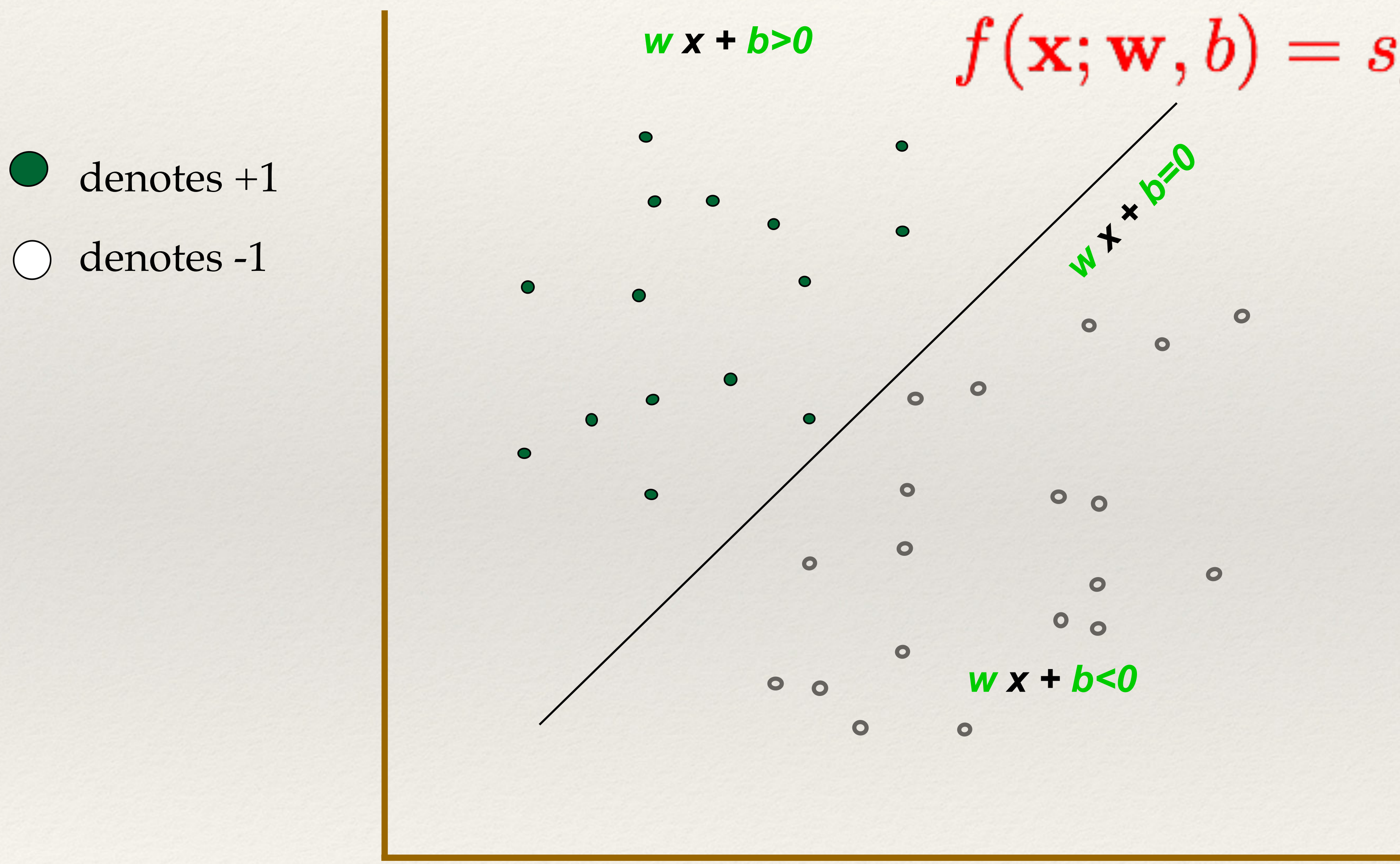
# *E9 205 Machine Learning for Signal Procesing*

## Support Vector Machines

9-10-2019

# Linear Classifiers

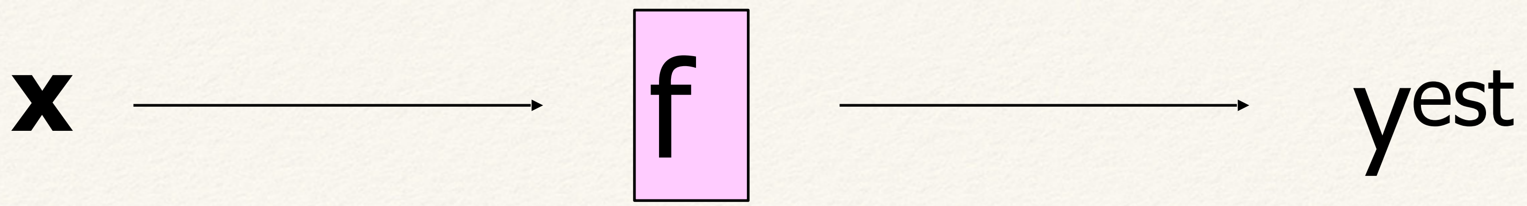

$$f(\mathbf{x}; \mathbf{w}, b) = sgn(\mathbf{w}^T \phi(\mathbf{x}) + b)$$




How would you classify this data?

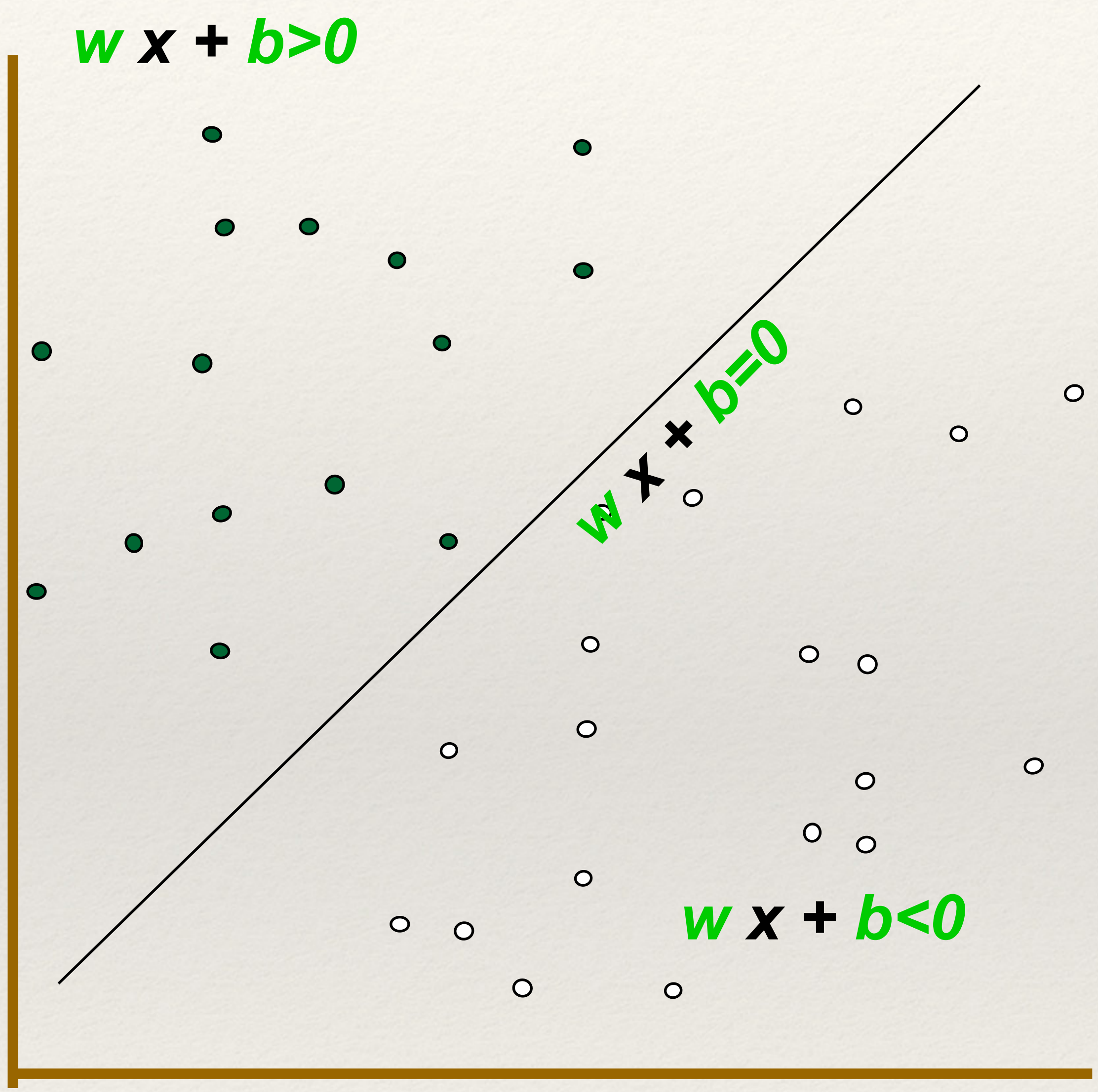
*"SVM and applications", Mingyue Tan. Univ of British Columbia*

