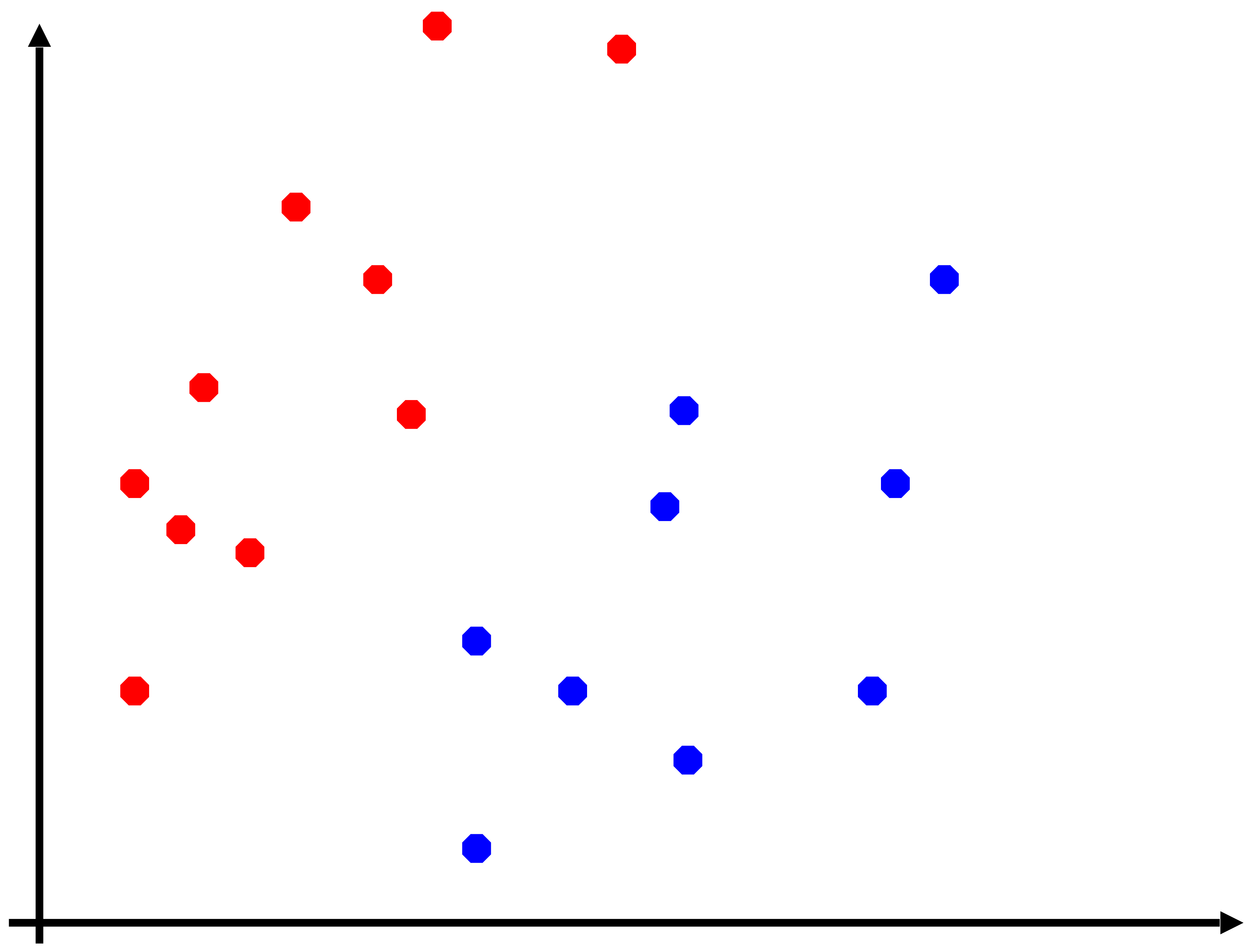
SVM

Machine Learning and Deep Learning
Lesson #6

AImage Lab
University of Modena and Reggio Emilia

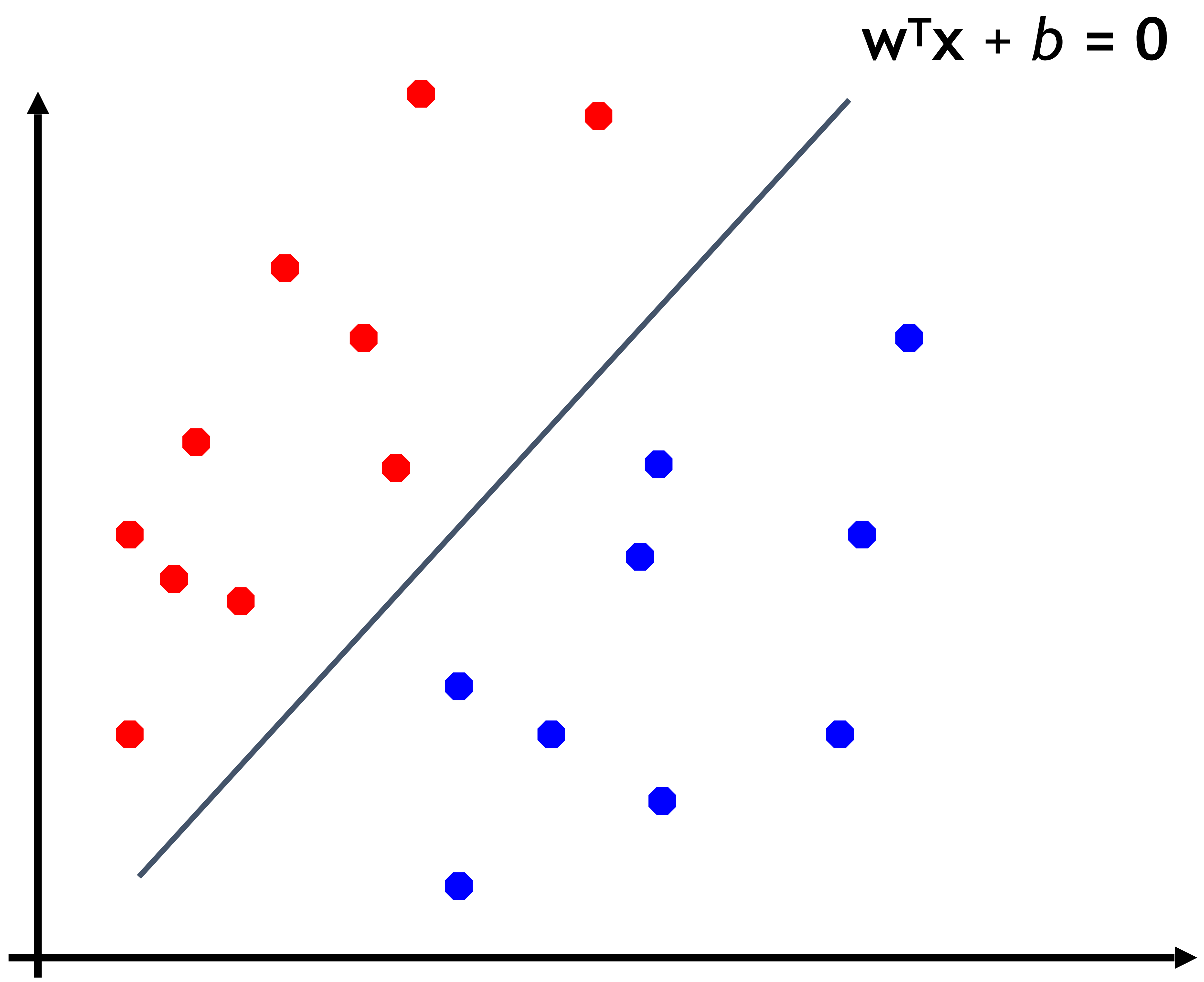
Linear classification Revisited: Linear Separators

