term for each constraint.

For example, minimize  $f(x)$  subject to the constraint  $g(w) \leq a$

$$\begin{aligned} &\underset{w}{\operatorname{argmin}} \quad f(w) \\ &\text{subject to} \quad g(w) \leq a \end{aligned}$$

Add **Lagrange term** of the form  $\lambda(g(w) - a)$  and optimize for  $w$  and  $\lambda$

$$\begin{aligned} &\underset{w, \lambda}{\operatorname{argmin}} \quad f(w) - \lambda(g(w) - a) \\ &\text{subject to} \quad \lambda > 0. \end{aligned}$$

# Maximizing the Margin $\gamma = \frac{1}{\|\mathbf{w}\|}$

$$\frac{1}{2}\|\mathbf{w}\|^2 \quad t_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1$$

Maximizing the margin,  $\gamma$ , becomes a **constrained optimization** problem, namely, to minimize the following:

$$\begin{aligned} & \underset{\mathbf{w}}{\operatorname{argmin}} \quad \frac{1}{2}\|\mathbf{w}\|^2 \\ & \text{subject to} \quad t_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1, \text{ for all } n \end{aligned}$$

We can incorporate the inequalities into the minimization by introducing **Lagrange multipliers**, resulting in the following:

$$\begin{aligned} & \underset{\mathbf{w}, \alpha}{\operatorname{argmin}} \quad \frac{1}{2}\mathbf{w}^\top \mathbf{w} - \sum_{n=1}^N \alpha_n (t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1) \\ & \text{subject to} \quad \alpha_n \geq 0, \text{ for all } n, \end{aligned}$$

note:  $\|\mathbf{w}\|^2 = \mathbf{w}^\top \mathbf{w}$

Recall how we maximize/minimize!

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{n=1}^N \alpha_n t_n \mathbf{x}_n \\ \frac{\partial}{\partial b} &= - \sum_{n=1}^N \alpha_n t_n. \end{aligned}$$

Set to 0!

$$\begin{aligned} \mathbf{w} &= \sum_{n=1}^N \alpha_n t_n \mathbf{x}_n \\ \sum_{n=1}^N \alpha_n t_n &= 0 \end{aligned}$$

These two identities must be satisfied at the optimum

# Maximizing the Margin $\gamma = \frac{1}{\|\mathbf{w}\|}$

$$\frac{1}{2}\|\mathbf{w}\|^2 \quad t_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1$$

$$\begin{aligned} & \underset{\mathbf{w}, \alpha}{\operatorname{argmin}} \quad \frac{1}{2}\mathbf{w}^\top \mathbf{w} - \sum_{n=1}^N \alpha_n (t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1) \\ & \text{subject to} \quad \alpha_n \geq 0, \text{ for all } n, \end{aligned} \quad \begin{aligned} \mathbf{w} &= \sum_{n=1}^N \alpha_n t_n \mathbf{x}_n \\ \sum_{n=1}^N \alpha_n t_n &= 0 \end{aligned}$$

Plug constraint for  $\mathbf{w}$  back into the objective function, to get:

$$\begin{aligned} & \frac{1}{2}\mathbf{w}^\top \mathbf{w} - \sum_{n=1}^N \alpha_n (t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1) \\ &= \frac{1}{2} \left( \sum_{m=1}^N \alpha_m t_m \mathbf{x}_m^\top \right) \left( \sum_{n=1}^N \alpha_n t_n \mathbf{x}_n \right) - \sum_{n=1}^N \alpha_n \left( t_n \left( \sum_{m=1}^N \alpha_m t_m \mathbf{x}_m^\top \mathbf{x}_n + b \right) - 1 \right) \\ &= \frac{1}{2} \sum_{n,m=1}^N \alpha_m \alpha_n t_m t_n \mathbf{x}_m^\top \mathbf{x}_n - \sum_{n,m=1}^N \alpha_m \alpha_n t_m t_n \mathbf{x}_m^\top \mathbf{x}_n - \sum_{n=1}^N \alpha_n t_n b + \sum_{n=1}^N \alpha_n \\ &= \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n,m=1}^N \alpha_m \alpha_n t_m t_n \mathbf{x}_m^\top \mathbf{x}_n \end{aligned}$$

This goes away by this constraint

This is the **dual** optimization problem (we have eliminated  $\mathbf{w}$ !)

