Classification: Up until now

- Classification with ANNs can be computationally expensive
- Hard to tune (number of layers, number of neurons, etc.)
- Prone to over-fitting (lower training error but higher test error)
- Local minimum
- Initialisation sensitive
- Any more?

- To deal with these issues we introduce

Support Vector Machine (SVM)

1: Support Vector Machines

- Background
- Linear SVM
- Soft-margin classifier
- Non-linear SVM and the kernel trick

Support Vector Machines

- Perceptron Neural Systems
  - Biologically motivated data-driven optimal (?) classifiers.
  - Massive parallel system
  - …but are slow when implemented and run on single CPU systems.
- Support Vector Machines
  - Data-driven optimal classifiers.
  - Same computational abilities as MLP.
  - Could be much faster on single CPU systems

Terminology: “Support Vectors”

We have already learned:

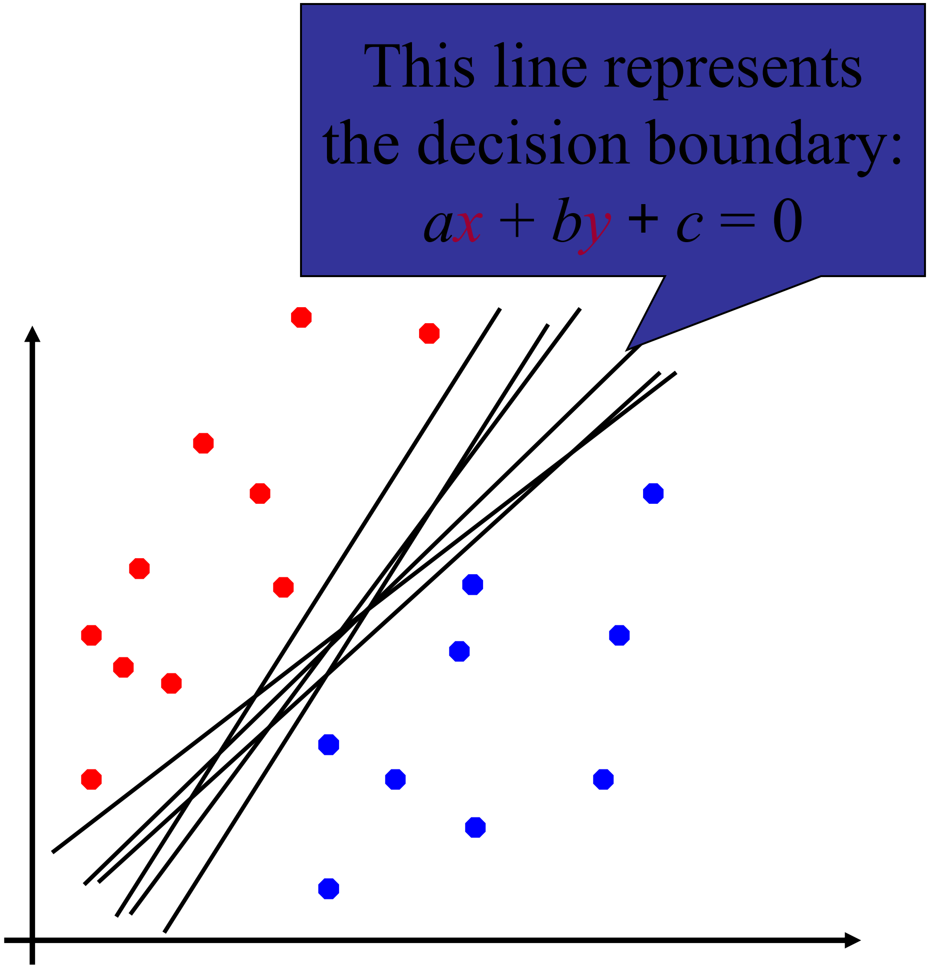
- An MLP can find an optimal decision boundary for a given set of (training-)samples for which target values are available.
- Training data is available in the form of multi-dimensional fixed sized **vectors**.
- The training algorithm is entirely driven by the training data.
- Thus, one could say that the vectors in the training set **support** the MLP in finding the decision boundary.
- Note that the decision boundary can also be found if we only have had the data which are closest to the decision boundary.

In other words:

- The **minimum sub-set of the training data** needed to find the decision boundary are called the **support vectors.**