- Binary classification can be viewed as the task of separating classes in feature space:



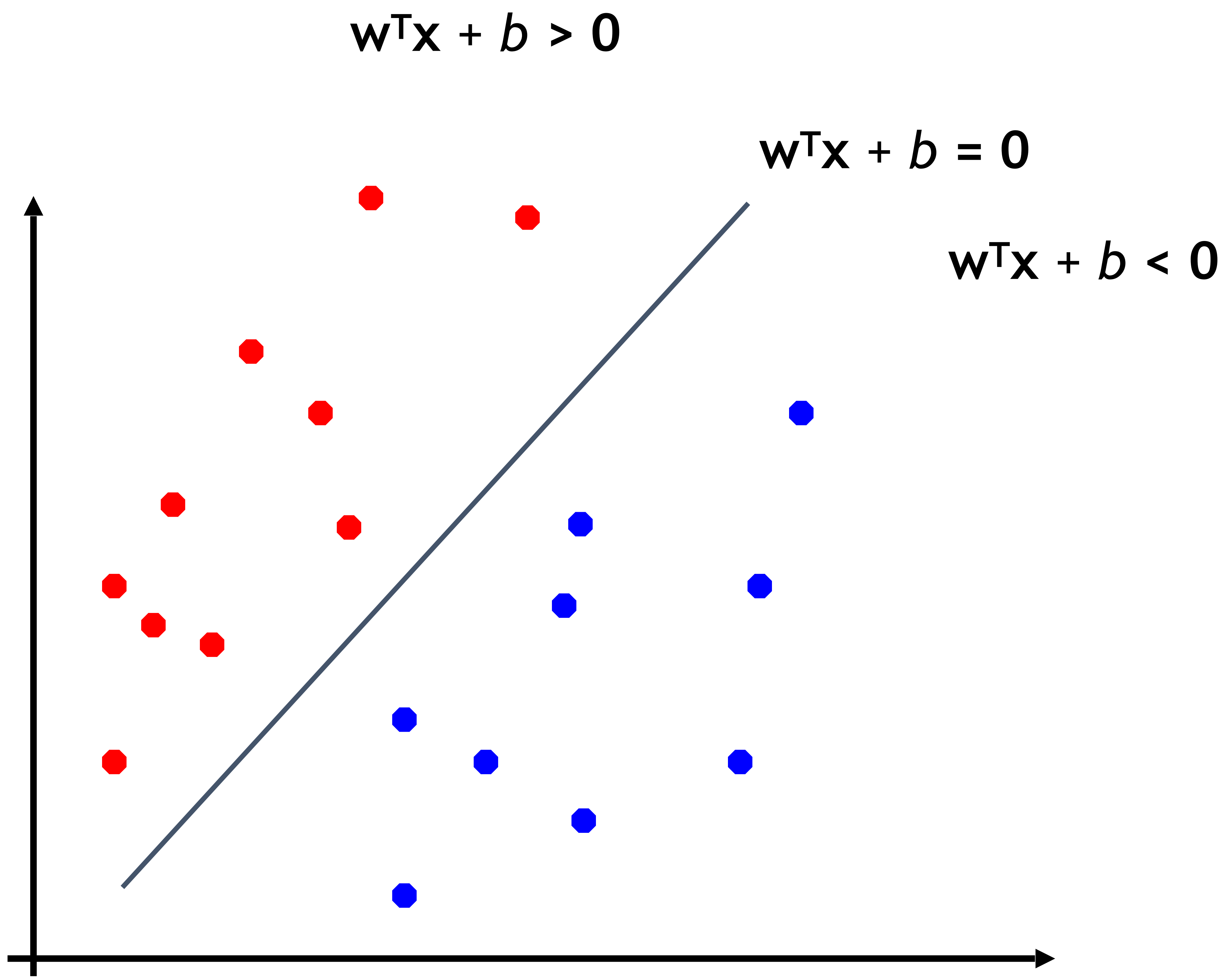
Linear classification Revisited: Linear Separators

- Binary classification can be viewed as the task of separating classes in feature space:



Linear classification Revisited: Linear Separators

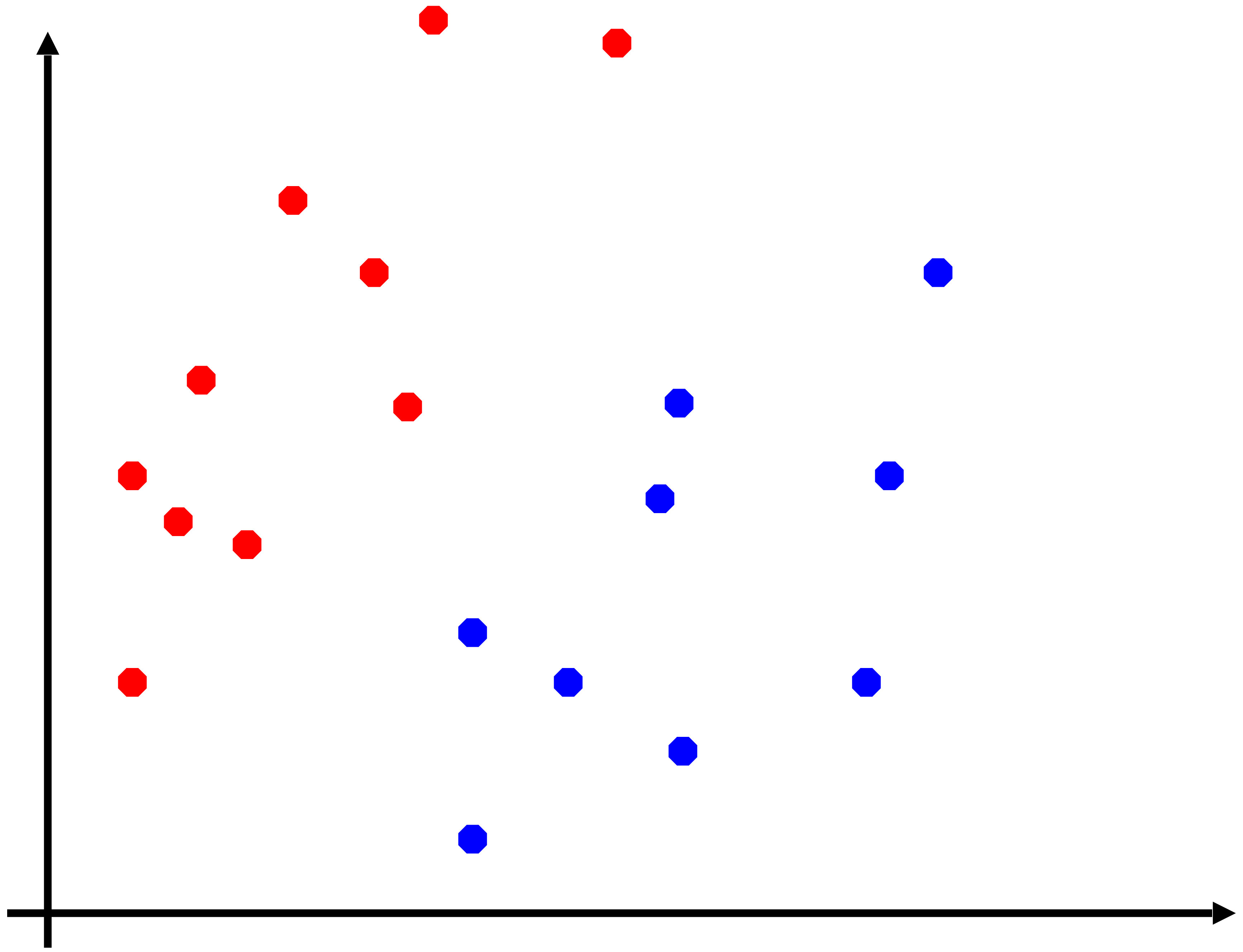
- Binary classification can be viewed as the task of separating classes in feature space:



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

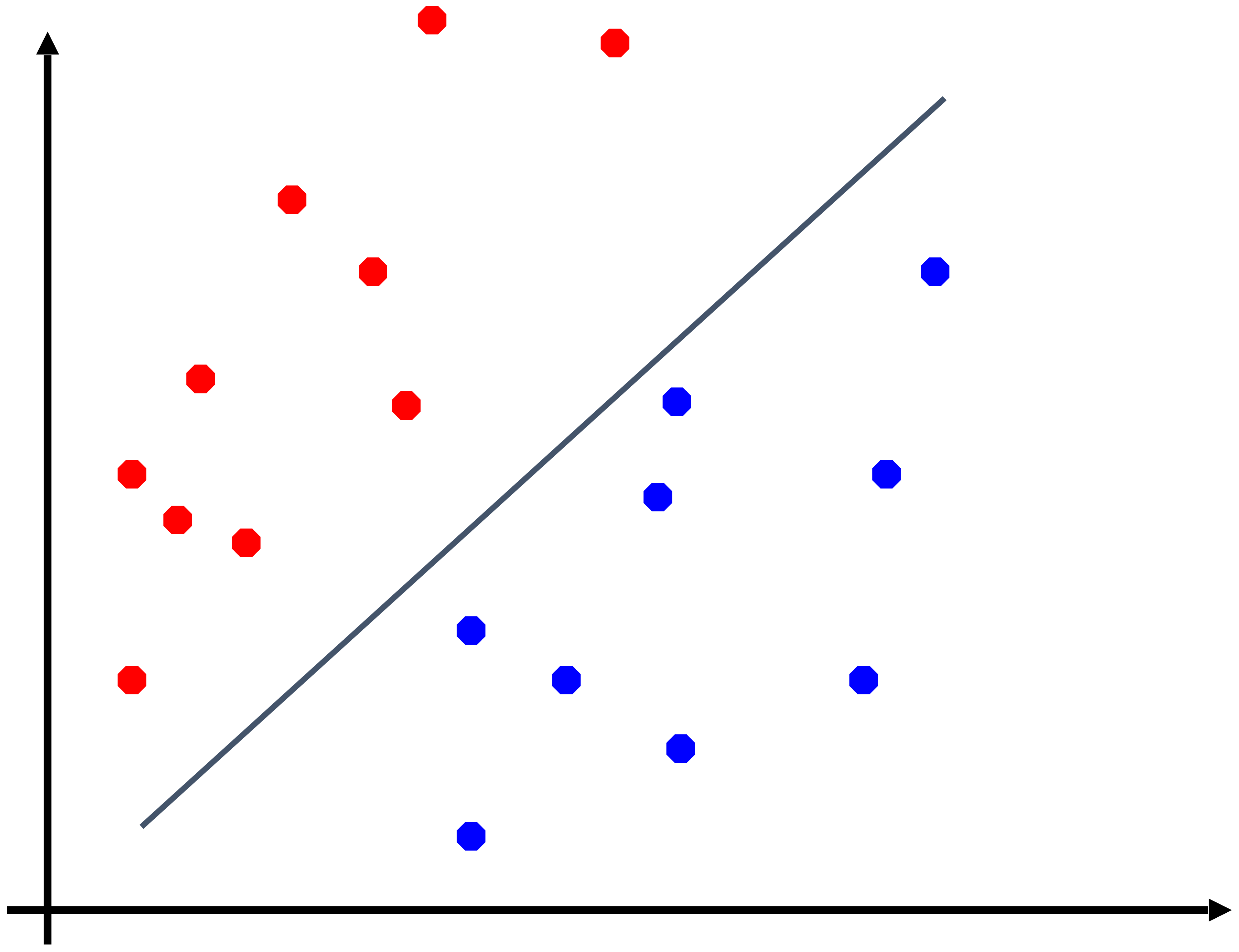
Linear Separators

- Which of the linear separators is optimal?



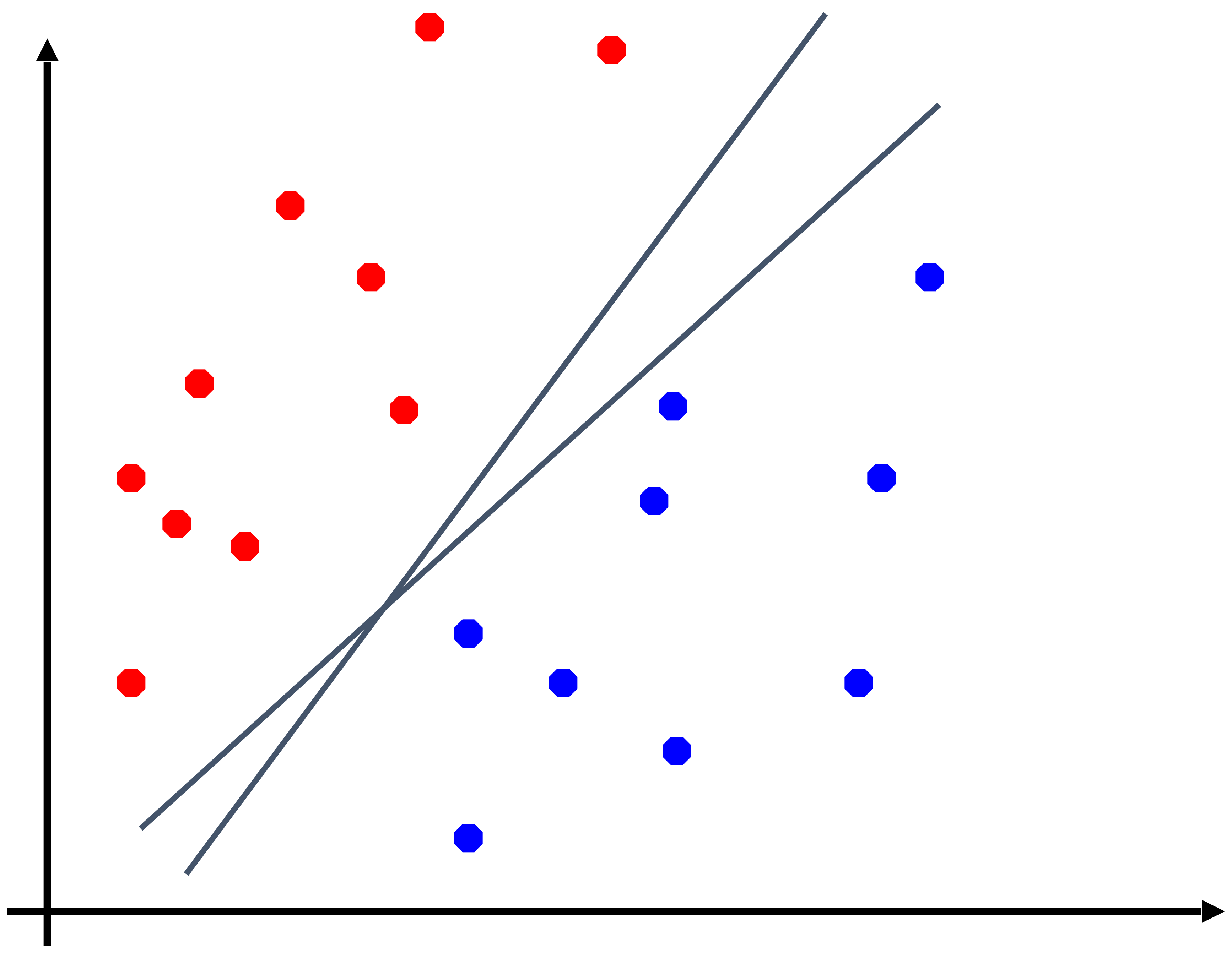
Linear Separators

- Which of the linear separators is optimal?



