

# **SUPPORT VECTOR MACHINE**

Mainly based on

<https://nlp.stanford.edu/IR-book/pdf/15svm.pdf>

# Overview

---

- SVM is a huge topic
  - Integration of MMDS, IIR, and Andrew Moore's slides here
- Our foci:
  - Geometric intuition → Primal form
    - Alternative interpretation from Empirical Risk Minimization point of view.
  - Understand the final solution (in the dual form)
    - Generalizes well to Kernel functions
  - SVM + Kernels

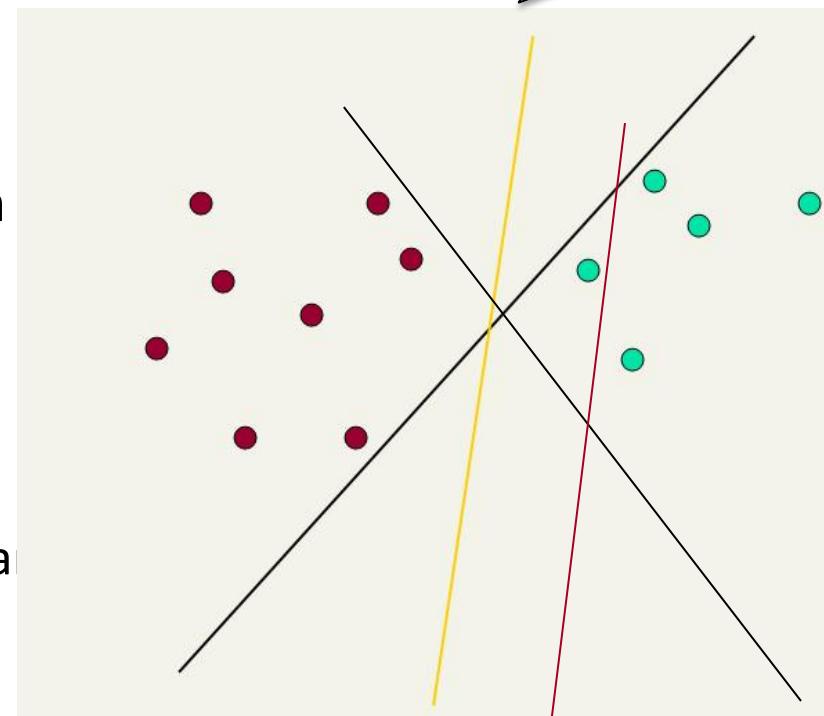
$$\mathbf{w}^T \mathbf{x}_i + b = 0$$

# Linear classifiers: Which Hyperplane?



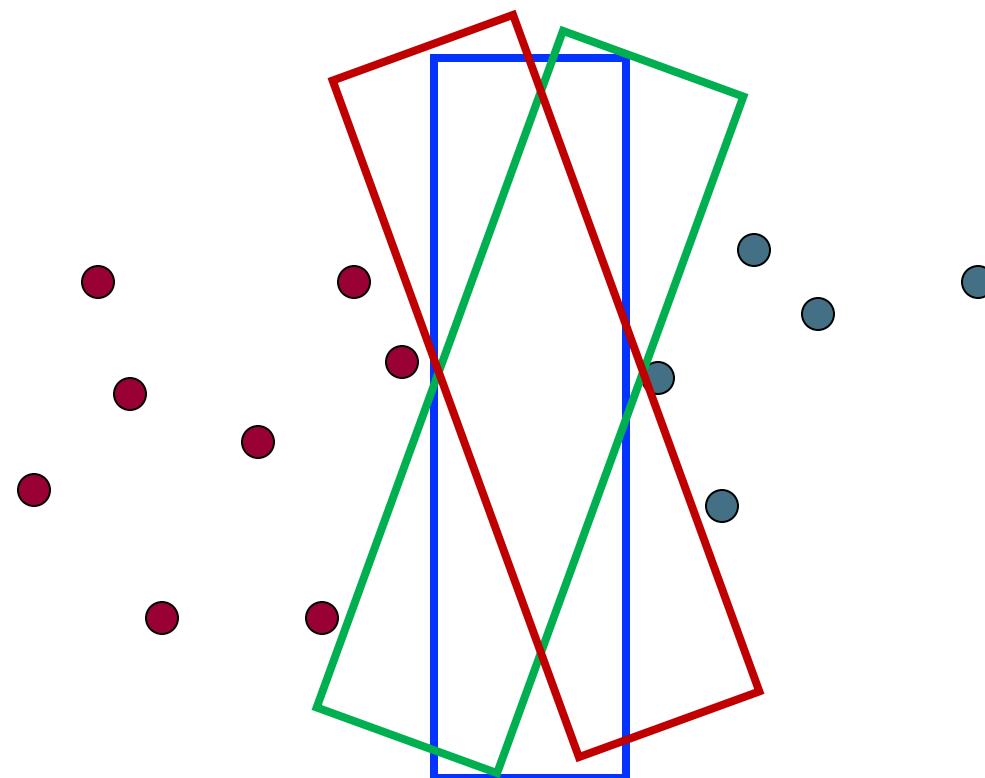
- Lots of possible solutions for  $a, b, c$ .
- Some methods find a separating hyperplane, but not the optimal one [according to some criterion of expected goodness]
  - E.g., perceptron
- Support Vector Machine (SVM) finds an optimal\* solution.
  - Maximizes the distance between the hyperplane and the “difficult points” close to decision boundary
  - One intuition: if there are no points near the decision surface, then there are no very uncertain classification decisions