# Linear Classifiers




$$f(\mathbf{x}; \mathbf{w}, b) = sgn(\mathbf{w}^T \phi(\mathbf{x}) + b)$$

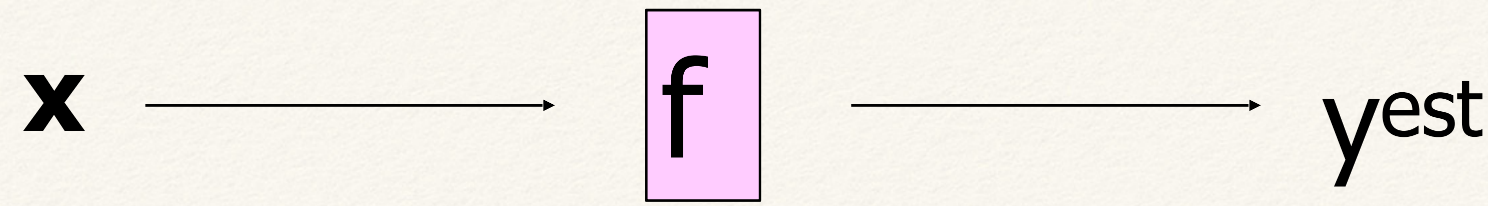
- denotes +1
- ° denotes -1



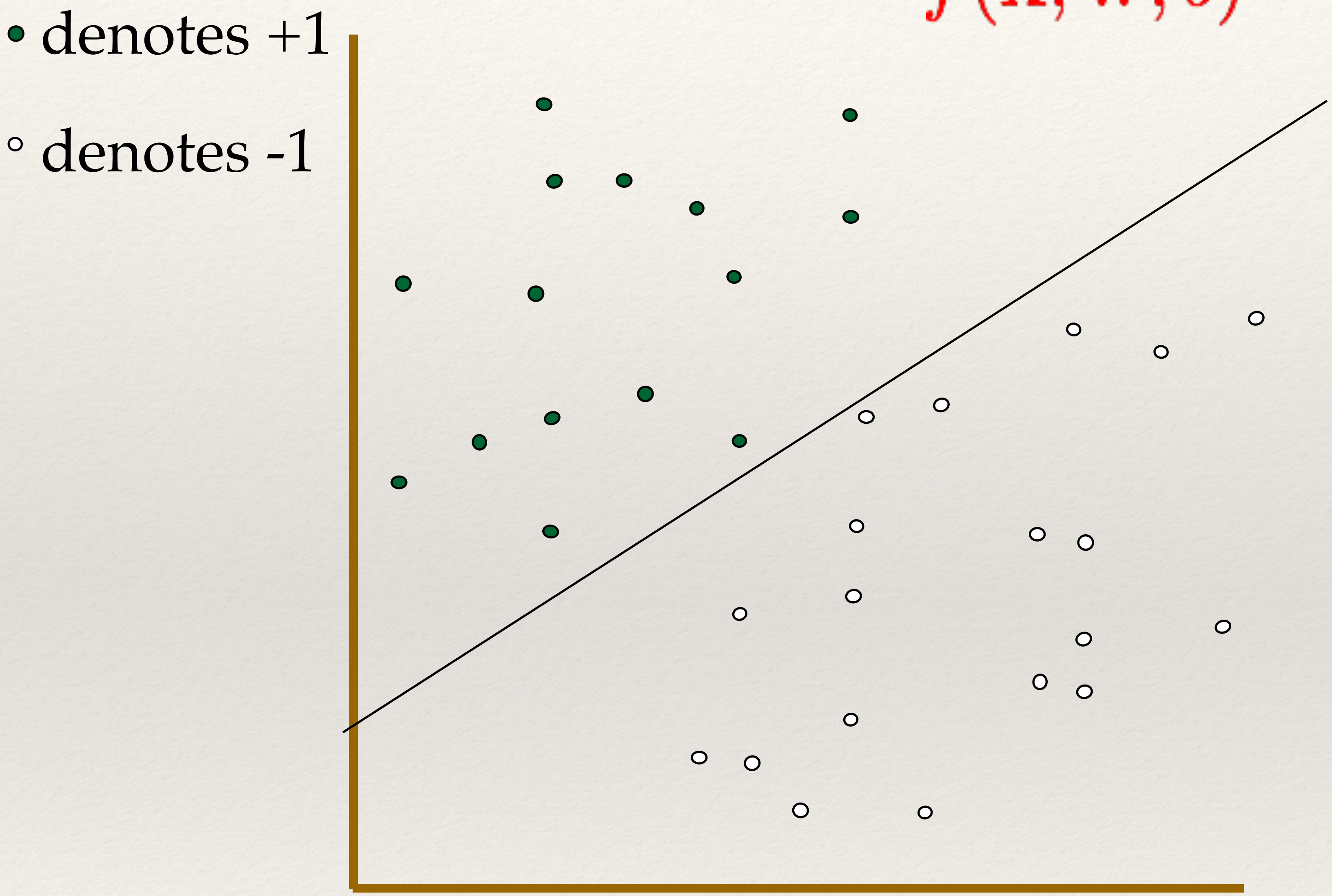

How would you classify this data?