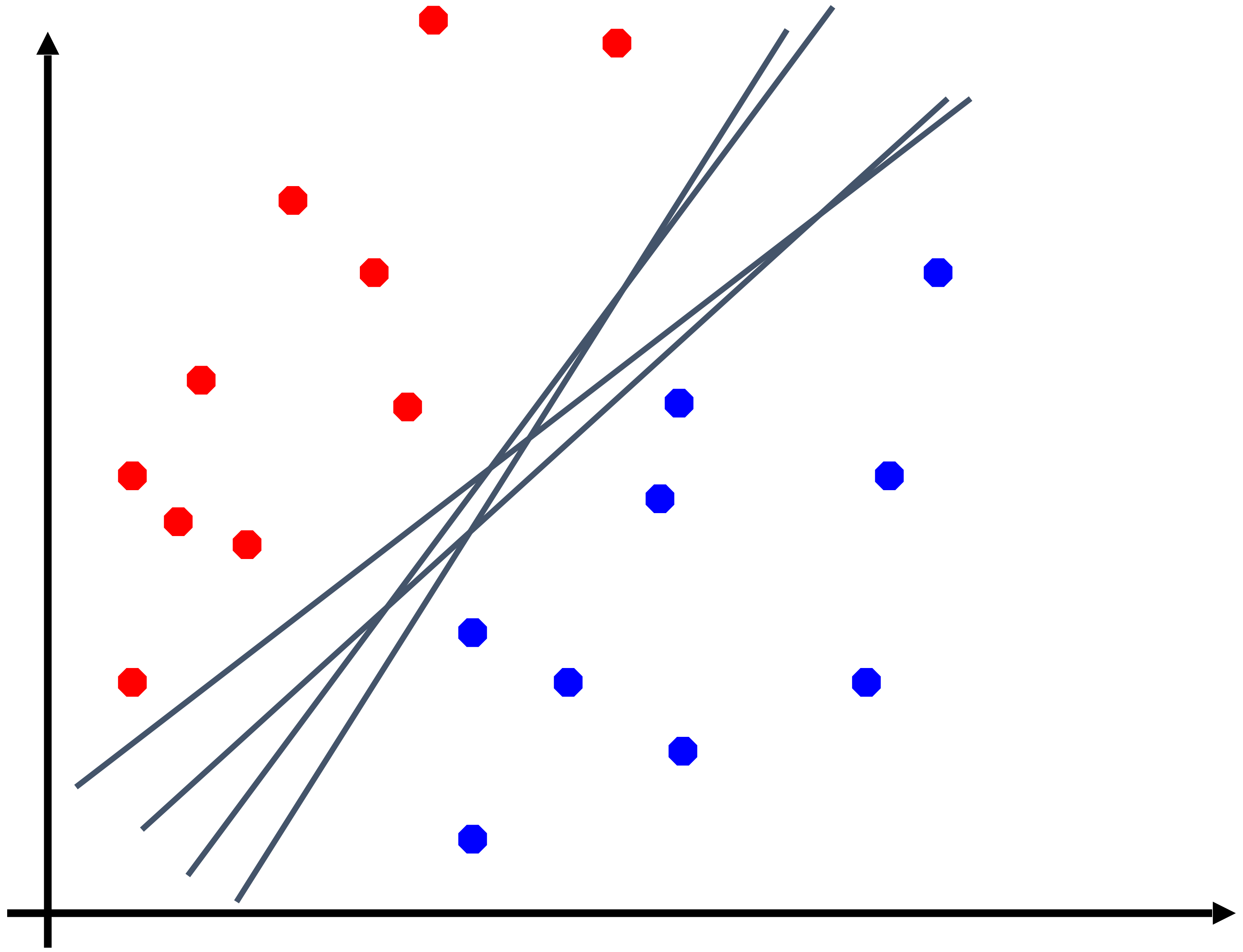
Linear Separators

- Which of the linear separators is optimal?



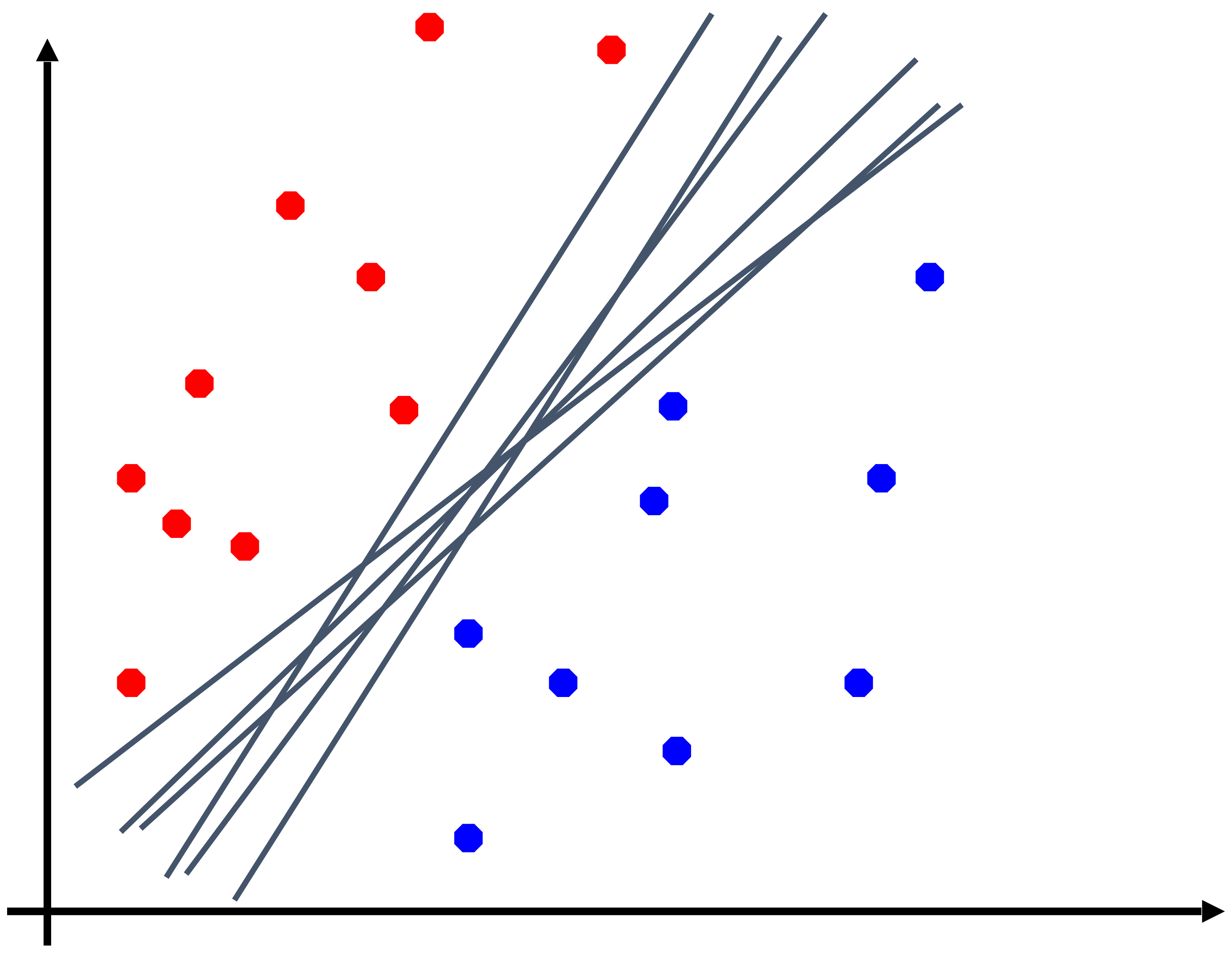
Linear Separators

- Which of the linear separators is optimal?