This line represents the decision boundary:  
 $ax + by - c = 0$



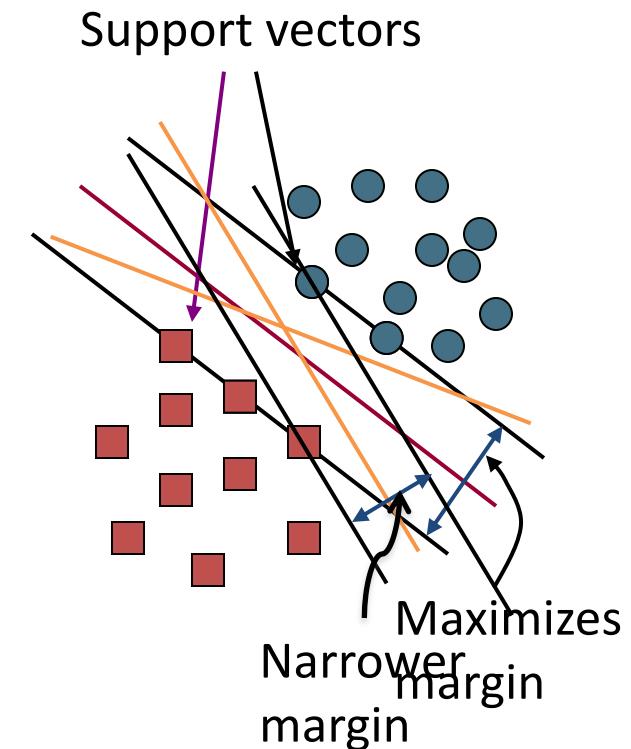
# Another intuition

- If you have to place a fat separator between classes, you have less choices, and so the capacity of the model has been decreased



# Support Vector Machine (SVM)

- SVMs maximize the *margin* around the separating hyperplane.
  - A.k.a. large margin classifiers
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- Solving SVMs is a *quadratic programming* problem
- Seen by many as the most successful current text classification method\*



\*but other discriminative methods often perform very similarly

# Maximum Margin: Formalization

- $\mathbf{w}$ : decision hyperplane normal vector
- $\mathbf{x}_i$ : data point  $i$
- $y_i$ : class of data point  $i$  (+1 or -1)
- Classifier is:  $f(\mathbf{x}_i) = \text{sign}(\mathbf{w}^T \mathbf{x}_i + b)$
- **Functional margin** of  $\mathbf{x}_i$  is:  $y_i (\mathbf{w}^T \mathbf{x}_i + b)$ 
  - But note that we can increase this margin simply by scaling  $\mathbf{w}$ ,  $b$ ....
- Functional margin of dataset is twice the minimum functional margin for any point
  - The factor of 2 comes from measuring the whole width of the margin