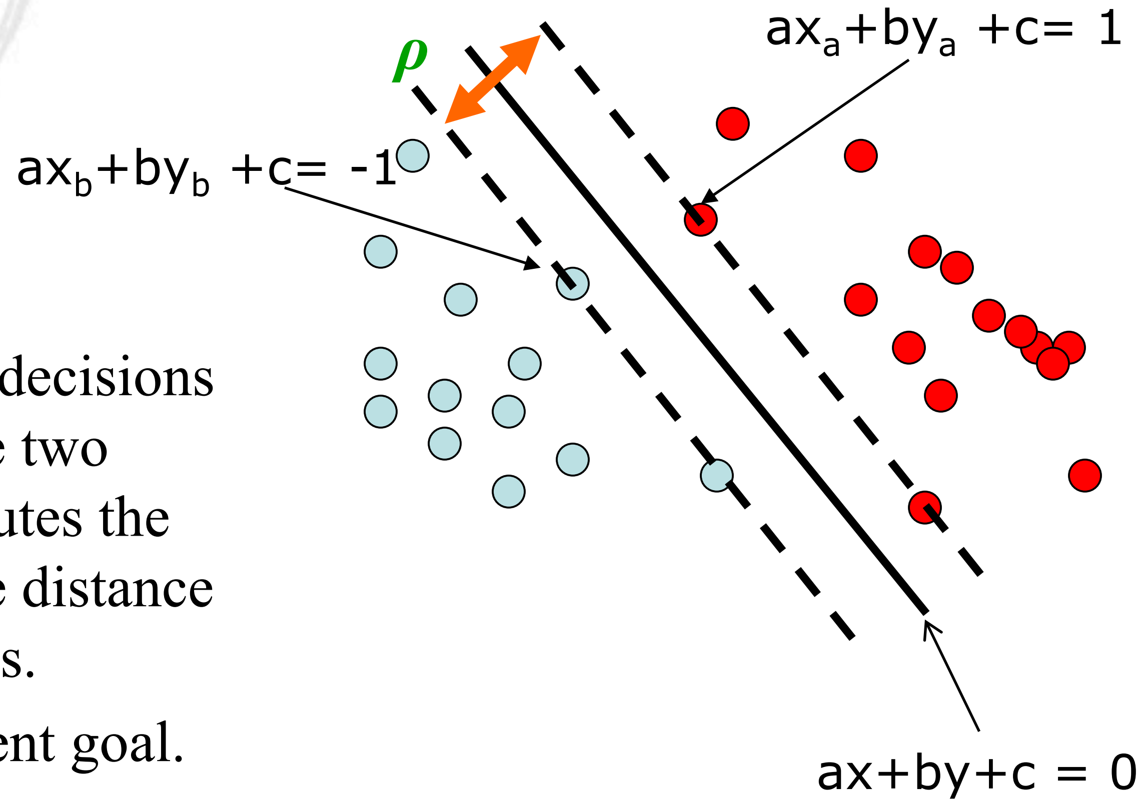
SVM - background

- Example: 2-dimensional, binary, linearly separable classification problems.
- Decision boundary is a line ax+by+c.
- Task is to find a, b, and c that separates the two classes of data.
- There may be infinite many a,b,c that meet the requirement.
- **Which one is best from your point of view**?



SVM – the aim

- SVM aims at **maximizing** the distance between the decision boundary and the "difficult points" close to decision boundary



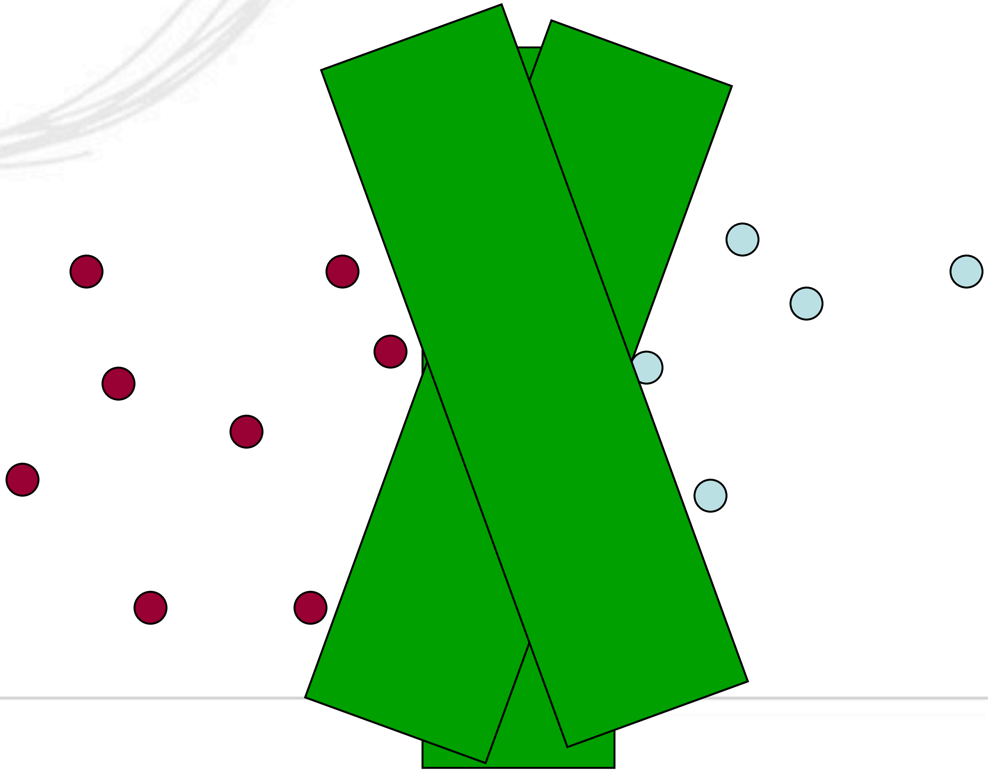
Among all the possible decisions boundaries that separate two classes, the SVM computes the one that **maximizes** the distance to the nearest data points.

Note: MLP has a different goal.

Alternative intuition

- Assume that we have "fat" separator lines. The fatter the line the less choices we have.

➢ SVM aims at finding the fattest possible separator line.



SVM – Formalization

For general n-dimensional learning problems we can describe the decision boundary as follows:

$$a_1x_1+a_2x_2+\ldots+a_nx_n+b=0$$

Or, more conveniently, in vector form as follows:

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

Thus, the vector **a** (and b) define a **hyperplane** (and its offset).

Formalization

- **w**: decision hyperplane normal vector
- The corresponding unit vector is **w**/|**w**|, where |**w**|=sqrt($\mathbf{w}^T\mathbf{w}$)
- $\mathbf{x}_i$: data point *i*
- y_i: class of data point *i* (**+1 or -1**) NB: Not 1/0
- *r*: the distance of a point from the hyperplane
- Classifier is: $f(\mathbf{x}_i) = \text{sign}(\mathbf{w}^T\mathbf{x}_i + b)$
- $\mathbf{x}'_i$: Point on the hyperplane nearest to $\mathbf{x}_i$.