Linear Separators

- Which of the linear separators is optimal?



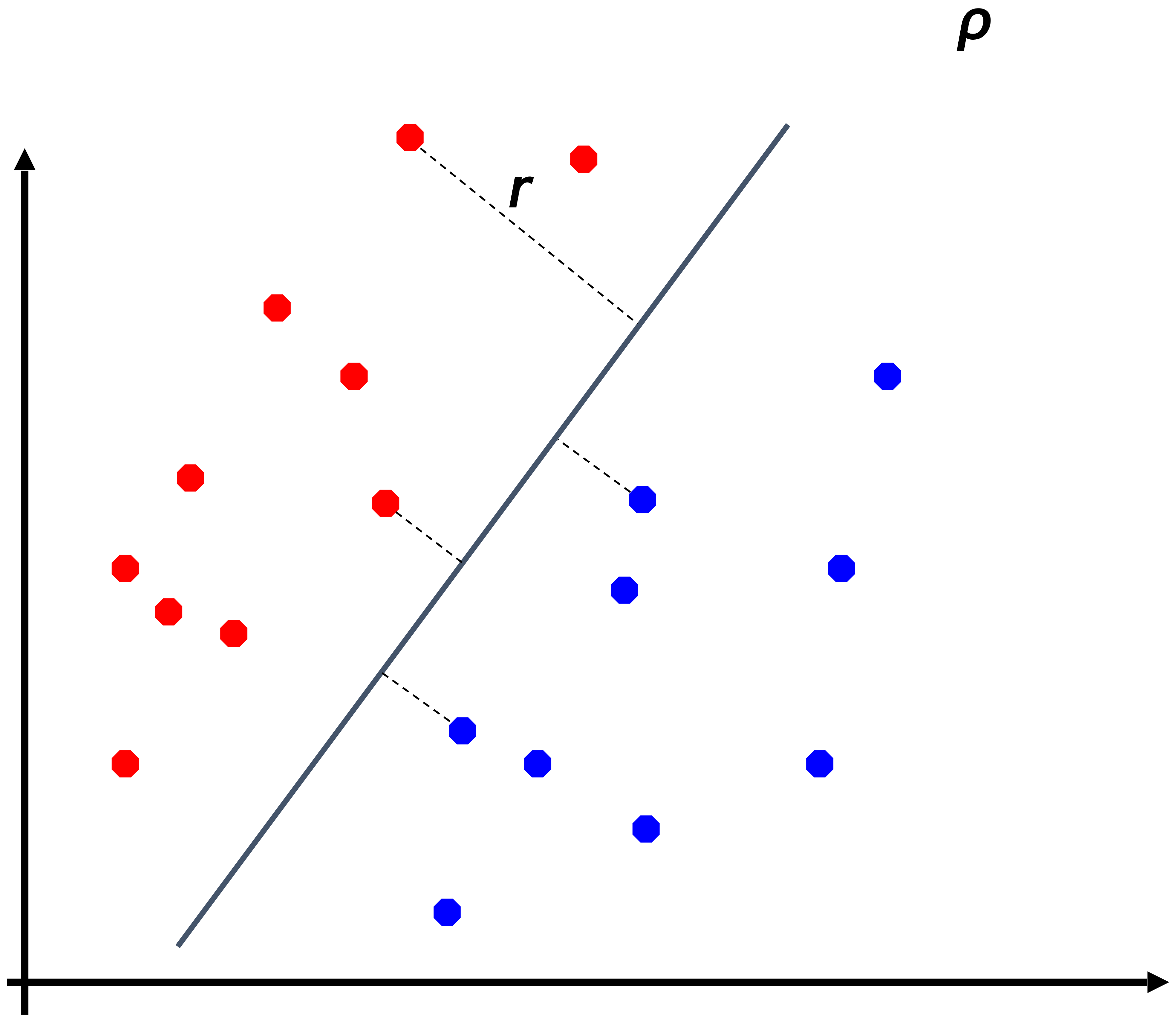
Classification Margin

- Distance from example x_i to the separator is

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

- Examples closest to the hyperplane are *support vectors*.

- *Margin ρ* of the separator is the distance between support vectors.



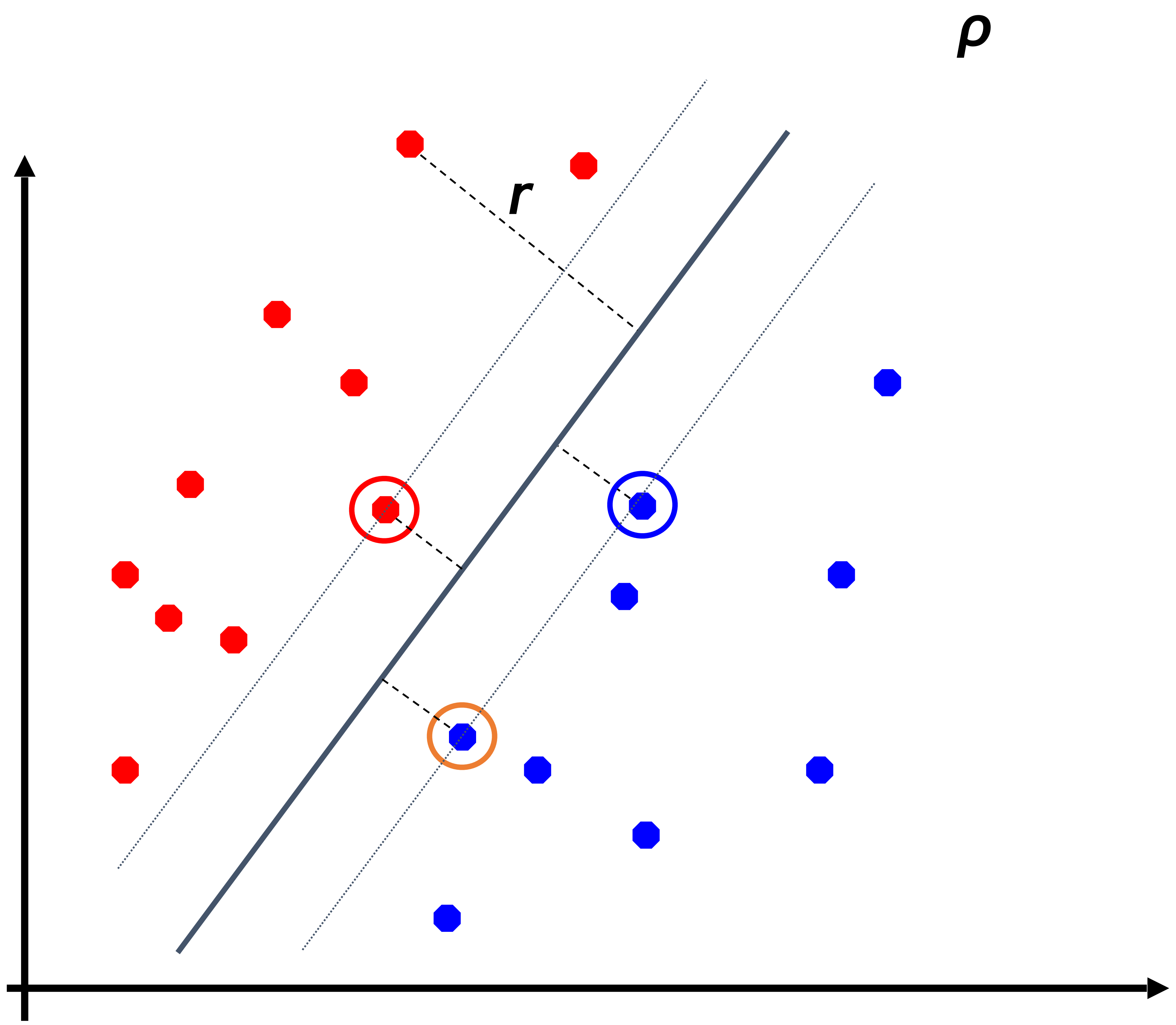
Classification Margin

- Distance from example x_i to the separator is

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

- Examples closest to the hyperplane are *support vectors*.