NB: Not 1/0

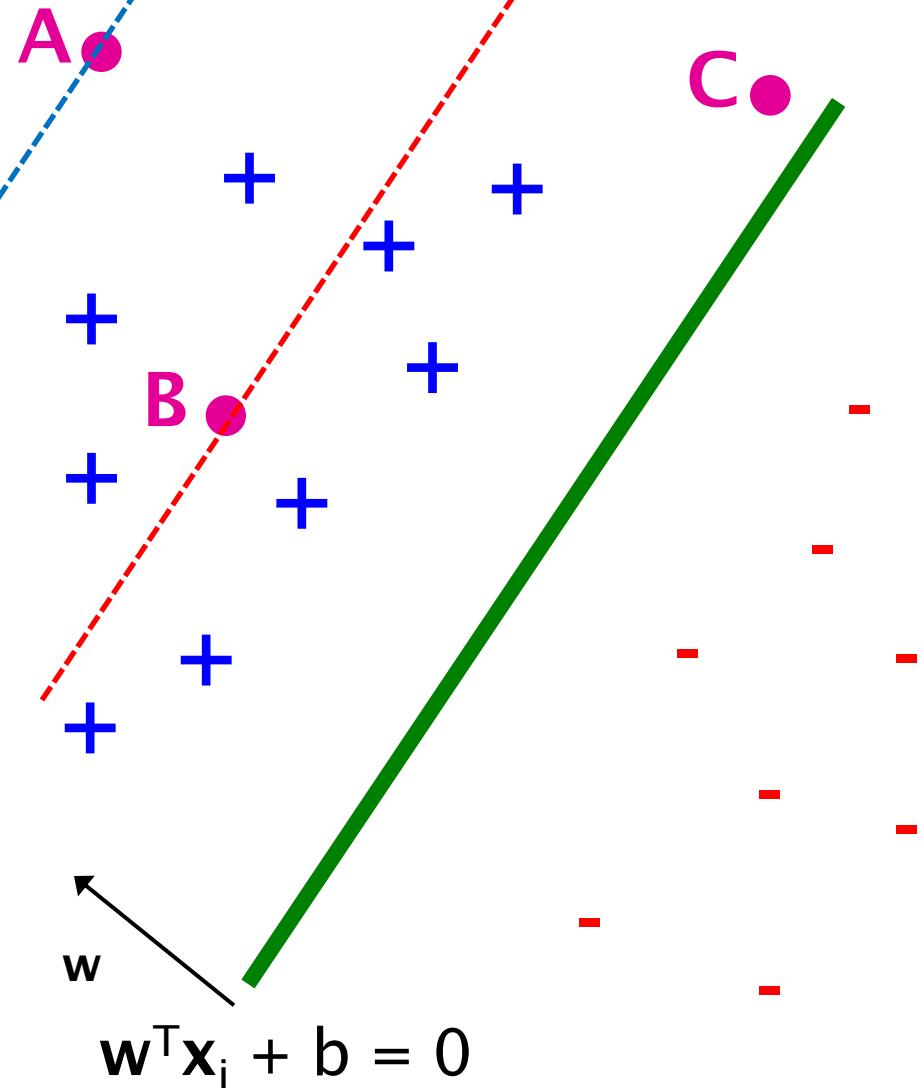


NB: a common  
trick

$$\mathbf{w}^T \mathbf{x}_i + b = 7.4$$

$$\mathbf{w}^T \mathbf{x}_i + b = 3.7$$

## Largest Margin

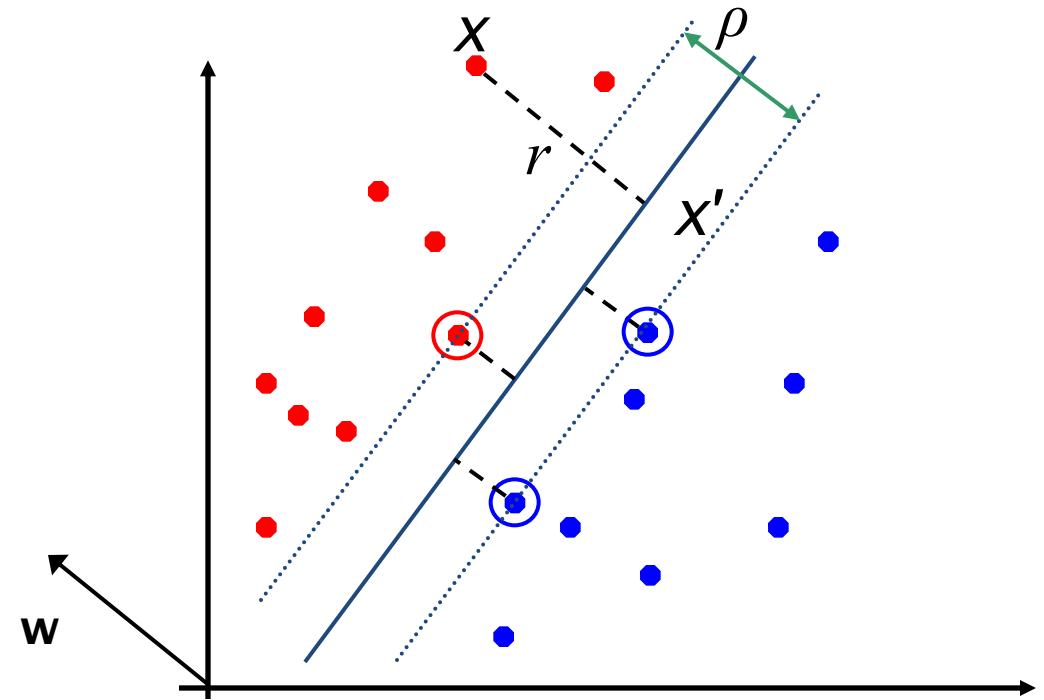


- Distance from the separating hyperplane corresponds to the “confidence” of prediction
- Example:
  - We are more sure about the class of A and B than of C

# Geometric Margin

- Distance from example to the separator is
- Examples closest to the hyperplane are ***support vectors***.
- Margin  $\rho$**  of the separator is the width of separation between support vectors of classes.

$$r = y \frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|}$$



## Algebraic derivation of finding $r$ :

Dotted line  $\mathbf{x}' - \mathbf{x}$  is perpendicular to decision boundary so parallel to  $\mathbf{w}$ . Unit vector is  $\mathbf{w}/|\mathbf{w}|$ , so line is  $r\mathbf{w}/|\mathbf{w}|$ , for some  $r$ .

$$\mathbf{x}' = \mathbf{x} - yr\mathbf{w}/|\mathbf{w}|.$$

$$\mathbf{x}' \text{ satisfies } \mathbf{w}^T \mathbf{x}' + b = 0.$$

$$\text{So } \mathbf{w}^T(\mathbf{x} - yr\mathbf{w}/|\mathbf{w}|) + b = 0$$

$$\text{Recall that } |\mathbf{w}| = \sqrt{\mathbf{w}^T \mathbf{w}}.$$

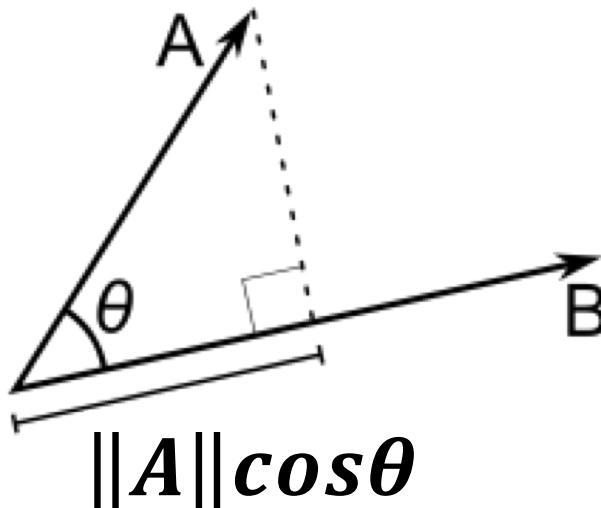
$$\text{So } \mathbf{w}^T \mathbf{x} - yr|\mathbf{w}| + b = 0$$

So, solving for  $r$  gives:

$$r = y(\mathbf{w}^T \mathbf{x} + b)/|\mathbf{w}|$$

# Help from Inner Product

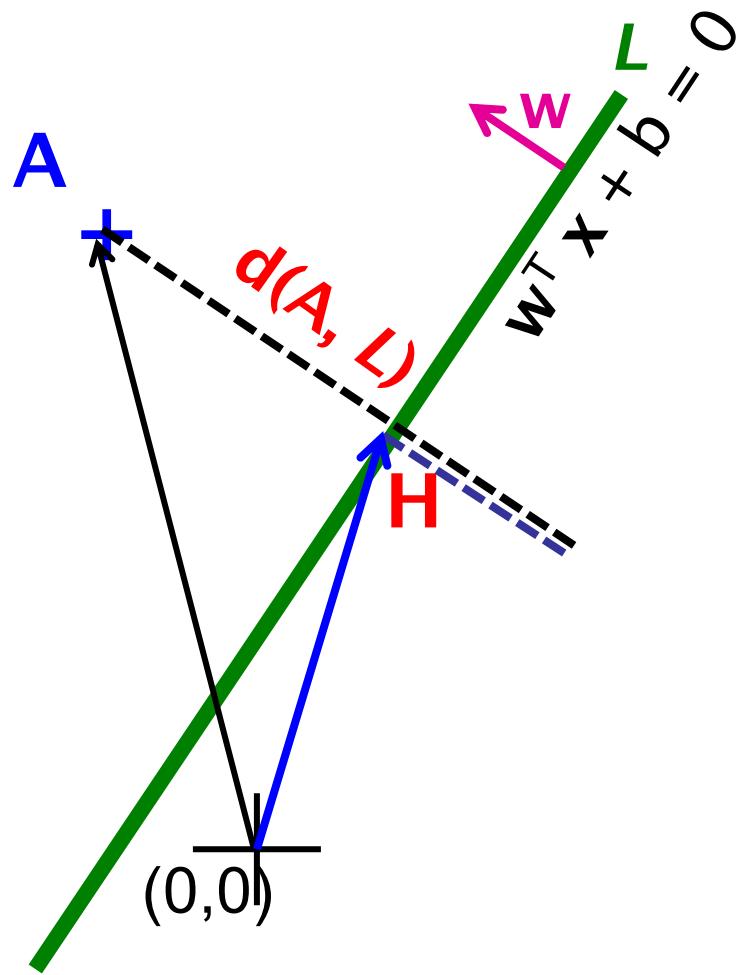
- Remember: Dot product / inner product



$$\langle A, B \rangle = \|A\| \|B\| \cos \theta$$

- (Or use an algebraic derivation similar to prev slide)
- A's projection onto B =  $\langle A, B \rangle / \langle B, B \rangle$