It is a **quadratic optimization** problem due to the  $\alpha_m \alpha_n$  term (matlab: quadprog)

# Making Predictions

Our final constraint problem:

$$\sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n,m=1}^N \alpha_m \alpha_n t_m t_n \mathbf{x}_m^\top \mathbf{x}_n$$

Subject to:  $\alpha_n \geq 0, \sum_{n=1}^N \alpha_n t_n = 0.$

Give it to a quadratic programming solver!

To predict, we need our decision function  $D(\mathbf{x})$

$$t_{\text{new}} = \text{sign}(\mathbf{w}^\top \mathbf{x}_{\text{new}} + b)$$

But we just optimized for  $\alpha$ 's!

Recall: 
$$\mathbf{w} = \sum_{n=1}^N \alpha_n t_n \mathbf{x}_n$$

So rewrite the decision function as:

$$t_{\text{new}} = \text{sign}\left(\sum_{n=1}^N \alpha_n t_n \mathbf{x}_n^\top \mathbf{x}_{\text{new}} + b\right)$$

To find  $b$ , we will use the fact that for the closest points,  $t_n (\mathbf{w}^\top \mathbf{x}_n + b) = 1$

$$b = t_n - \sum_{m=1}^N \alpha_m t_m \mathbf{x}_m^\top \mathbf{x}_n$$

(note that  $t_n = 1/t_n$  in this case)

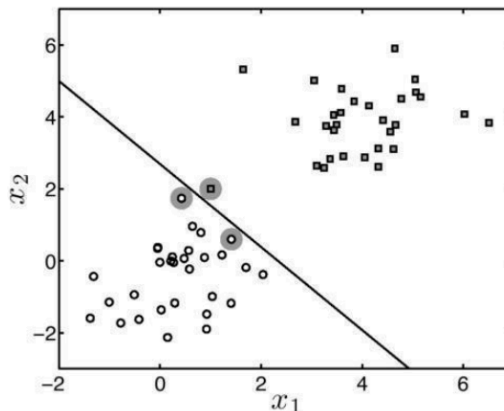
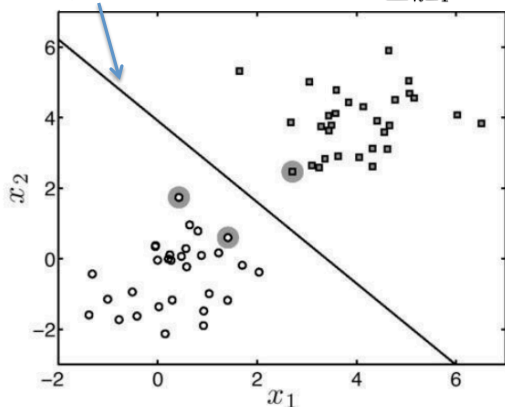
NOTE: after optimizing for all  $\alpha_n$ 's, the only  $\alpha$ 's that are **non-zero** are the support vectors! <sup>1</sup>

## Hard Margin SMV

$$t_{\text{new}} = \text{sign}\left(\sum_{n=1}^N \alpha_n t_n \mathbf{x}_n^\top \mathbf{x}_{\text{new}} + t_n - \sum_{m=1}^N \alpha_m t_m \mathbf{x}_m^\top \mathbf{x}_n\right)$$

After optimizing for all  $\alpha_n$ 's, the only  $\alpha$ 's that are **non-zero** are the support vectors!