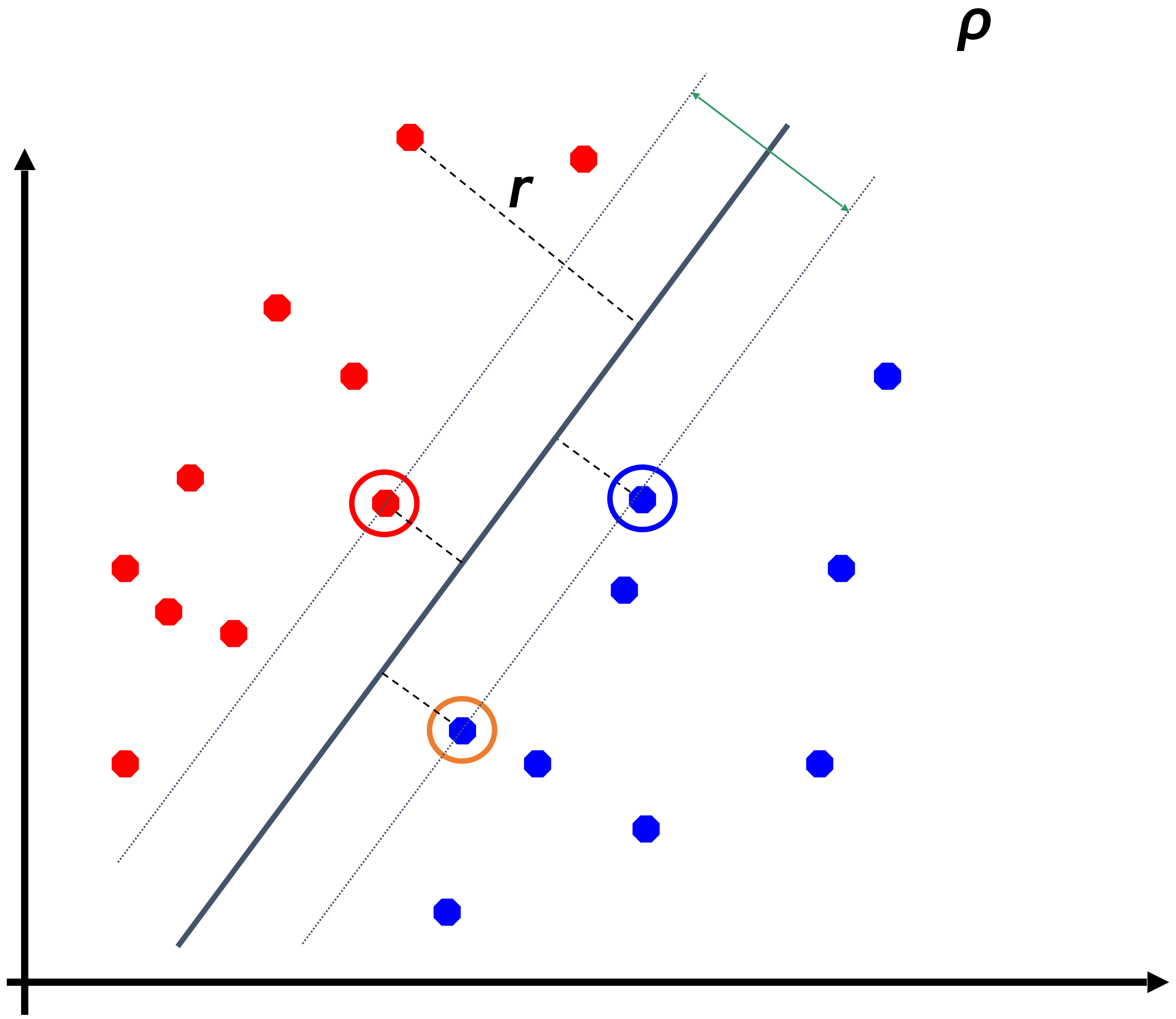
- *Margin ρ* of the separator is the distance between support vectors.



Classification Margin

- Distance from example x_i to the separator is

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

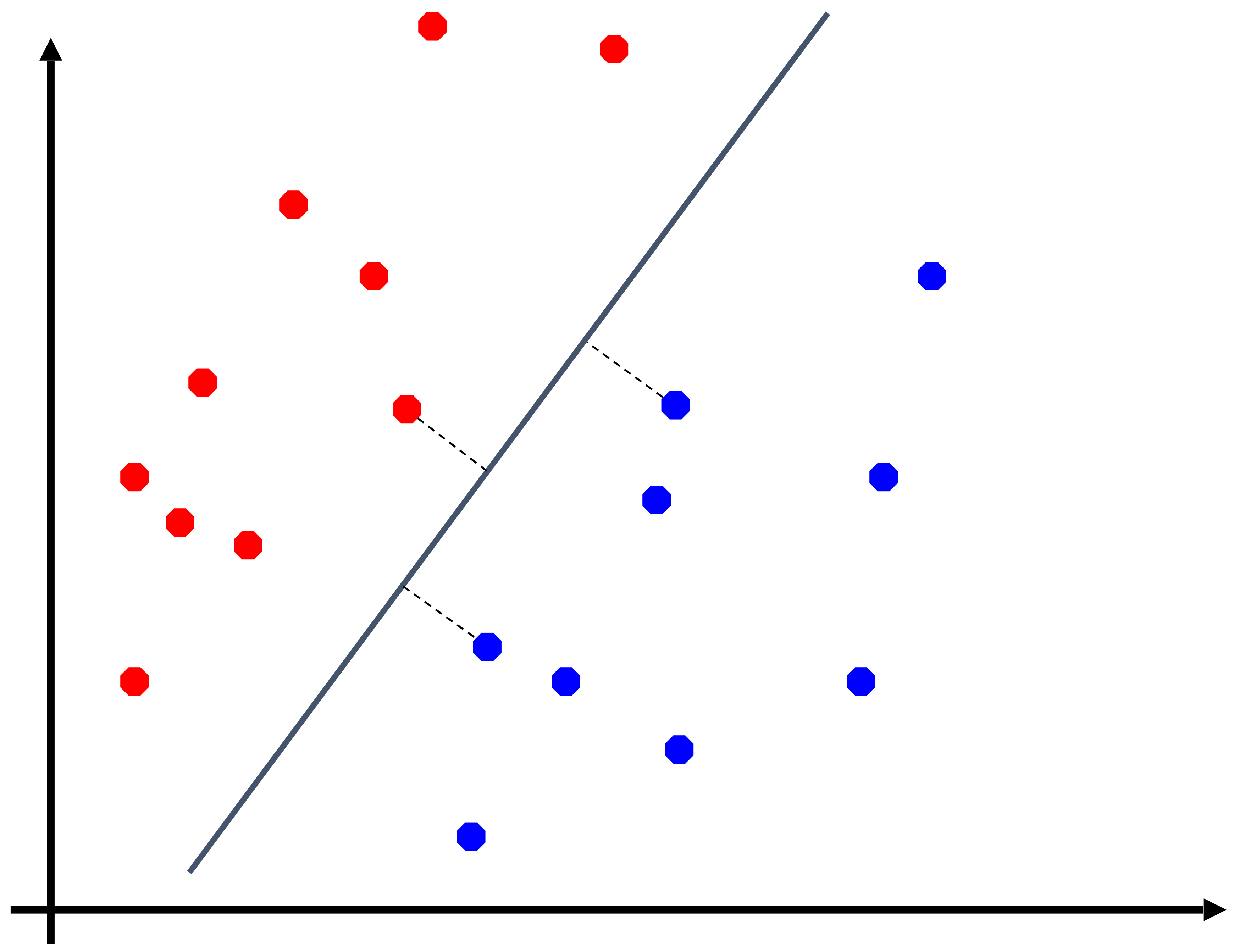


- Examples closest to the hyperplane are *support vectors*.