# Derivation of the Geometric Margin



- $d(A, L) = \langle A, w \rangle / \langle w, w \rangle$   
-  $\langle H, w \rangle / \langle w, w \rangle$
- Note that  $H$  is on the line  $L$ , so  
 $\langle w, H \rangle + b = 0$
- Therefore  $= (\langle A, w \rangle + b) / \langle w, w \rangle$

# Linear SVM Mathematically

## The linearly separable case

---

- Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow for a training set  $\{(\mathbf{x}_i, y_i)\}$

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \quad \text{if } y_i = 1$$

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1 \quad \text{if } y_i = -1$$

- For support vectors, the inequality becomes an equality
- Then, since each example's distance from the hyperplane is

$$r = y \frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|}$$

- The margin is:

$$\rho = \frac{2}{\|\mathbf{w}\|}$$

# Derivation of $\rho$

- Hyperplane

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

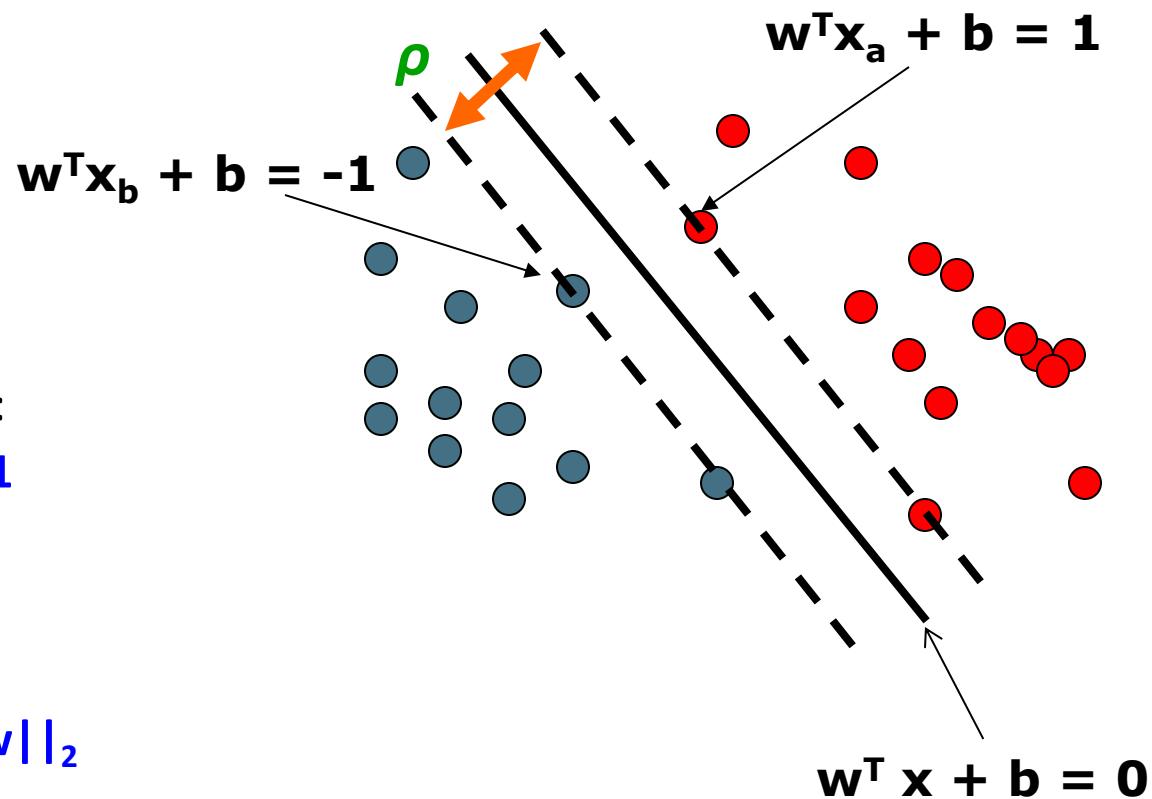
- Extra scale constraint:

$$\min_{i=1,\dots,n} |\mathbf{w}^T \mathbf{x}_i + b| = 1$$

- This implies:

$$\mathbf{w}^T (\mathbf{x}_a - \mathbf{x}_b) = 2$$

$$\rho = \| \mathbf{x}_a - \mathbf{x}_b \|_2 = 2 / \| \mathbf{w} \|_2$$



# Solving the Optimization Problem

Find  $\mathbf{w}$  and  $b$  such that

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

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

- This is now optimizing a *quadratic* function subject to *linear* constraints
- Quadratic optimization problems are a well-known class of mathematical programming problem, and many (intricate) algorithms exist for solving them (with many special ones built for SVMs)
- The solution involves constructing a *dual problem* where a *Lagrange multiplier*  $\alpha_i$  is associated with every constraint in the primary problem:

Find  $\alpha_1 \dots \alpha_N$  such that

$Q(\boldsymbol{\alpha}) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$  is maximized and