*"SVM and applications", Mingyue Tan. Univ of British Columbia*

# Linear Classifiers

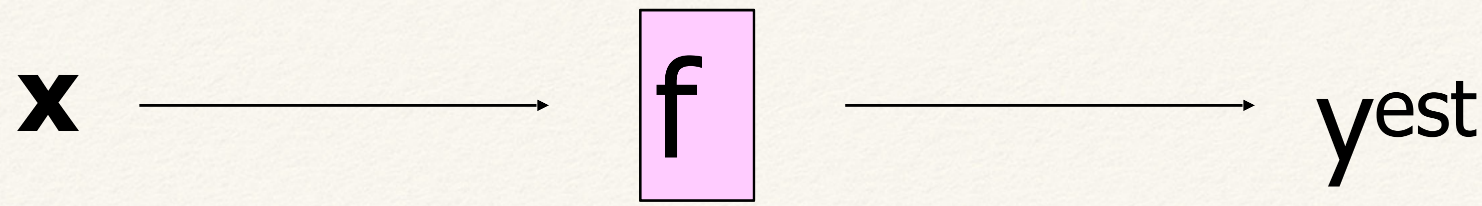



$$f(\mathbf{x}; \mathbf{w}, b) = sgn(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) + b)$$



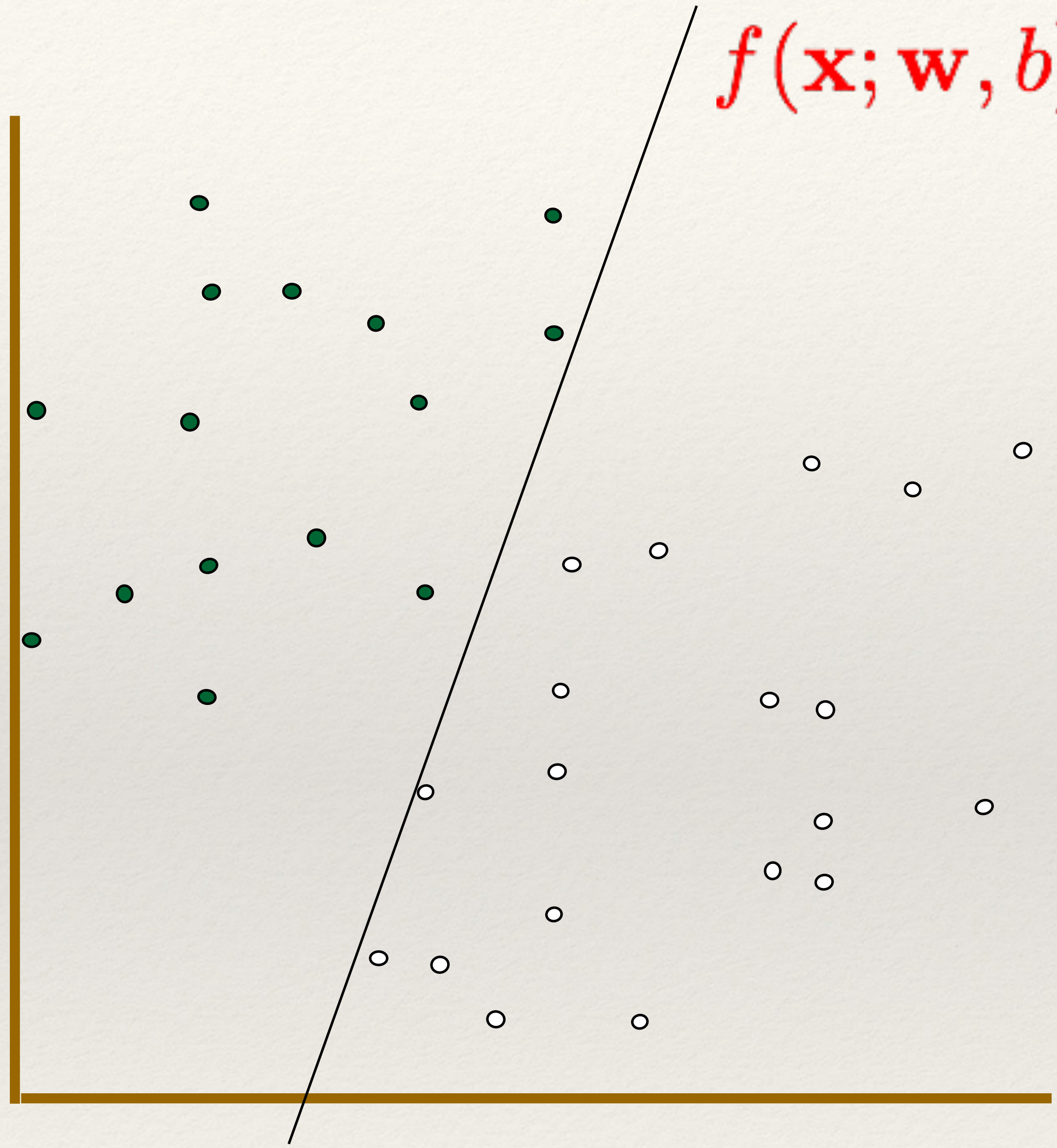

How would you classify this data?

# Linear Classifiers




$$f(\mathbf{x}; \mathbf{w}, b) = sgn(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) + b)$$

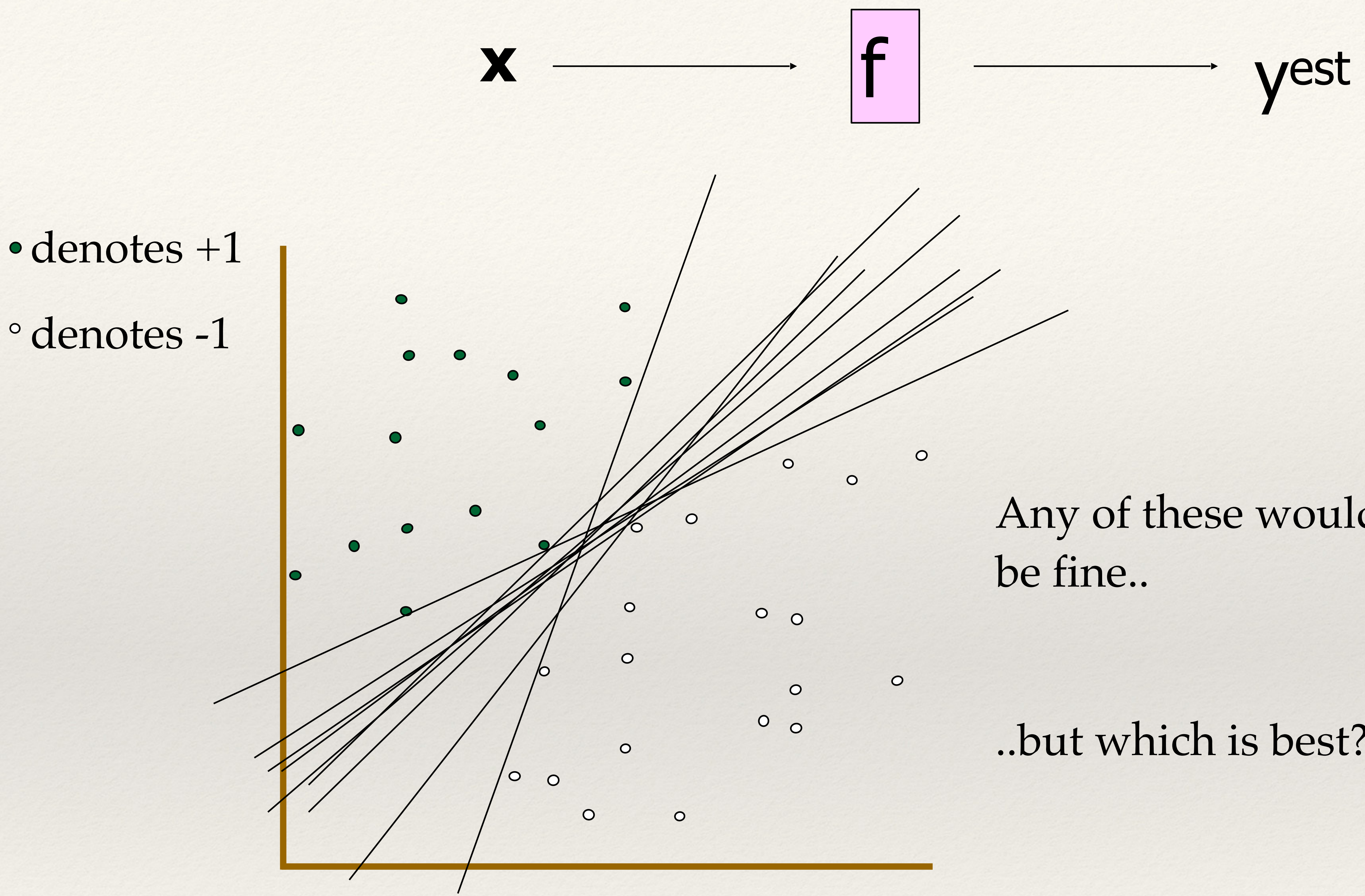
- denotes +1
- ◦ denotes -1



How would you classify this data?