Thus, **x**' is a translation of **x** by *r*:

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

Geometric Margin

- This satisfies $\mathbf{w}^T\mathbf{x}' + b = 0$, and hence we can write

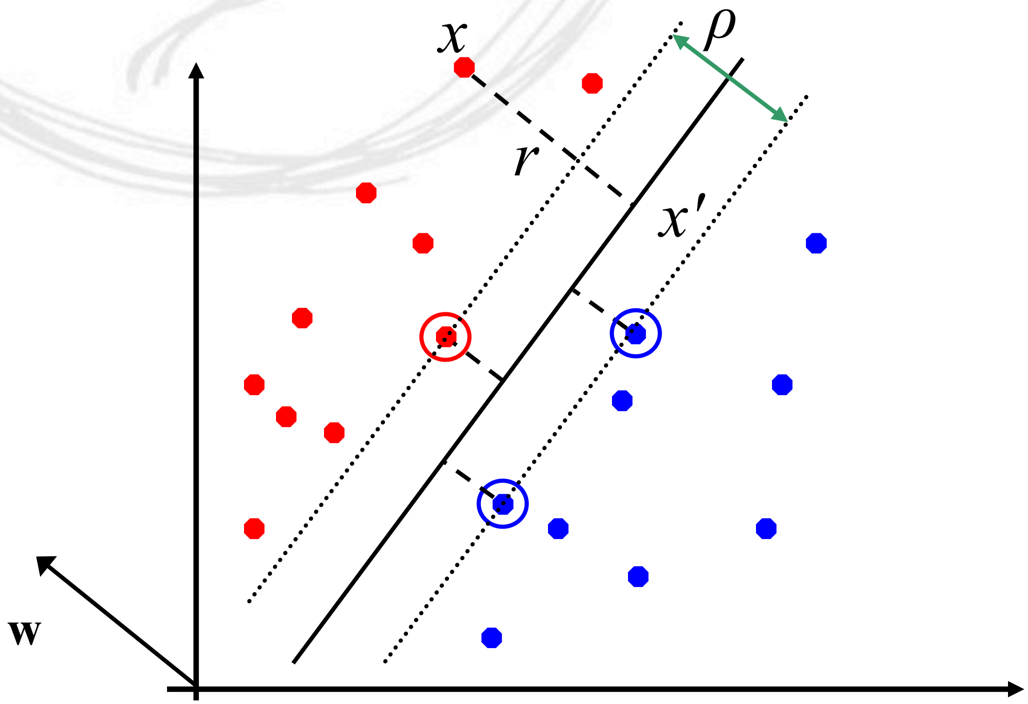
$$\mathbf{w}^T(\mathbf{x} - y r \mathbf{w}) + b = 0$$

- Solved for r gives: $r = y\dfrac{\mathbf{w}^T\mathbf{x} + b}{|\mathbf{w}|}$

- This is called the **geometric margin**.
- Note: The geometric margin is invariant to scale. We can impose any scale without affecting the geometric margin.
- It is convenient to set a scale so that the geometric margin of any data point is at least 1.

Geometric Margin

- **Examples closest to the hyperplane are *support vectors*.**
- ***Margin*** ρ of the separator is the width of separation between support vectors of classes.



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 $r\mathbf{w}/|\mathbf{w}|$.
$\mathbf{x}' = \mathbf{x} - yr\mathbf{w}/|\mathbf{w}|$.
$\mathbf{x}'$ satisfies $\mathbf{w}^T\mathbf{x}' + b = 0$.
So $\mathbf{w}^T(\mathbf{x} - yr\mathbf{w}/|\mathbf{w}|) + b = 0$
Recall that $|\mathbf{w}| = \mathrm{sqrt}(\mathbf{w}^T\mathbf{w})$.
So $\mathbf{w}^T\mathbf{x} - yr|\mathbf{w}| + b = 0$
So, solving for r gives:
$r = y(\mathbf{w}^T\mathbf{x} + b)/|\mathbf{w}|$

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 $\{(\mathbf{x_i}, y_i)\}$

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

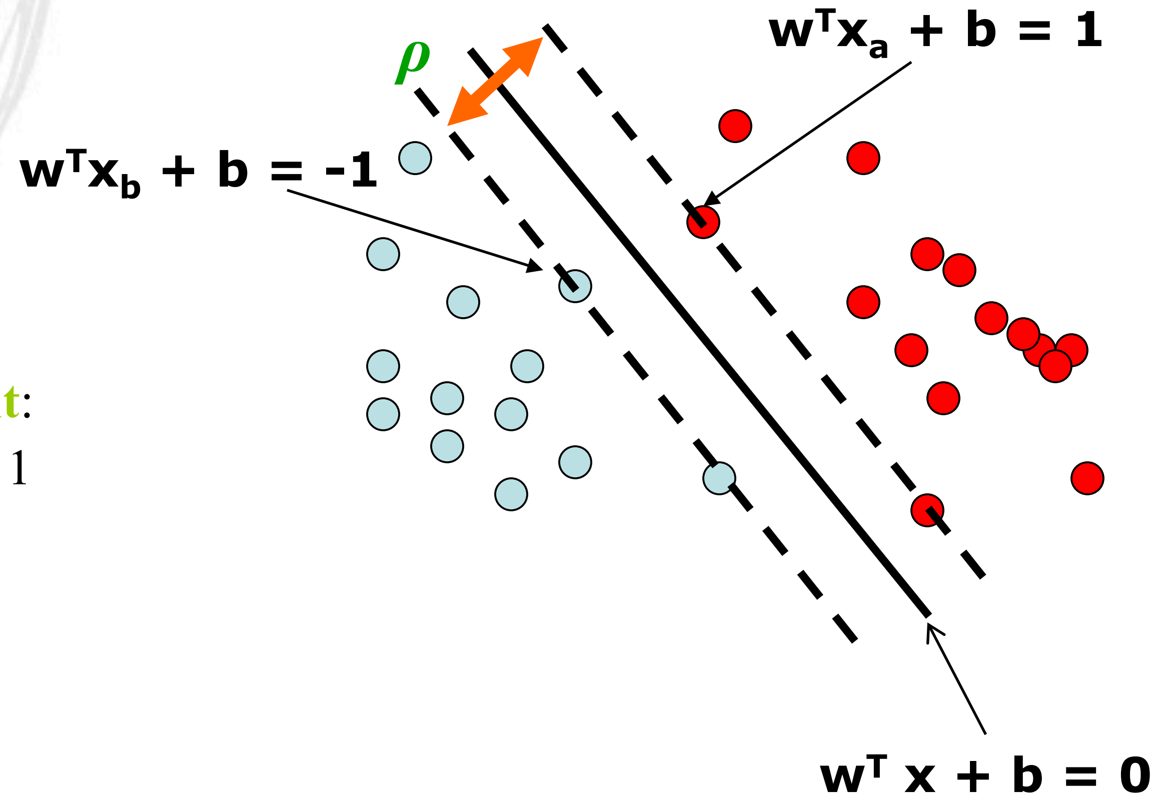
$$\mathbf{w^T x_i} + b \leq -1 \quad \text{if } y_i = -1$$