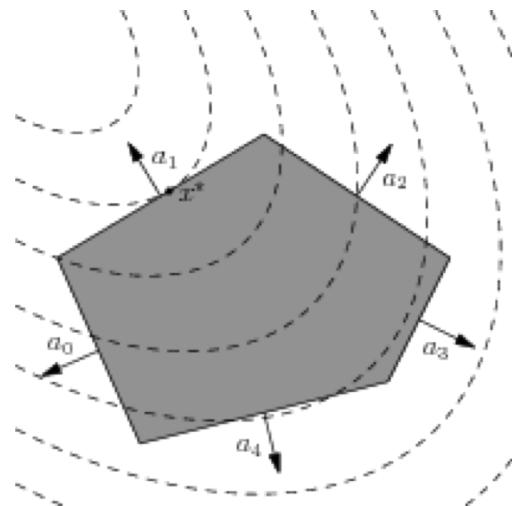
$$(1) \quad \sum \alpha_i y_i = 0$$

$$(2) \quad \alpha_i \geq 0 \text{ for all } \alpha_i$$

# Geometric Interpretation

Find  $\mathbf{w}$  and  $b$  such that  
 $\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$  is minimized;  
and for all  $\{(\mathbf{x}_i, y_i)\}$ :  $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

- What are fixed and what are variables?
- **Linear** constraint(s): Where can the variables be?
- **Quadratic** objective function:



# The Optimization Problem Solution

- The solution has the form:

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i \quad b = y_k - \mathbf{w}^T \mathbf{x}_k \text{ for any } \mathbf{x}_k \text{ such that } \alpha_k \neq 0$$

- Each non-zero  $\alpha_i$ , indicates that corresponding  $\mathbf{x}_i$  is a support vector.**
- Then the scoring function will have the form:

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

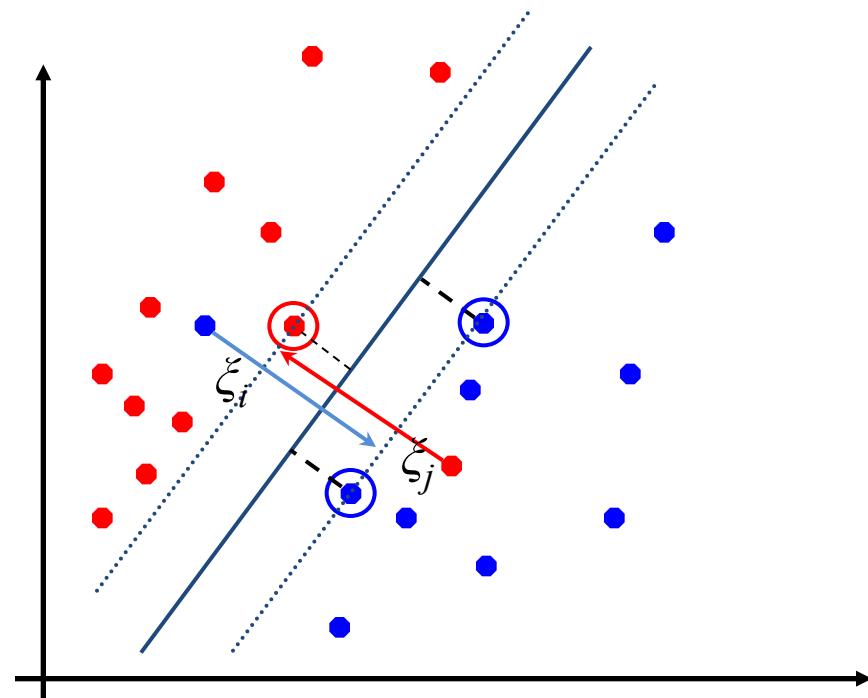
$$f(\mathbf{x}) = \sum_{sv \in SV} y_{sv} \cdot \alpha_{sv} \cdot \langle \mathbf{x}_{sv}, \mathbf{x} \rangle + b$$

*Q: What are the model parameters? What does  $f(x)$  mean intuitively?*

- Classification is based on the sign of  $f(\mathbf{x})$
- Notice that it relies on an **inner product** between the test point  $\mathbf{x}$  and the support vectors  $\mathbf{x}_i$ 
  - We will return to this later.
- Also *keep in mind* that solving the optimization problem involved computing the inner products  $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$  between all pairs of training points.

# Soft Margin Classification

- If the training data is not linearly separable, *slack variables*  $\xi_i$  can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
  - Let some points be moved to where they belong, at a cost
- Still, try to minimize training set errors, and to place hyperplane “far” from each class (large margin)



# Soft Margin Classification

[Optional]

## Mathematically

---

- The old formulation:

Find  $\mathbf{w}$  and  $b$  such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$
$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

- The new formulation incorporating slack variables:

Find  $\mathbf{w}$  and  $b$  such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_i \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$
$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \text{ for all } i$$

- Parameter  $C$  can be viewed as a way to control overfitting
  - A regularization term

# Alternative View

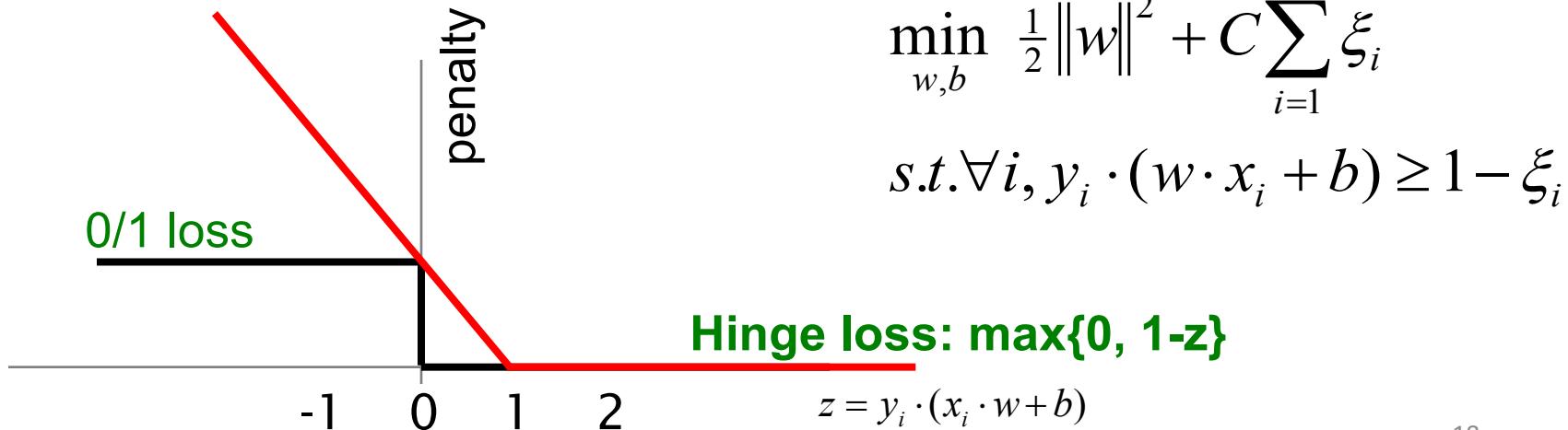
- SVM in the “natural” form

$$\arg \min_{w,b} \underbrace{\frac{1}{2} w \cdot w}_{\text{Margin}} + C \cdot \sum_{i=1}^n \max \{0, 1 - y_i (w \cdot x_i + b)\}$$