*"SVM and applications", Mingyue Tan. Univ of British Columbia*

# Linear Classifiers

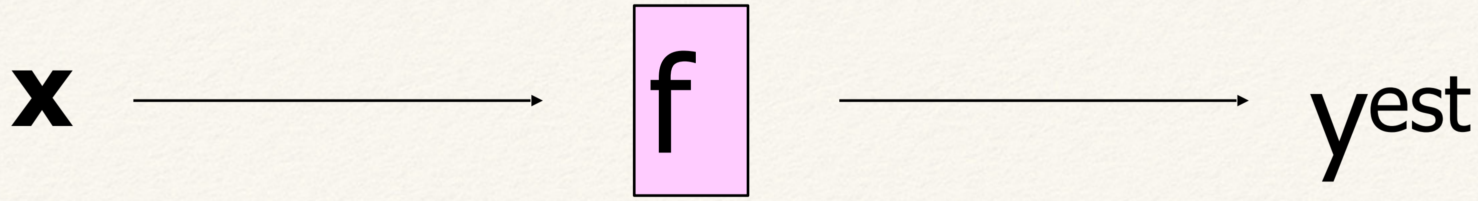



Any of these would be fine..

..but which is best?

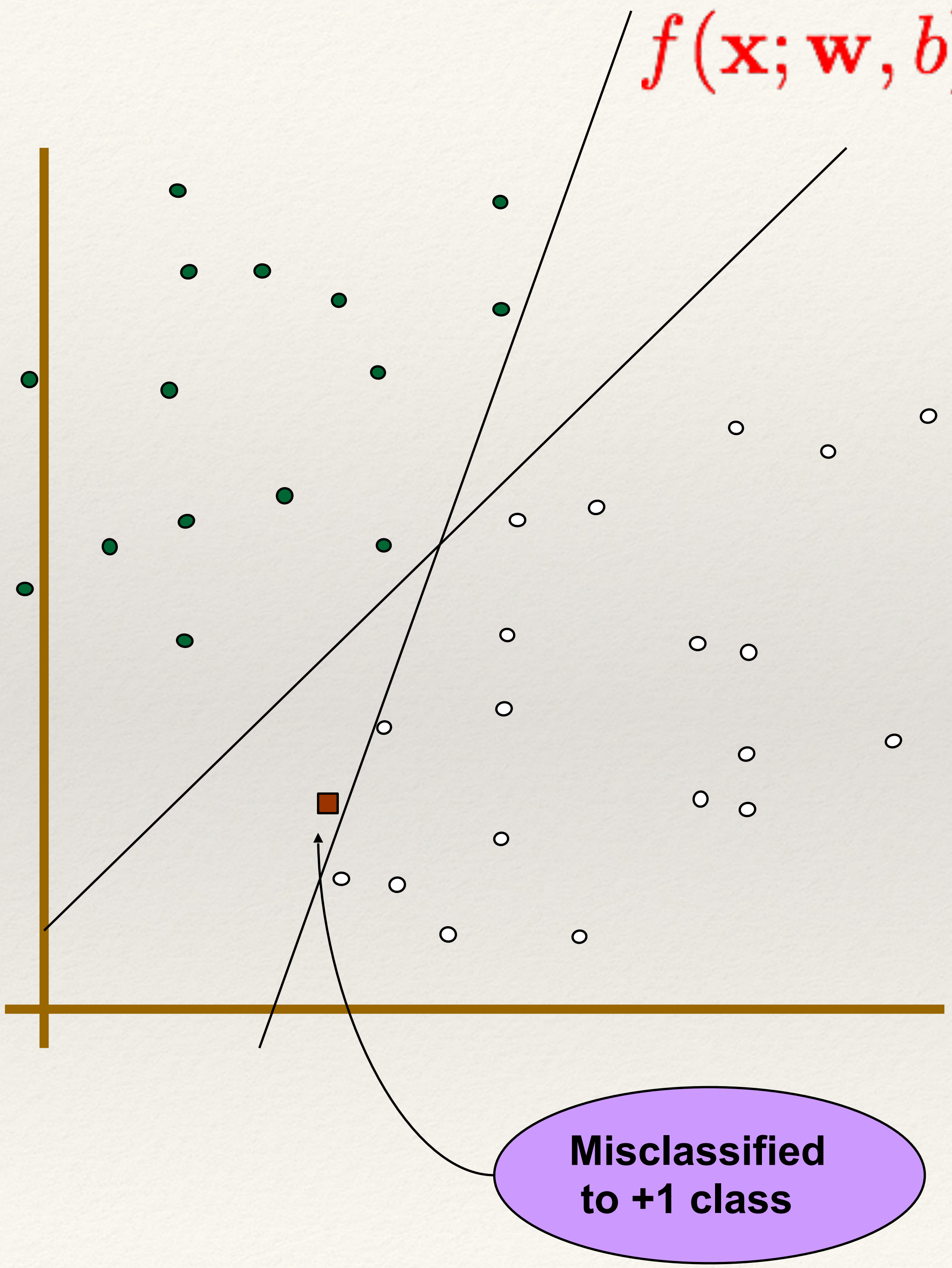
*"SVM and applications", Mingyue Tan. Univ of British Columbia*