- 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: $\rho = \dfrac{2}{|\mathbf{w}|}$

Linear Support Vector Machine (SVM)



- **Hyperplane**
  $\mathbf{w}^T \mathbf{x} + \mathrm{b} = 0$
- **Extra scale constraint**:
  $\min_{\mathbf{i=1,\ldots,n}} |\mathbf{w}^T\mathbf{x}_\mathbf{i} + \mathrm{b}| = 1$

Linear SVMs Mathematically

- Then we can formulate the *quadratic optimization problem:*

Find **w** and b such that

$\rho = \frac{2}{|\mathbf{w}|}$ is maximized; and for all $\{(\mathbf{x_i}, y_i)\}$

$\mathbf{w}^T\mathbf{x_i} + b \geq 1$ if $y_i=1$; $\mathbf{w}^T\mathbf{x_i} + b \leq -1$ if $y_i = -1$

- A better formulation (min |**w**| = max 1/ |**w**|):

Find **w** and b such that

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

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

Solving the Optimization Problem

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\, \mathbf{w}^T\mathbf{w}$ 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* α_i is associated with every constraint in the primary problem.
- These methods are described in subjects on advanced math. The particulars will be omitted here. Instead we refer to **readily available software** that can solve quadratic programming problems very efficiently. See LIBSVM.

Solving the Optimization Problem

- Find the optimal w and b

$$\{\mathbf{w}^\star, b^\star\} = \min \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{Subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1,\ i = 1, 2, \cdots, n$$

- The primal problem
  - The cost function $\|\mathbf{w}\|^2/2$ is a convex function
  - The constraints are linear in $\mathbf{w}$
- Lagrangian function

$$J(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^\top \mathbf{w} - \sum_{i=1}^{n} \alpha_i \left[y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 \right]$$

- Two conditions of optimality

$$\frac{\partial J(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = \mathbf{0} \Longrightarrow \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i; \qquad \frac{\partial J(\mathbf{w}, b, \alpha)}{\partial b} = \mathbf{0} \Longrightarrow \sum_{i=1}^{n} \alpha_i y_i = 0$$

Solving the Optimization Problem

- Kuhn-Tucker conditions

$$\alpha_i \left[y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 \right] = 0, \; i = 1, 2, \cdots, n$$

  – Only the training samples satisfy $y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 = 0$ can have nonzero α, and they are called "support vectors"

- The dual problem (find the optimal α, a QP problem)

$$\{\alpha^\star\} = \max \left[\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \right]$$

$$\text{Subject to} \;\; 1) \; \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$2) \; \alpha_i \geq 0, \; i = 1, 2, \cdots, n$$

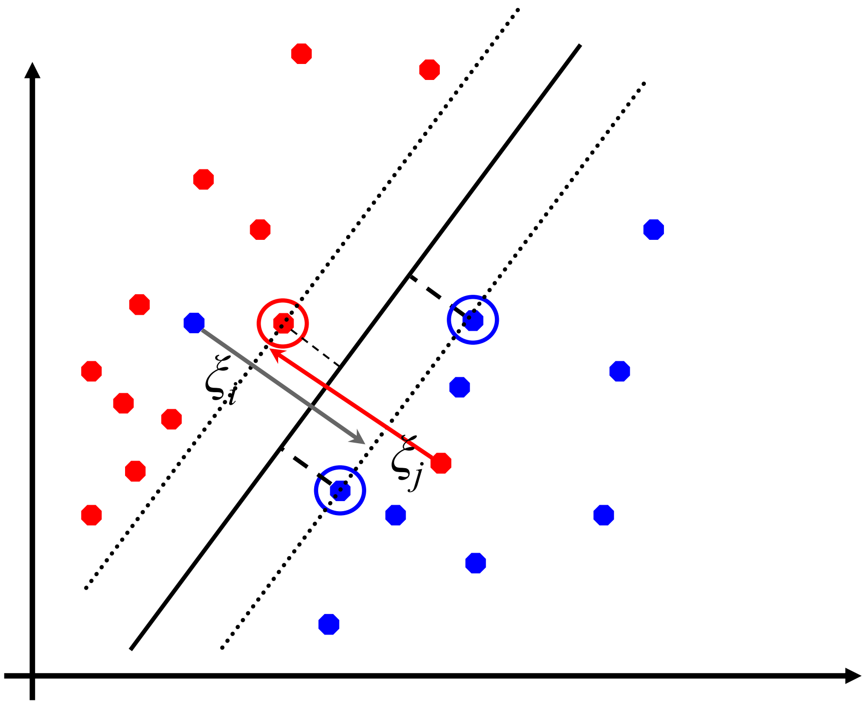
$$\mathbf{w}^\star = \sum_{i=1}^{n} \alpha_i^\star y_i \mathbf{x}_i \qquad b^\star = 1 - \mathbf{w}^\star \mathbf{x}^s$$