↑  
Empirical loss  $L$  (how well we fit training data)

Hyper-parameter related to regularization

- SVM uses “Hinge Loss”:



**[Optional]**

# Soft Margin Classification – Solution

- The dual problem for soft margin classification:

Find  $\alpha_1 \dots \alpha_N$  such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$  is maximized and

- (1)  $\sum \alpha_i y_i = 0$
- (2)  $0 \leq \alpha_i \leq C$  for all  $\alpha_i$

- Neither slack variables  $\xi_i$ , nor their Lagrange multipliers appear in the dual problem!
- Again,  $\mathbf{x}_i$  with non-zero  $\alpha_i$  will be support vectors.
- Solution to the dual problem is:

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

$$b = y_k(1 - \xi_k) - \mathbf{w}^T \mathbf{x}_k \text{ where } k = \operatorname{argmax}_{k'} \alpha_{k'}$$

w is not needed explicitly for classification!

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

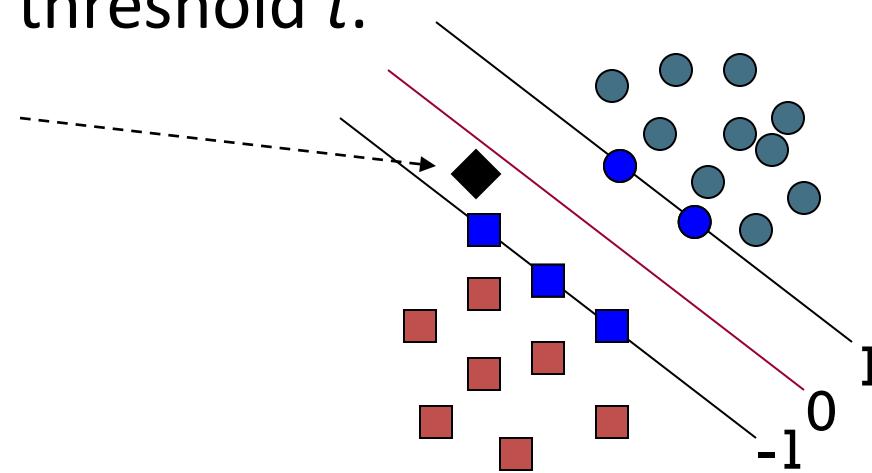
# Classification with SVMs

- Given a new point  $\mathbf{x}$ , we can score its projection onto the hyperplane normal:
  - i.e., compute score:  $\mathbf{w}^T \mathbf{x} + b = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$ 
    - Decide class based on whether  $<$  or  $> 0$
  - Can set confidence threshold  $t$ .

Score  $> t$ : yes

Score  $< -t$ : no

Else: don't know



# Linear SVMs: Summary

- The classifier is a *separating hyperplane*.
- The most “important” training points are the 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  $\alpha_i$ .
- Both in the dual formulation of the problem and in the solution, training points appear only inside inner products:

Find  $\alpha_1 \dots \alpha_N$  such that

$\mathbf{Q}(\boldsymbol{\alpha}) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$  is maximized and

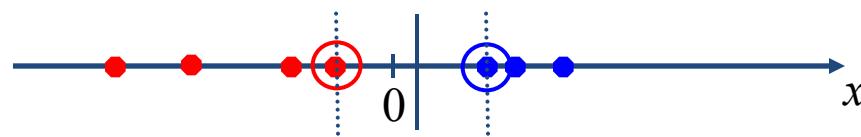
$$(1) \sum \alpha_i y_i = 0$$

$$(2) 0 \leq \alpha_i \leq C \text{ for all } \alpha_i$$

$$f(\mathbf{x}) = \sum_{sv \in SV} y_{sv} \cdot \alpha_{sv} \cdot \langle \mathbf{x}_{sv}, \mathbf{x} \rangle + b$$

# Non-linear SVMs

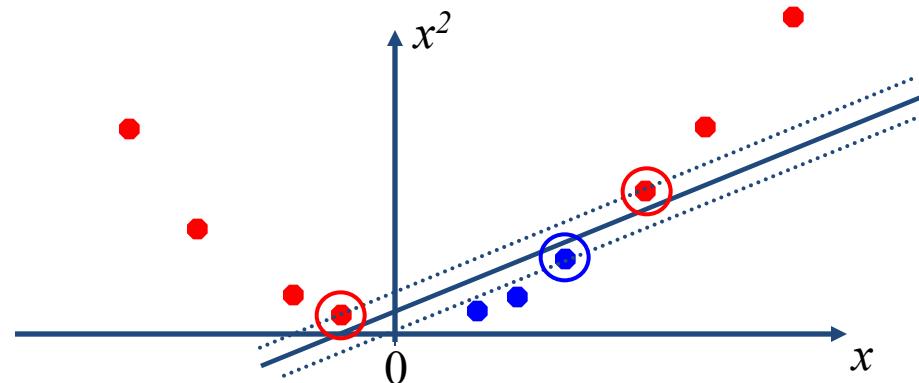
- Datasets that are linearly separable (with some noise) work out great:



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

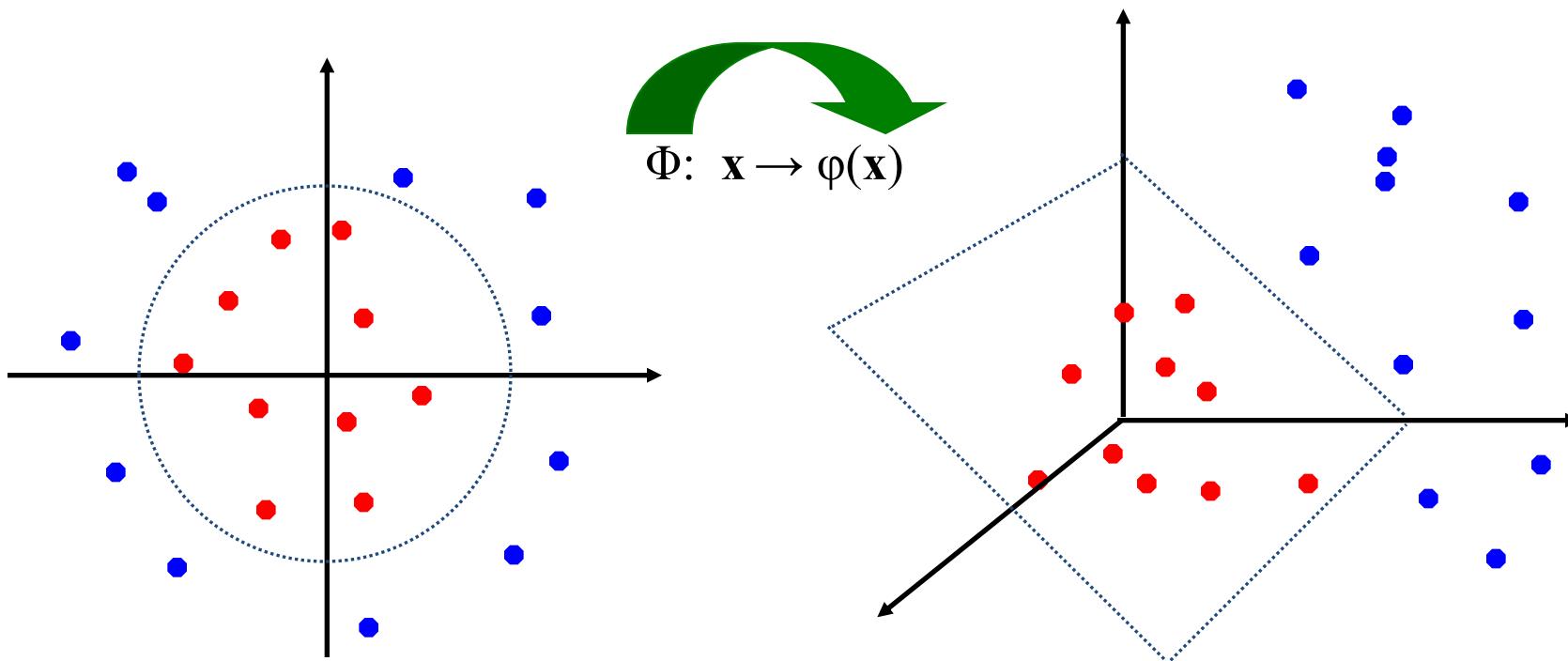


- How about ... mapping data to a higher-dimensional space:



# Non-linear SVMs: Feature spaces

- General idea: the original feature 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 an inner product between vectors  
$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$$
- If every datapoint is mapped into high-dimensional space via some transformation  $\Phi$ :  $\mathbf{x} \rightarrow \phi(\mathbf{x})$ , the inner 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.
  - Usually, no need to construct the feature space explicitly.

**What about**  $K(\mathbf{u}, \mathbf{v}) = (1 + \mathbf{u}^\top \mathbf{v})^3$  ?

# Example

$$\begin{aligned} K(\mathbf{u}, \mathbf{v}) &= (1 + \mathbf{u}^\top \mathbf{v})^2 \\ &= 1 + 2\mathbf{u}^\top \mathbf{v} + (\mathbf{u}^\top \mathbf{v})^2 \\ &= 1 + 2\mathbf{u}^\top \mathbf{v} + \left( \sum_i \mathbf{u}_i \mathbf{v}_i \right)^2 \end{aligned}$$

$$\begin{aligned} &= 1 + 2\mathbf{u}^\top \mathbf{v} + \left( \sum_i \mathbf{u}_i^2 \mathbf{v}_i^2 + \sum_{i \neq j} \mathbf{u}_i \mathbf{u}_j \mathbf{v}_i \mathbf{v}_j \right) \\ &= \phi(\mathbf{u})^\top \phi(\mathbf{v}) \end{aligned}$$

**O(d<sup>2</sup>) new cross-term features**

$$\phi(\mathbf{u}) = [1 \quad \sqrt{2}\mathbf{u}_1 \quad \dots \quad \sqrt{2}\mathbf{u}_d \quad \mathbf{u}_1^2 \quad \dots \quad \mathbf{u}_d^2 \quad \mathbf{u}_1 \mathbf{u}_2 \quad \dots \quad \mathbf{u}_{d-1} \mathbf{u}_d]^\top$$

$$\phi(\mathbf{v}) = [1 \quad \sqrt{2}\mathbf{v}_1 \quad \dots \quad \sqrt{2}\mathbf{v}_d \quad \mathbf{v}_1^2 \quad \dots \quad \mathbf{v}_d^2 \quad \mathbf{v}_1 \mathbf{v}_2 \quad \dots \quad \mathbf{v}_{d-1} \mathbf{u}_d]^\top$$

Linear

Non-linear

Non-linear + feature combination

# Why feature combinations?

---

- Examples:
  - Two categorical features (age & married) encoded as one-hot encoding → combination = conjunction rules
    - e.g.,  $1[\text{age in } [30, 40] \text{ AND married} = \text{FALSE}]$
  - [..., eagerness-for-travel, income, ...] → combination indicates how much to spend on travel
  - NLP, feature vector =  $1[w \in x]$  → combination indicates two word co-occurrence (where phrase/multi-word expression (MWE) is just a special case)
- $\mathbf{x} \rightarrow \varphi(\mathbf{x})$ , then a linear model in the new feature space is just  $\mathbf{w}^T \varphi(\mathbf{x}) + b$ 
  - each feature combination will be assigned a weight  $w_i$
  - irrelevant features combinations will get 0 weight