# Linear Classifiers




$$f(\mathbf{x}; \mathbf{w}, b) = sgn(\mathbf{w}^T \phi(\mathbf{x}) + b)$$

- denotes +1
- ◦ denotes -1

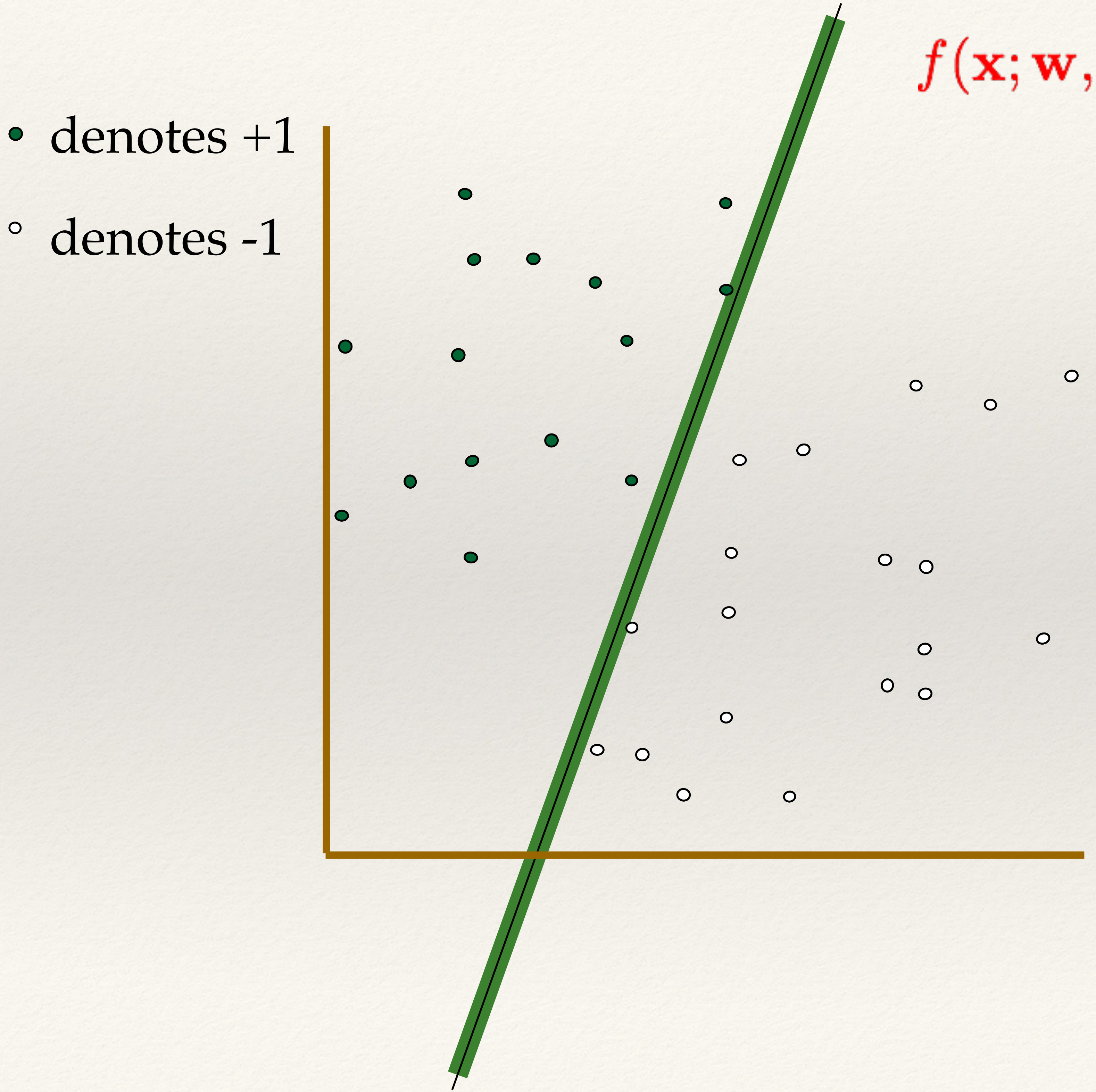



How would you classify this data?

*"SVM and applications", Mingyue Tan. Univ of British Columbia*

# Linear Classifiers


$$f(\mathbf{x};\mathbf{w},b) = sgn(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) + b)$$




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

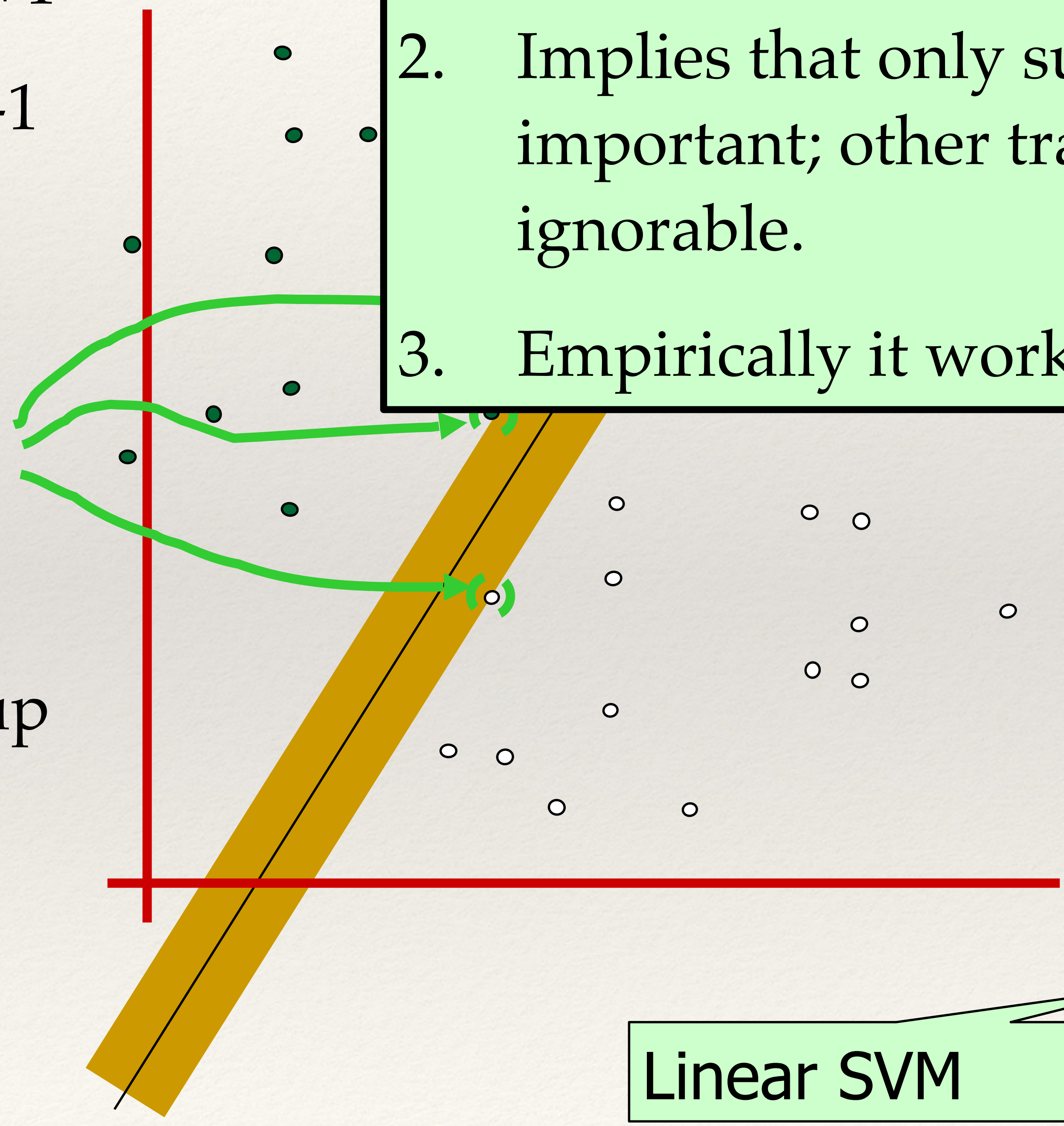
# Maximum Margin

**x** → f → yest

1. Maximizing the margin is good according to intuition
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very very well.

• denotes +1

° denotes -1



Support Vectors are those data points that the margin pushes up against

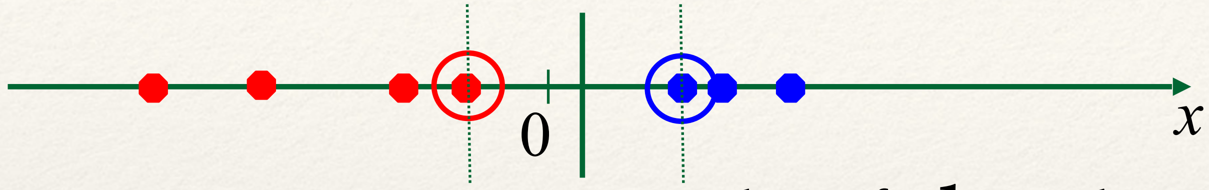
linear classifier with the, um, maximum margin.