( $\mathbf{w}^\top \mathbf{x} + b = 0$ , where  $\mathbf{w} = \sum_{n=1}^N \alpha_n t_n \mathbf{x}_n$ )



## Soft Margin SVM

- To allow points to lie on the wrong side of the boundary, need to “slacken” the constraints

$$t_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n \quad \text{where } \xi_n \geq 0$$

- If  $0 \leq \xi_n \leq 1$ 
  - Then the point lies on the correct side of the boundary, but within the boundary margin
- If  $\xi_n > 1$ 
  - Then the point lies on the “wrong” side of the boundary

## Soft Margin SVM

- To allow points to lie on the wrong side of the boundary, need to “slacken” the constraints

$$t_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n \quad \text{where } \xi_n \geq 0$$

- The optimization task becomes:

$$\underset{\mathbf{w}}{\operatorname{argmin}} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{n=1}^N \xi_n$$

$$\text{subject to } \xi_n \geq 0 \text{ and } t_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n \text{ for all } n$$

Recall, the original constrained optimization was:

$$\underset{\mathbf{w}}{\operatorname{argmin}} \quad \frac{1}{2} \|\mathbf{w}\|^2$$

subject to  $t_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1$ , for all  $n$

- $C$  controls to what extent we are willing to allow points to sit within the margin itself or on the wrong side of the decision boundary

# Soft Margin SVM

- To allow points to lie on the wrong side of the boundary, need to “slacken” the constraints

$$t_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 - \xi_n \quad \text{where} \quad \xi_n \geq 0$$

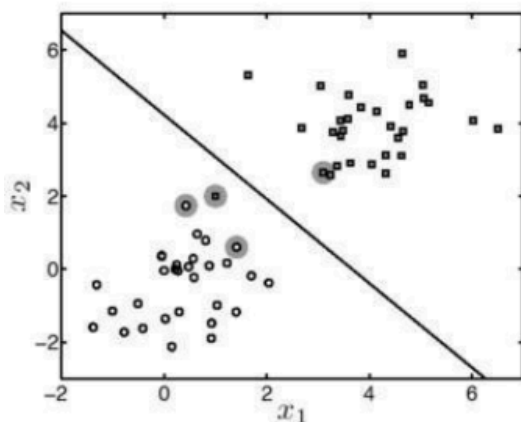
- It turns out that incorporating the new constraint  $C$  does not change the overall optimization much!:

Recall:  $\sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n,m=1}^N \alpha_n \alpha_m t_n t_m \mathbf{x}_n^\top \mathbf{x}_m \quad \alpha_n \geq 0, \sum_{n=1}^N \alpha_n t_n = 0.$

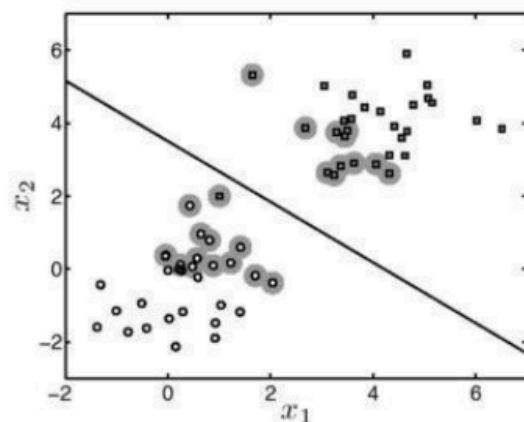
$$\begin{aligned} & \underset{\mathbf{w}}{\operatorname{argmax}} \quad \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n,m=1}^N \alpha_n \alpha_m t_n t_m \mathbf{x}_n^\top \mathbf{x}_m \\ & \text{subject to} \quad \sum_{n=1}^N \alpha_n t_n = 0 \quad \text{and} \quad 0 \leq \alpha_n \leq C, \text{ for all } n. \end{aligned}$$

Just adds an upper bound on the influence any training point can have

## Decision Boundary & Support Vectors for different $C$



(a)  $C = 1$



(b)  $C = 0.01$