# Inner product in an **infinite** dimensional space!

[Optional]

RBF kernel:

$$e^{-\gamma \|x_i - x_j\|^2} = e^{-\gamma(x_i - x_j)^2} = e^{-\gamma x_i^2 + 2\gamma x_i x_j - \gamma x_j^2}$$

$$= e^{-\gamma x_i^2 - \gamma x_j^2} \left( 1 + \frac{2\gamma x_i x_j}{1!} + \frac{(2\gamma x_i x_j)^2}{2!} + \frac{(2\gamma x_i x_j)^3}{3!} + \dots \right)$$

$$= e^{-\gamma x_i^2 - \gamma x_j^2} \left( 1 \cdot 1 + \sqrt{\frac{2\gamma}{1!}} x_i \cdot \sqrt{\frac{2\gamma}{1!}} x_j + \sqrt{\frac{(2\gamma)^2}{2!}} x_i^2 \cdot \sqrt{\frac{(2\gamma)^2}{2!}} x_j^2 + \dots \right)$$

$$= \phi(x_i)^T \phi(x_j) \quad , \text{ where}$$

$$\phi(x) = e^{-\gamma x^2} \left[ 1, \sqrt{\frac{2\gamma}{1!}} x, \sqrt{\frac{(2\gamma)^2}{2!}} x^2, \sqrt{\frac{(2\gamma)^3}{3!}} x^3, \dots \right]^T$$

# String Kernel

---

- $K(s_1, s_2)$  should evaluate the similarity between the two strings
  - Without this,  $\text{sim}(\text{"actor"}, \text{"actress"}) = 0$
- Intuition:
  - consider all **substrings** as (binary) “features”
  - inner product in that “enhanced” feature space means the number of common substrings the two share.
- Variants:
  - (more complex): consider subsequences (with possibly gap penalty)
  - (simpler): consider all k-grams, and use Jaccard
    - $\text{bigrams(actor)} = \{\text{ac, ct, to, or}\}$
    - $\text{bigrams(actress)} = \{\text{ac, ct, tr, re, es, ss}\}$
    - $\text{Jaccard(actor, actress)} = 2/8$

# Kernels

- Why use kernels?

- Make non-separable problem separable.
  - Map data into better representational space
  - Can be learned to capture some notion of “similarity”

- Common kernels

- Linear
  - Polynomial  $K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^T \mathbf{z})^d$ 
    - Gives feature combinations
  - Radial basis function (infinite dimensional space)

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

- Text classification:

- Usually use linear or polynomial kernels

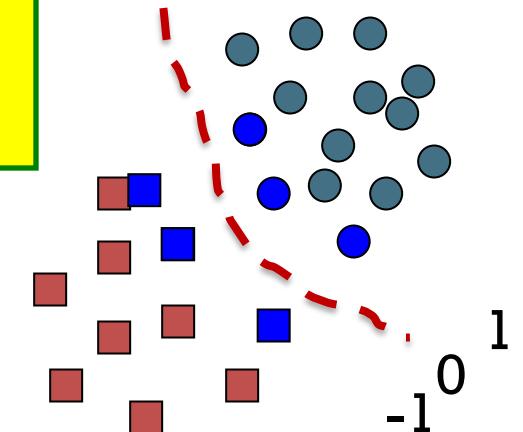
# Classification with SVMs **with Kernels**

- Given a new point  $\mathbf{x}$ , we can compute its score

i.e.,

$$\sum_{sv \in SV} \alpha_{sv} K(\mathbf{x}_{sv}, \mathbf{x}) + b$$

- Decide class based on whether  $<$  or  $> 0$
- E.g., in scikit-learn



- linear:  $\langle x, x' \rangle$ .
- polynomial:  $(\gamma \langle x, x' \rangle + r)^d$ .  $d$  is specified by keyword `degree`,  $r$  by `coef0`.
- rbf:  $\exp(-\gamma \|x - x'\|^2)$ .  $\gamma$  is specified by keyword `gamma`, must be greater than 0.
- sigmoid ( $\tanh(\gamma \langle x, x' \rangle + r)$ ), where  $r$  is specified by `coef0`.

# Pros and Cons of the SVM Classifier

---

- Pros
  - High accuracy
  - Fast classification speed
  - Works with cases where #features > #samples
  - Can adapt to different objects (due to Kernel)
    - Any  $K(u, v)$  can be used if symmetric, continuous, and positive semi-definite.
    - Or explicitly engineer the feature space.
- Cons
  - Training does not scale well with number of training samples ( $O(d*n^2)$  or  $O(d*n^3)$ )
  - Hyper-parameters need tuning

# Resources for today's lecture

---

- Christopher J. C. Burges. 1998. A Tutorial on Support Vector Machines for Pattern Recognition
- S. T. Dumais. 1998. Using SVMs for text categorization, IEEE Intelligent Systems, 13(4)
- S. T. Dumais, J. Platt, D. Heckerman and M. Sahami. 1998. Inductive learning algorithms and representations for text categorization. *CIKM '98*, pp. 148-155.
- Yiming Yang, Xin Liu. 1999. A re-examination of text categorization methods. 22nd Annual International SIGIR
- Tong Zhang, Frank J. Oles. 2001. Text Categorization Based on Regularized Linear Classification Methods. *Information Retrieval* 4(1): 5-31
- Trevor Hastie, Robert Tibshirani and Jerome Friedman. *Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer-Verlag, New York.
- T. Joachims, *Learning to Classify Text using Support Vector Machines*. Kluwer, 2002.
- Fan Li, Yiming Yang. 2003. A Loss Function Analysis for Classification Methods in Text Categorization. *ICML* 2003: 472-479.
- Tie-Yan Liu, Yiming Yang, Hao Wan, et al. 2005. Support Vector Machines Classification with Very Large Scale Taxonomy, *SIGKDD Explorations*, 7(1): 36-43.
- ‘Classic’ Reuters-21578 data set: <http://www.daviddlewis.com/resources/testcollections/reuters21578/>