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

A support vector

Soft Margin Classification

- If the training data is **not linearly separable**, *slack variables* ξ_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 Mathematically

- The **old formulation**:

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

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

> Find **w** and b such that
> $\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
> $y_i\,(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

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

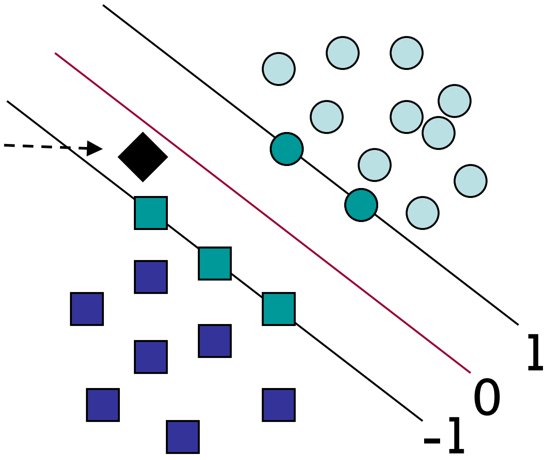
Classification with SVMs

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

Score > t: yes

Score < $-t$: no

else: don't know

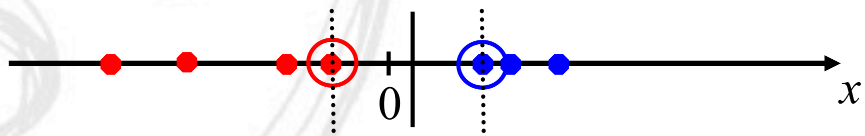


Linear SVMs: Summary

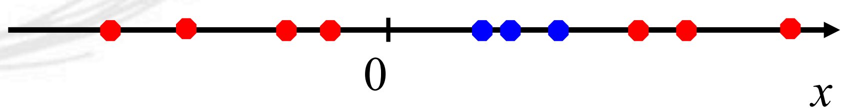
- The classifier is a *separating* ***hyperplane***.
- The most "important" training points are the **support vectors**; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x_i}$ are support vectors.

Non-linear SVMs

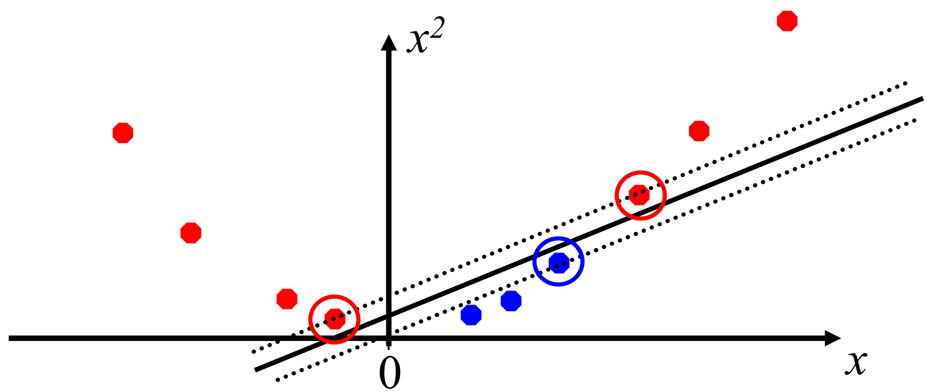
- Datasets that are linearly separable (with some noise) work out great:

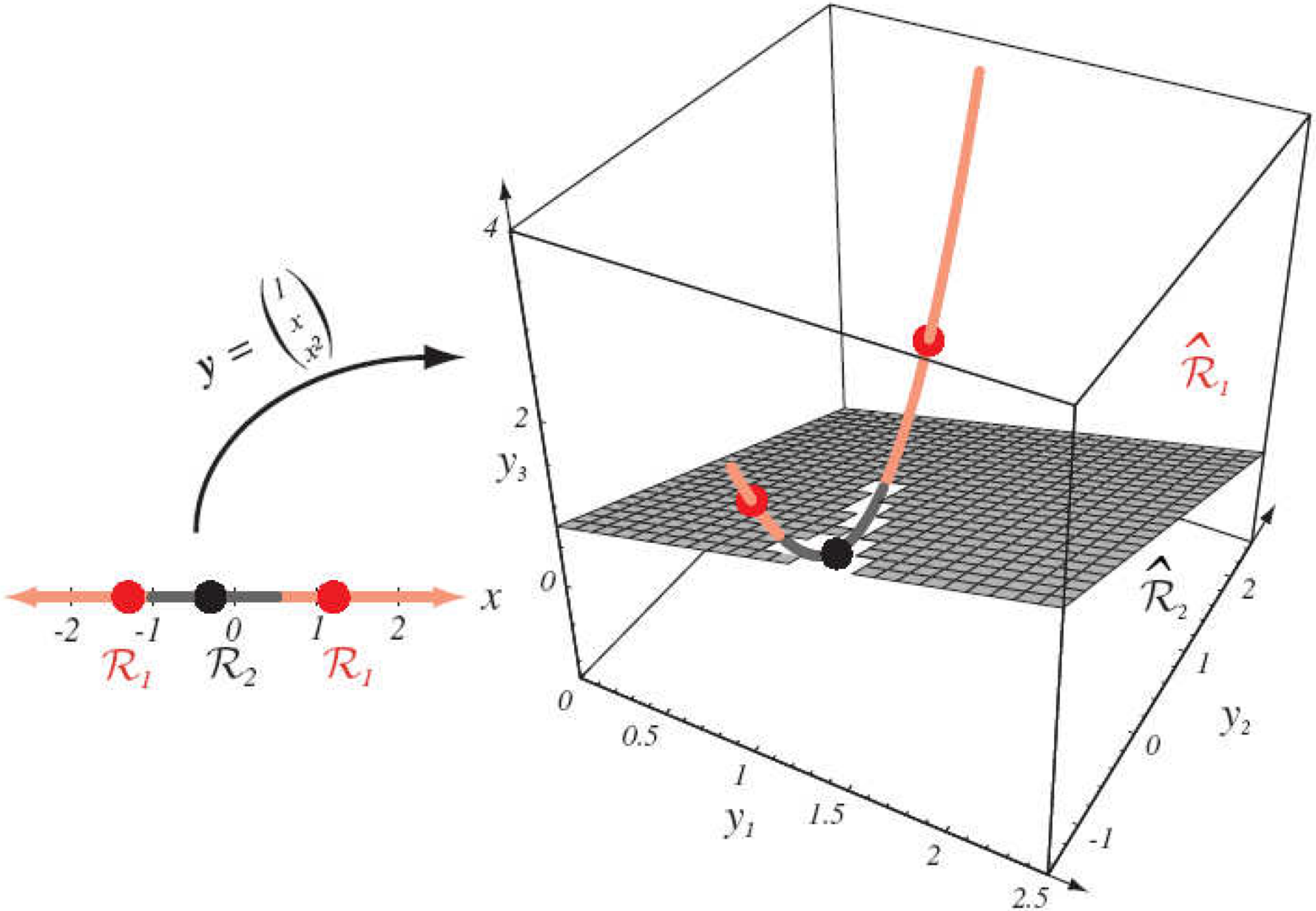


- But what are we going to do if the dataset is just too hard?

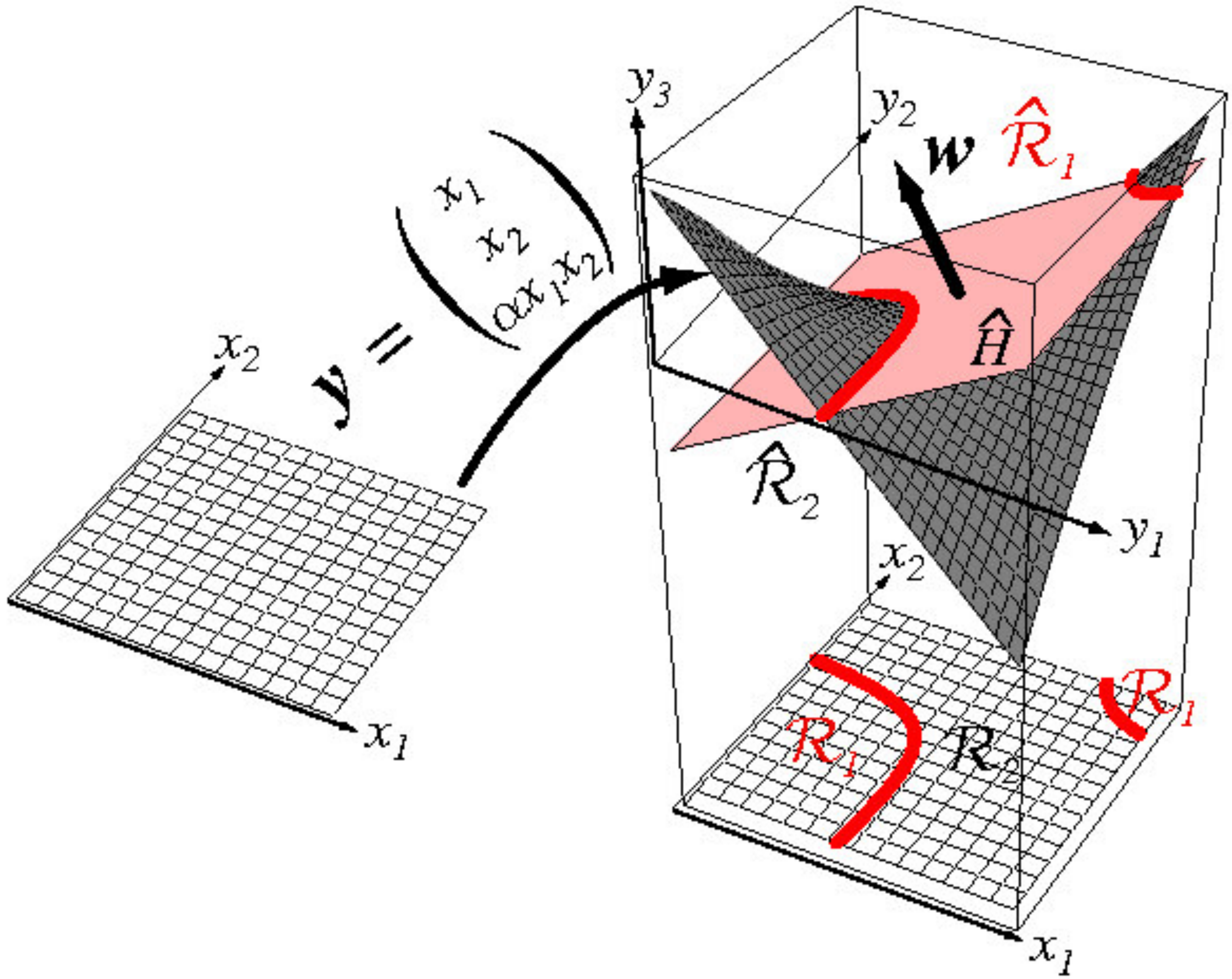


- How about … **mapping data to a higher-dimensional space**:



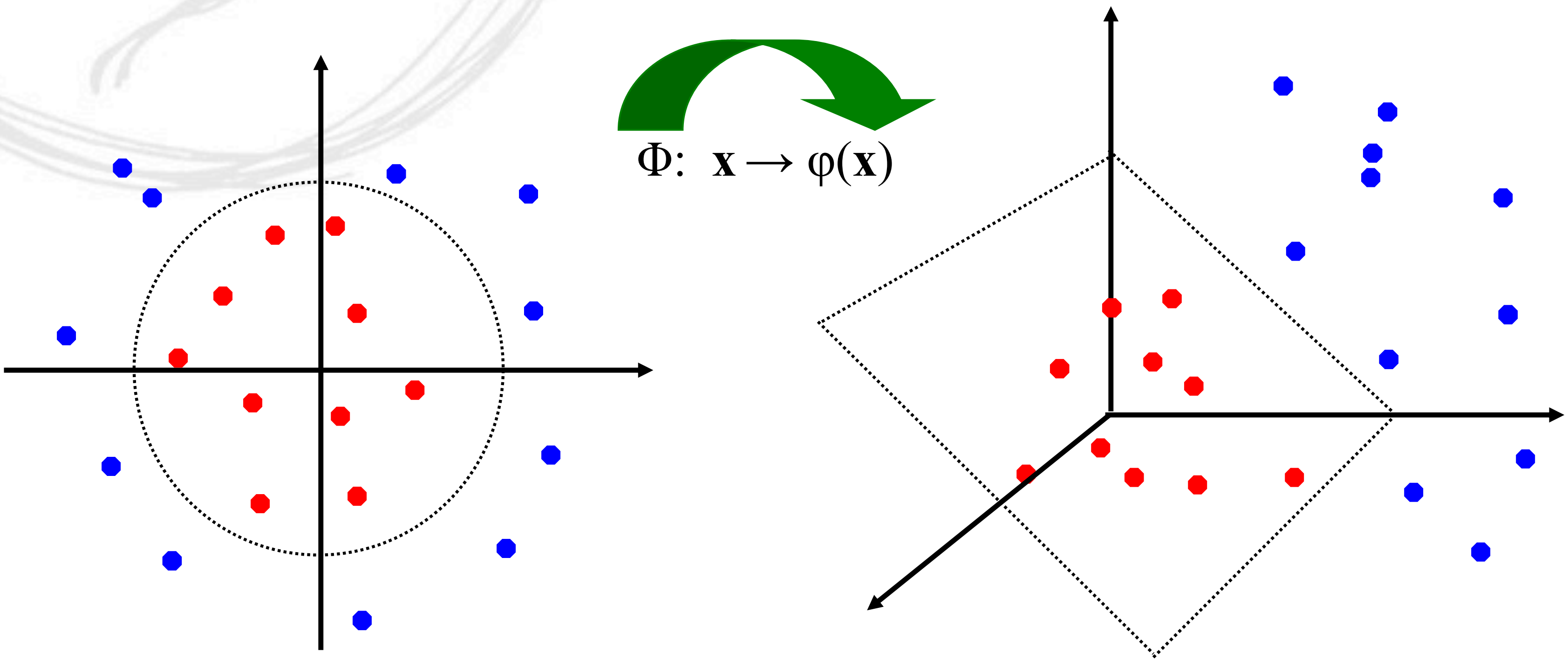


$y = \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix}$
x
-2
-1
0
1
2
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{R}_1$
4
2
y_3
0
$\hat{\mathcal{R}}_1$
$\hat{\mathcal{R}}_2$
0
0.5
1
y_1
1.5
2
2.5
2
1
y_2
0
-1


$\mathbf{y} = \begin{pmatrix} x_1 \\ x_2 \\ \alpha x_1 x_2 \end{pmatrix}$
x_1
x_2
y_1
y_2
y_3
$\boldsymbol{w}$
$\hat{\mathcal{R}}_1$
$\hat{\mathcal{R}}_2$
$\hat{H}$
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{R}_1$
x_1
x_2

Non-linear SVMs: Kernel trick

- 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 $K(x_i,x_j)=x_i^Tx_j$**
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner 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.

The "Kernel Trick"

•**Example**:

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

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

$K(\mathbf{x_i},\mathbf{x_j})=(1 + \mathbf{x_i}^T\mathbf{x_j})^2 = 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$