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

Linear SVM

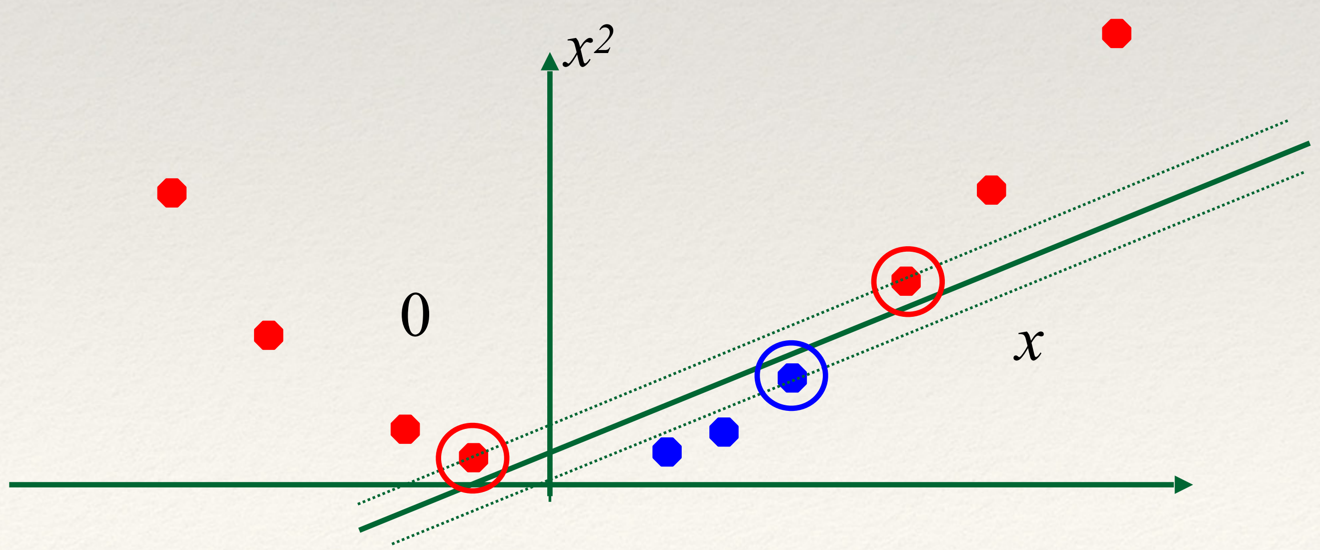
# Non-linear SVMs

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



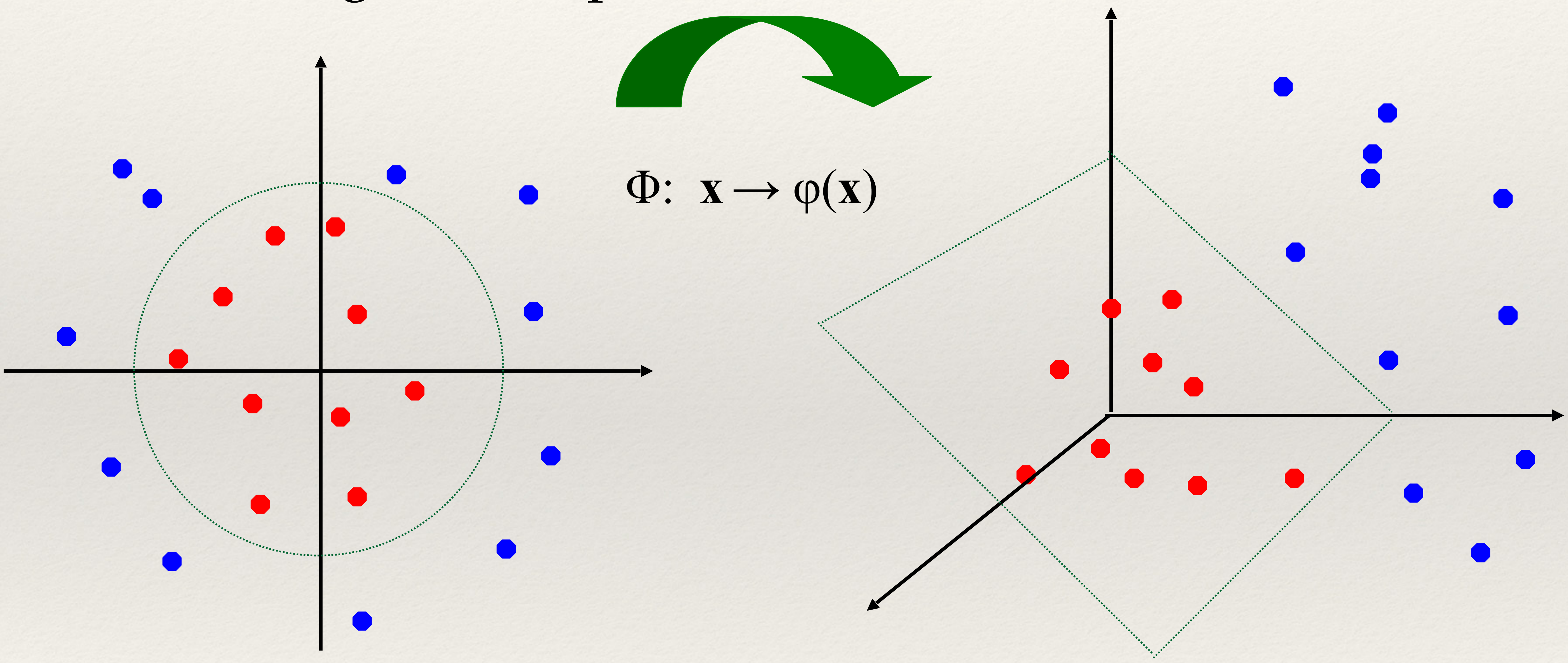

- But what are we going to do if the dataset is just too hard?


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



# Non-linear SVMs: Feature spaces

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



# The "Kernel Trick"

- The linear classifier relies on dot product between vectors $k(x_i,x_j)=\mathbf{x_i}^T\mathbf{x_j}$
- If every data point is mapped into high-dimensional space via some transformation $\Phi$: $\mathbf{x} \rightarrow \phi(\mathbf{x})$, the dot product becomes:

$$k(\mathbf{x_i},\mathbf{x_j})= \phi(\mathbf{x_i})^T\phi(\mathbf{x_j})$$

- 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 $k(\mathbf{x_i},\mathbf{x_j})=(1 + \mathbf{x_i}^T\mathbf{x_j})^2$,

  Need to show that $K(\mathbf{x_i},\mathbf{x_j})= \phi(\mathbf{x_i})^T\phi(\mathbf{x_j})$:

$$k(\mathbf{x_i},\mathbf{x_j})=(1 + \mathbf{x_i}^T\mathbf{x_j})^2,$$

$$= 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$

$$= [1\;\; x_{i1}^2\;\; \sqrt{2}\, x_{i1}x_{i2}\;\; x_{i2}^2\;\; \sqrt{2}x_{i1}\;\; \sqrt{2}x_{i2}]^T [1\;\; x_{j1}^2\;\; \sqrt{2}\, x_{j1}x_{j2}\;\; x_{j2}^2\;\; \sqrt{2}x_{j1}\;\; \sqrt{2}x_{j2}]$$

$$= \phi(\mathbf{x_i})^T\phi(\mathbf{x_j}), \quad \text{where } \phi(\mathbf{x}) = [1\;\; x_1^2\;\; \sqrt{2}\, x_1x_2\;\; x_2^2\;\; \sqrt{2}x_1\;\; \sqrt{2}x_2]$$