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

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

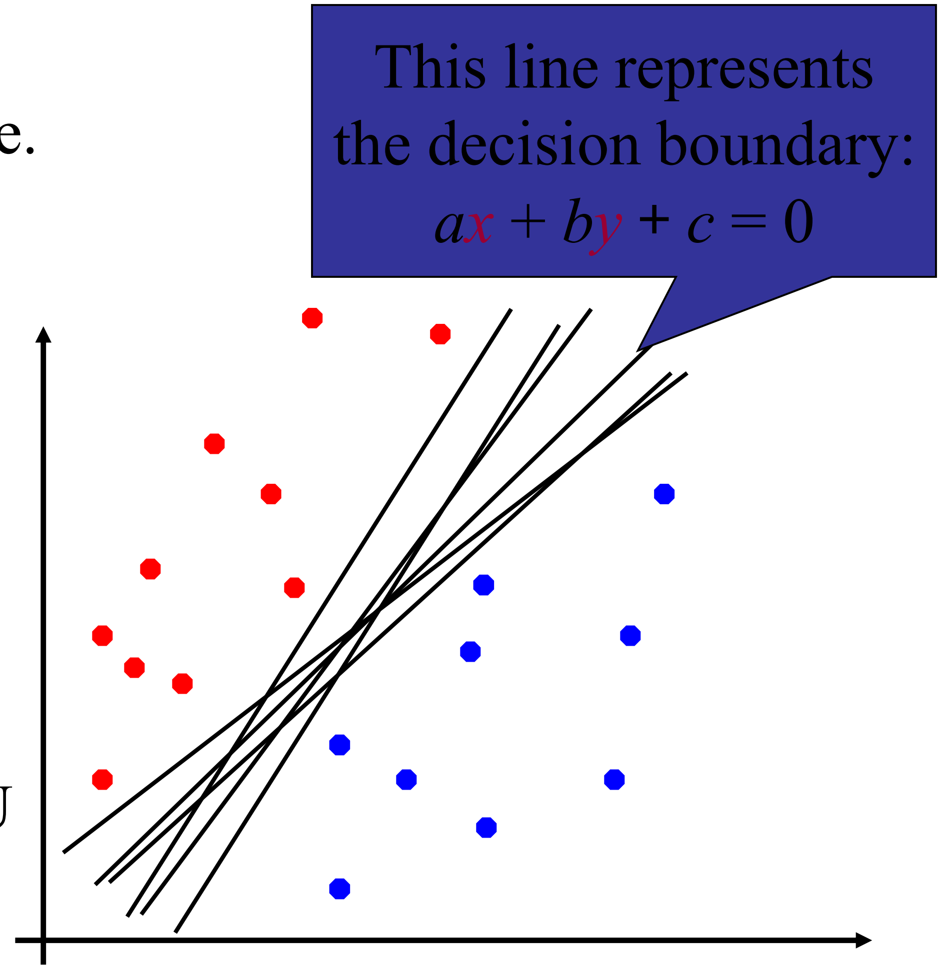
Kernels

- Why use kernels?
  - Make non-separable problem separable (in another space).
  - Map data into better representational space
- Common kernels
  - Linear kernel **$K(x,z) = x^T z$**
  - Polynomial kernel **$K(x,z) = (1+x^T z)^d$**
    - Gives feature conjunctions
  - Radial basis function (infinite dimensional space)

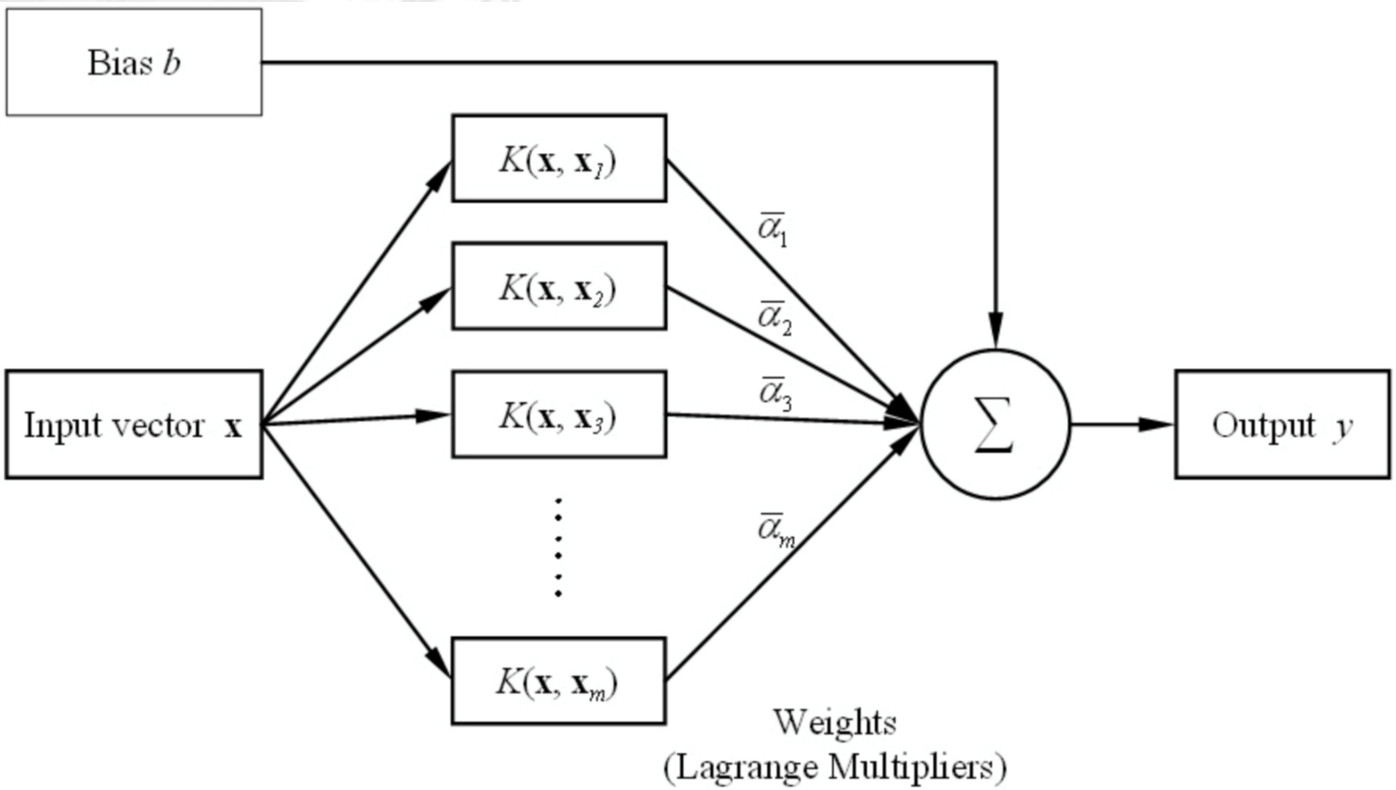
$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}$$

Summary

- 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
- Popularly applied on single CPU systems.



Summary



Hidden Nodes
(Support Vectors)

$$y = f(\mathbf{x}) = \sum_{k=1}^{m} \overline{\alpha}_k \cdot K(\mathbf{x}, \mathbf{x}_k) + b$$