# What Functions are Kernels?

- For many functions $k(\mathbf{x}_i,\mathbf{x}_j)$ checking that

  $k(\mathbf{x}_i,\mathbf{x}_j)= \boldsymbol{\phi}(\mathbf{x}_i)^T\boldsymbol{\phi}(\mathbf{x}_j)$ can be cumbersome.
- Mercer's theorem: Every semi-positive definite symmetric function is a kernel
  - Semi-positive definite symmetric functions correspond to a semi-positive definite symmetric Gram matrix:

$\mathbf{K}$ =

| | | | | |
|---|---|---|---|---|
| $k(\mathbf{x}_1,\mathbf{x}_1)$ | $k(\mathbf{x}_1,\mathbf{x}_2)$ | $k(\mathbf{x}_1,\mathbf{x}_3)$ | … | $k(\mathbf{x}_1,\mathbf{x}_N)$ |
| $k(\mathbf{x}_2,\mathbf{x}_1)$ | $k(\mathbf{x}_2,\mathbf{x}_2)$ | $k(\mathbf{x}_2,\mathbf{x}_3)$ | | $k(\mathbf{x}_2,\mathbf{x}_N)$ |
| … | … | … | … | … |
| $k(\mathbf{x}_N,\mathbf{x}_1)$ | $k(\mathbf{x}_N,\mathbf{x}_2)$ | $k(\mathbf{x}_N,\mathbf{x}_3)$ | … | $k(\mathbf{x}_N,\mathbf{x}_N)$ |

# Examples of Kernel Functions

- Linear: $k(\mathbf{x_i},\mathbf{x_j})= \mathbf{x_i}^T\mathbf{x_j}$
- Polynomial of power $p$: $k(\mathbf{x_i},\mathbf{x_j})= (1+ \mathbf{x_i}^T\mathbf{x_j})^p$
- Gaussian (radial-basis function network):

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

- Sigmoid: $k(\mathbf{x_i},\mathbf{x_j})= \tanh(\beta_0\mathbf{x_i}^T\mathbf{x_j} + \beta_1)$

# SVM Formulation

- Goal - **1) Correctly classify all training data**

$$\left.\begin{array}{ll}\mathbf{w}^T\phi(\mathbf{x}_n)+b\geq 1 & if \quad t_n=+1\\ \mathbf{w}^T\phi(\mathbf{x}_n)+b\leq 1 & if \quad t_n=-1\end{array}\right\}$$

$$t_n(\mathbf{w}^T\phi(\mathbf{x}_n)+b)\geq 1$$

**2) Define the Margin**

$$\frac{1}{||\mathbf{w}||}min_n\ [t_n(\mathbf{w}^T\phi(\mathbf{x}_n)+b)]$$

**3) Maximize the Margin**

$$argmax_{\mathbf{w},b}\left\{\frac{1}{||\mathbf{w}||}min_n\ [t_n(\mathbf{w}^T\phi(\mathbf{x}_n)+b)]\right\}$$

- Equivalently written as

$$argmin_{\mathbf{w},b}\ \frac{1}{2}||\mathbf{w}||^2 \text{ such that } t_n(\mathbf{w}^T\phi(\mathbf{x}_n)+b)\geq 1$$

# Solving the Optimization Problem

- Need to optimize a *quadratic* function subject to *linear* constraints.
- Quadratic optimization problems are a well-known class of mathematical programming problems, and many (rather intricate) algorithms exist for solving them.
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* $a_n$ is associated with every constraint in the primary problem:
- The dual problem in this case is maximized

Find $\{a_1, .., a_N\}$ such that

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} t_n t_m a_n a_m k(\mathbf{x}_n, \mathbf{x}_m) \text{ maximized}$$

and $\sum_n a_n t_n = 0$, $a_n \geq 0$

# Solving the Optimization Problem

- The solution has the form:

$$\mathbf{w} = \sum_{n=1}^{N} a_n \phi(\mathbf{x}_n)$$

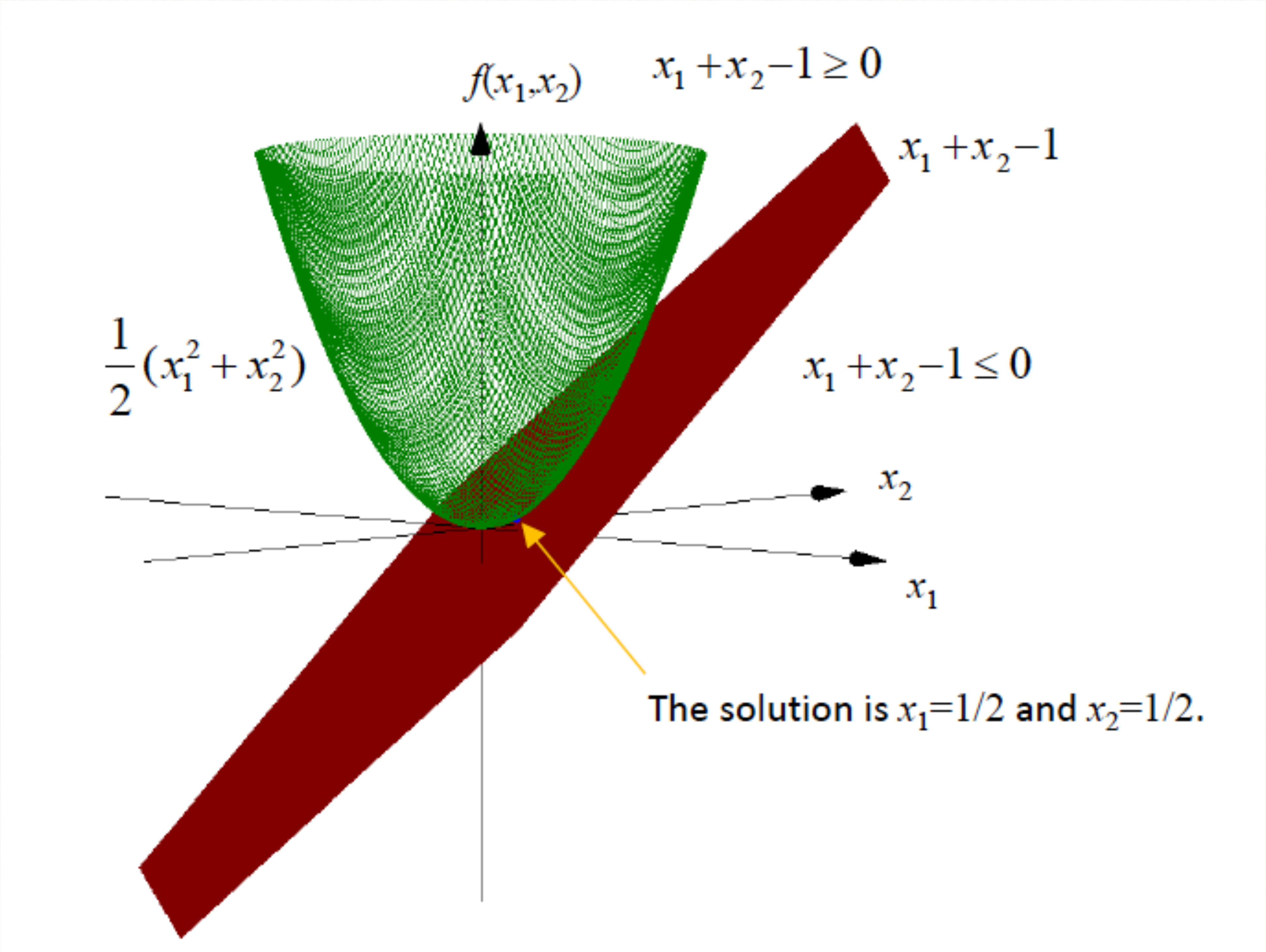
- Each non-zero $a_n$ indicates that corresponding $\mathbf{x_n}$ is a support vector. Let S denote the set of support vectors.

$$b = y(\mathbf{x}_n) - \sum_{m \in S} a_m k(\mathbf{x}_m, \mathbf{x}_n)$$

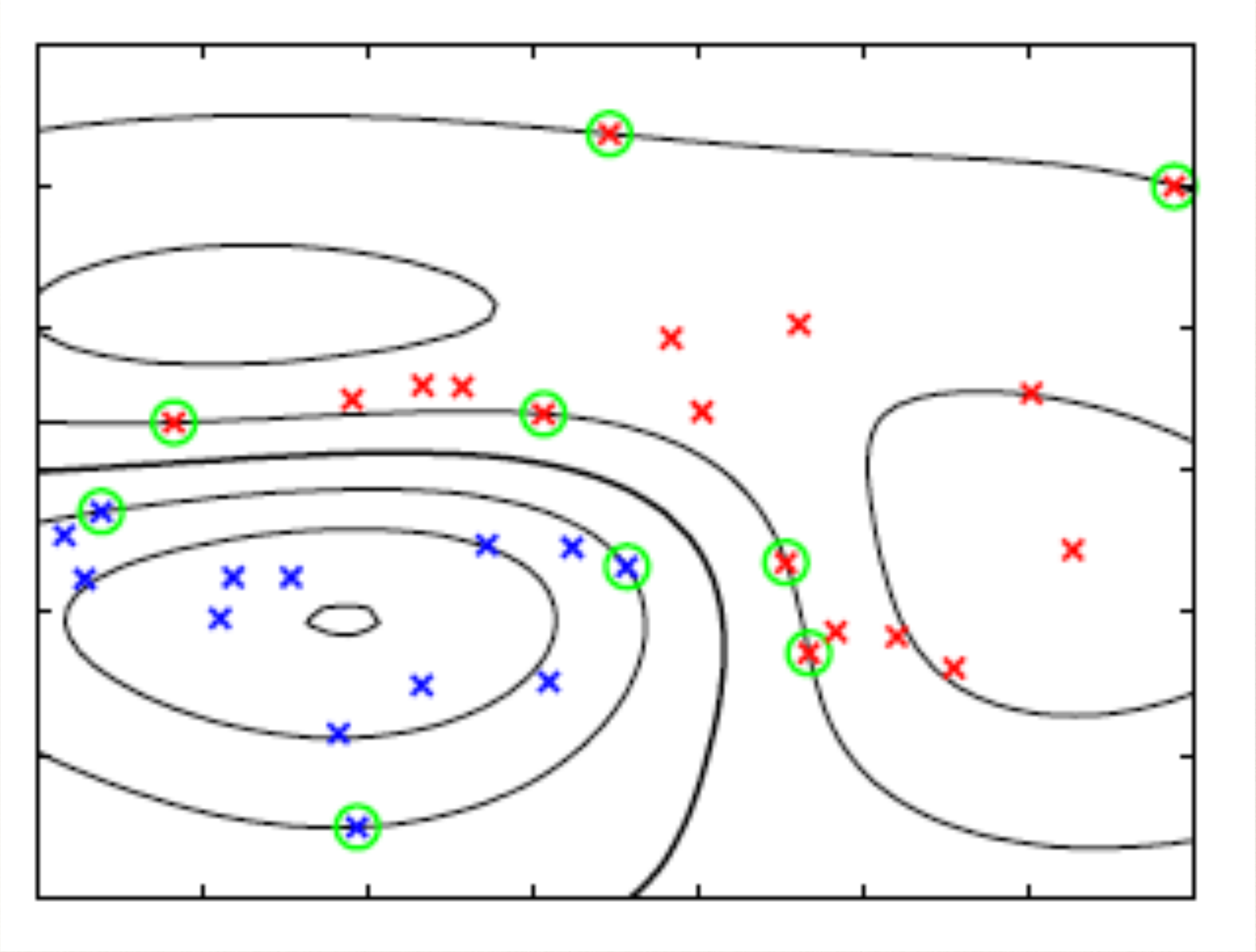
- And the classifying function will have the form:

$$y(\mathbf{x}) = \sum_{n \in S} a_n k(\mathbf{x}_n, \mathbf{x}) + b$$

# Solving the Optimization Problem



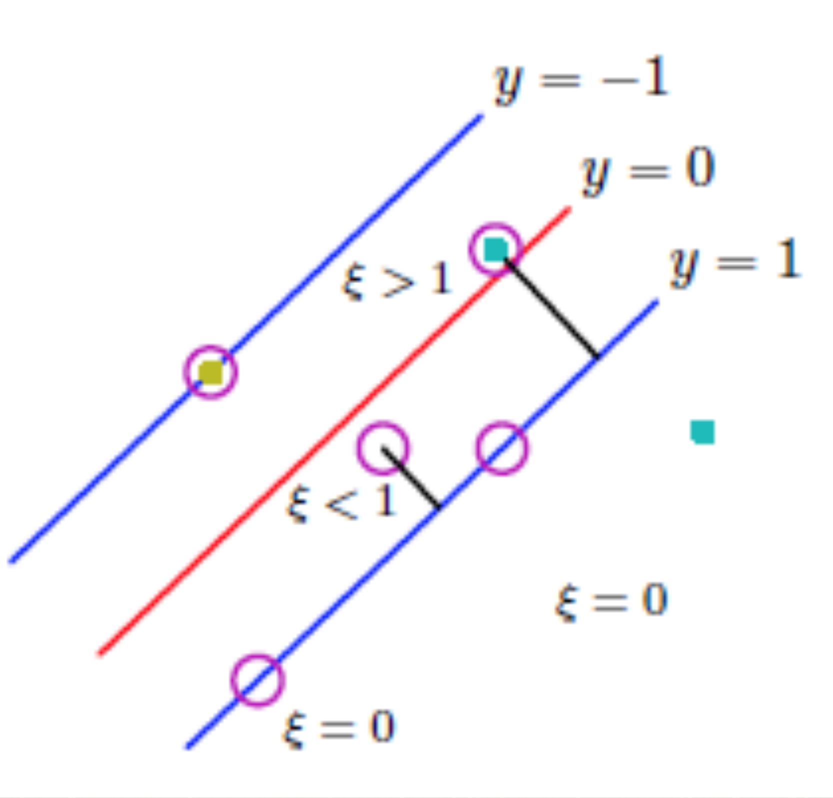
# Visualizing Gaussian Kernel SVM



# Overlapping class boundaries

- The classes are not linearly separable - Introducing slack variables $\zeta_n$
- Slack variables are non-negative $\zeta_n \geq 0$
- They are defined using

$$t_n y(\mathbf{x}_n) \geq 1 - \zeta_n$$




- The upper bound on mis-classification

$$\sum_n \zeta_n$$

- The cost function to be optimized in this case

$$C \sum_n \zeta_n + \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

# SVM Formulation - overlapping classes

- Formulation very similar to previous case except for additional constraints

$$0 \leq a_n \leq C$$

- Solved using the dual formulation - sequential minimal optimization algorithm
- Final classifier is based on the sign of

$$y(\mathbf{x}) = \sum_{n \in S} a_n k(\mathbf{x}_n, \mathbf{x}) + b$$

# Overlapping class boundaries