- *Margin ρ* of the separator is the distance between support vectors.

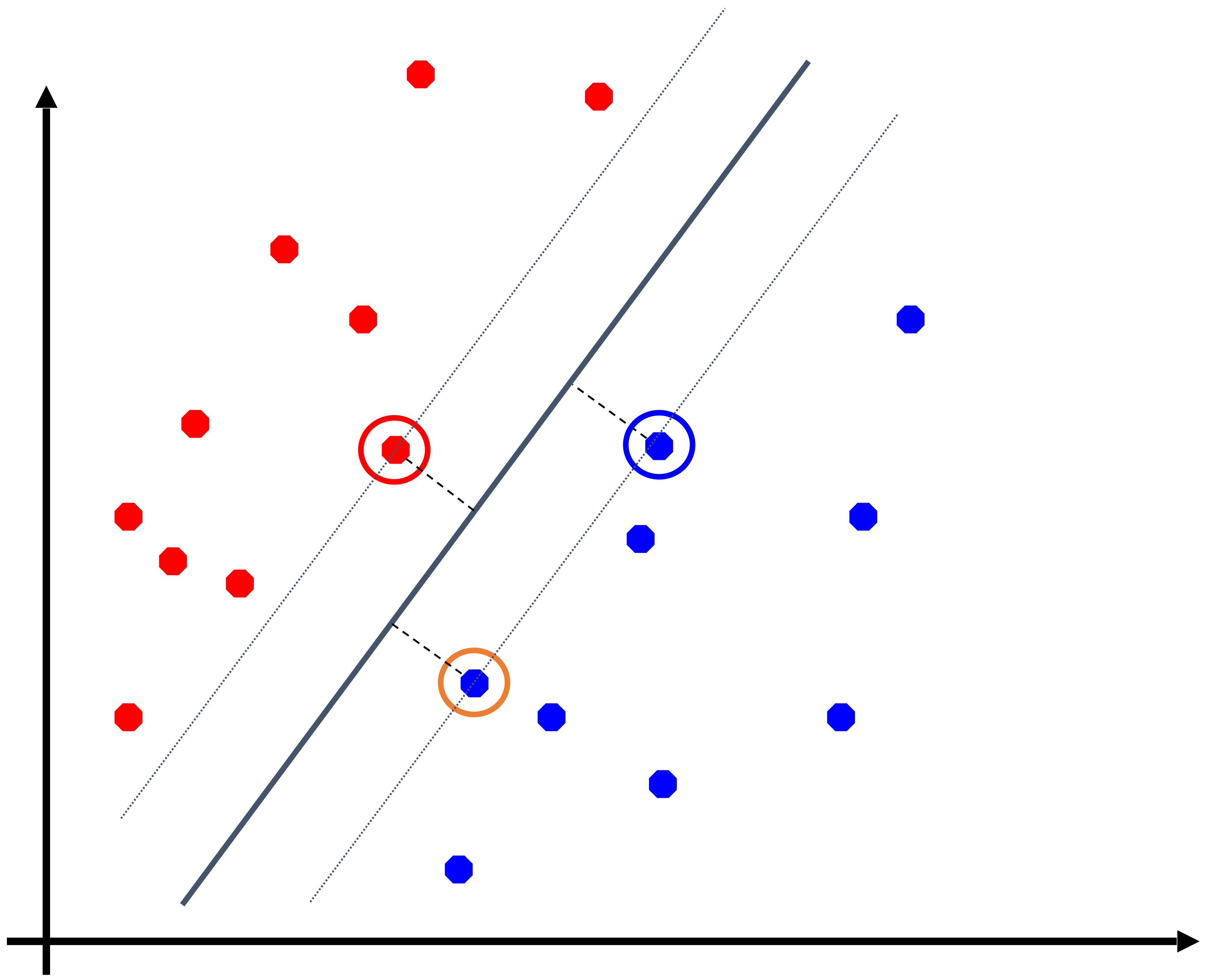
Maximum Margin Classification

- Maximizing the margin is good according to intuition and PAC theory.
- Implies that only support vectors matter; other training examples are ignorable.



Maximum Margin Classification

- Maximizing the margin is good according to intuition and PAC theory.
- Implies that only support vectors matter; other training examples are ignorable.



Linear SVM Mathematically

- Let training set $\{(x_i, y_i)\}_{i=1..n}$, $x_i \in R^d$, $y_i \in \{-1, 1\}$ be separated by a hyperplane with margin ρ. Then for each training example (x_i, y_i):

$$\begin{aligned} \mathbf{w}^T\mathbf{x}_i + b &\leq -\rho/2 \quad \text{if } y_i = -1 \\ \mathbf{w}^T\mathbf{x}_i + b &\geq \rho/2 \quad \text{if } y_i = 1 \end{aligned} \quad \Leftrightarrow \quad \boldsymbol{y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq \rho/2}$$

- For every support vector x_s the above inequality is an equality. After rescaling w and b by $\rho/2$ in the equality, we obtain that distance between each x_s and the hyperplane is

$$r = \frac{\mathrm{y}_s(\mathbf{w}^T\mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

- Then the margin can be expressed through (rescaled) w and b as:

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

Linear SVMs Mathematically (cont.)

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

Find w and b such that

$$\rho = \frac{2}{\|\mathbf{w}\|} \quad \text{is maximized}$$

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

Which can be reformulated as:

Find w and b such that

$\Phi(\mathbf{w}) = ||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$ is minimized

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

Solving the Optimization Problem
Solution 1: QP

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

Find $a_1 \ldots a_n$ such that
$Q(\alpha) = \Sigma a_i - \frac{1}{2}\Sigma\Sigma a_i a_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma a_i y_i = 0$
(2) $a_i \geq 0$ for all a_i

- Need to optimize a *quadratic* function subject to *linear* constraints.
- Quadratic optimization problems are a well-known class of mathematical programming problems for which several (non-trivial) algorithms exist.
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* a_i is associated with every inequality constraint in the primal (original) problem:

The Optimization Problem Solution

- Given a solution $a_1 \dots a_n$ to the dual problem, solution to the primal is:

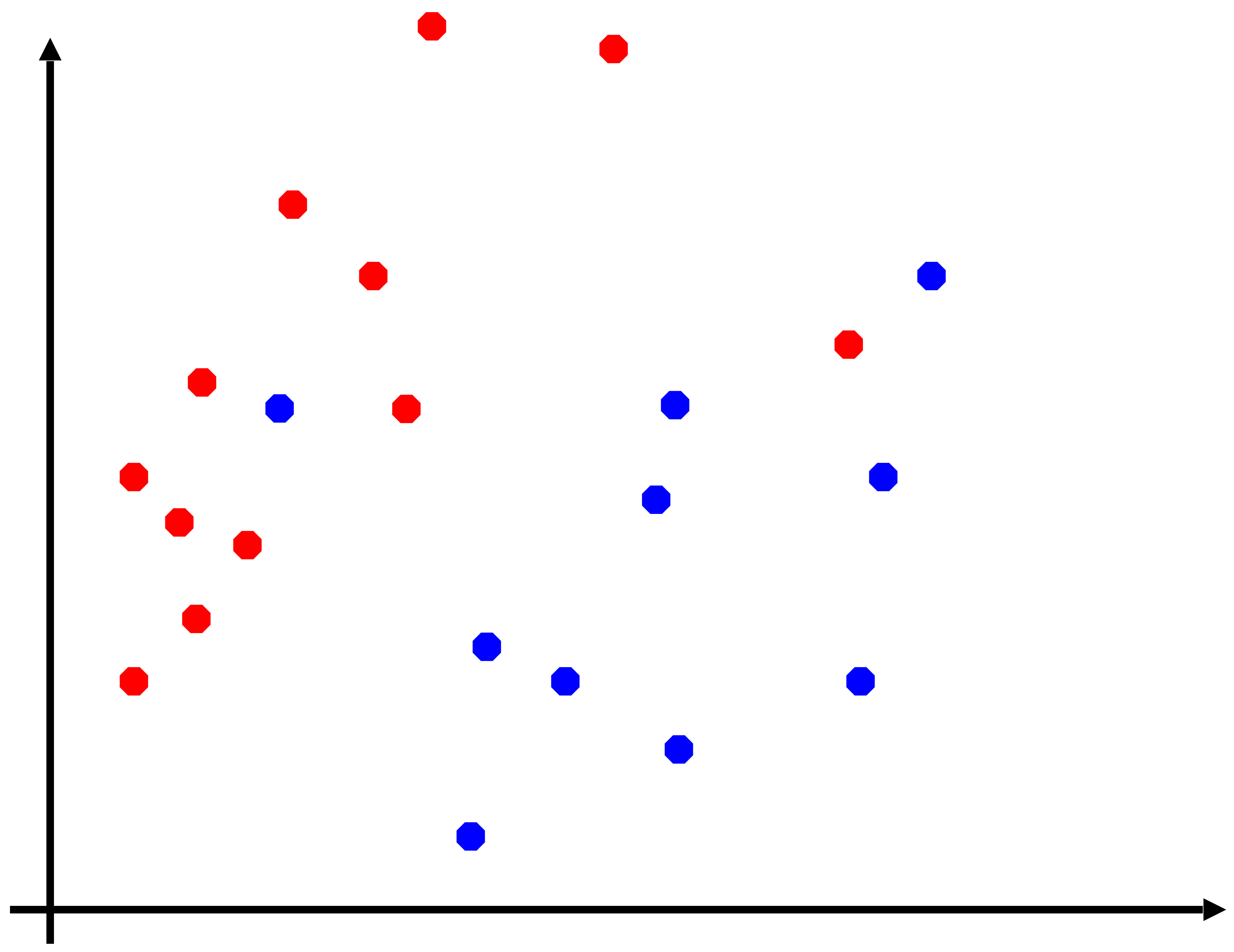
$$\boxed{\mathbf{w} = \Sigma a_i y_i \mathbf{x}_i \qquad b = y_k - \Sigma a_i y_i \mathbf{x}_i{}^T \mathbf{x}_k \quad \text{for any } a_k > 0}$$

- Each non-zero a_i indicates that corresponding x_i is a support vector.
- Then the classifying function is (note that we don't need w explicitly):

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

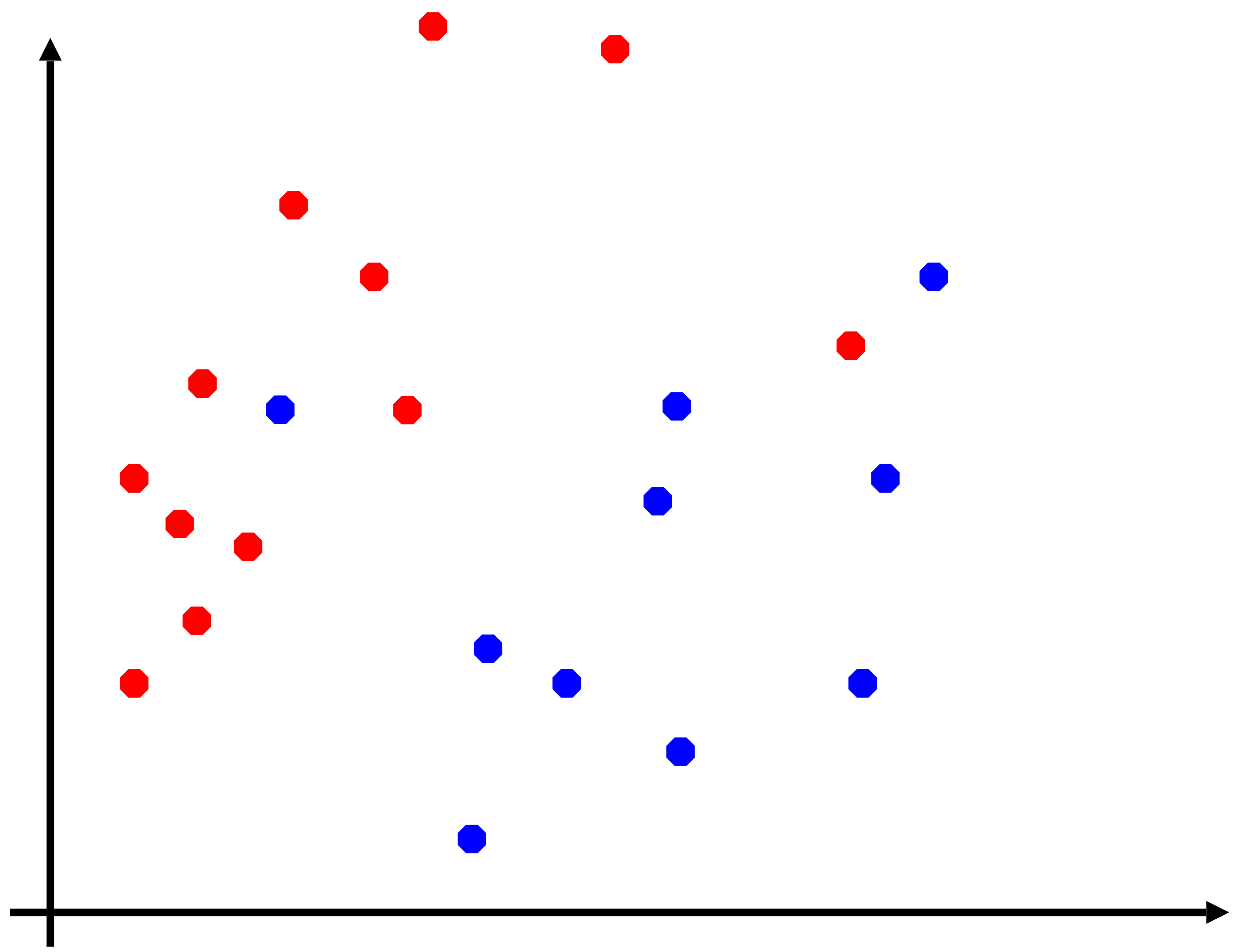
- Notice that it relies on an *inner product* between the test point x and the support vectors x_i – we will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $x_i^T x_j$ between all training points.

Soft Margin Classification



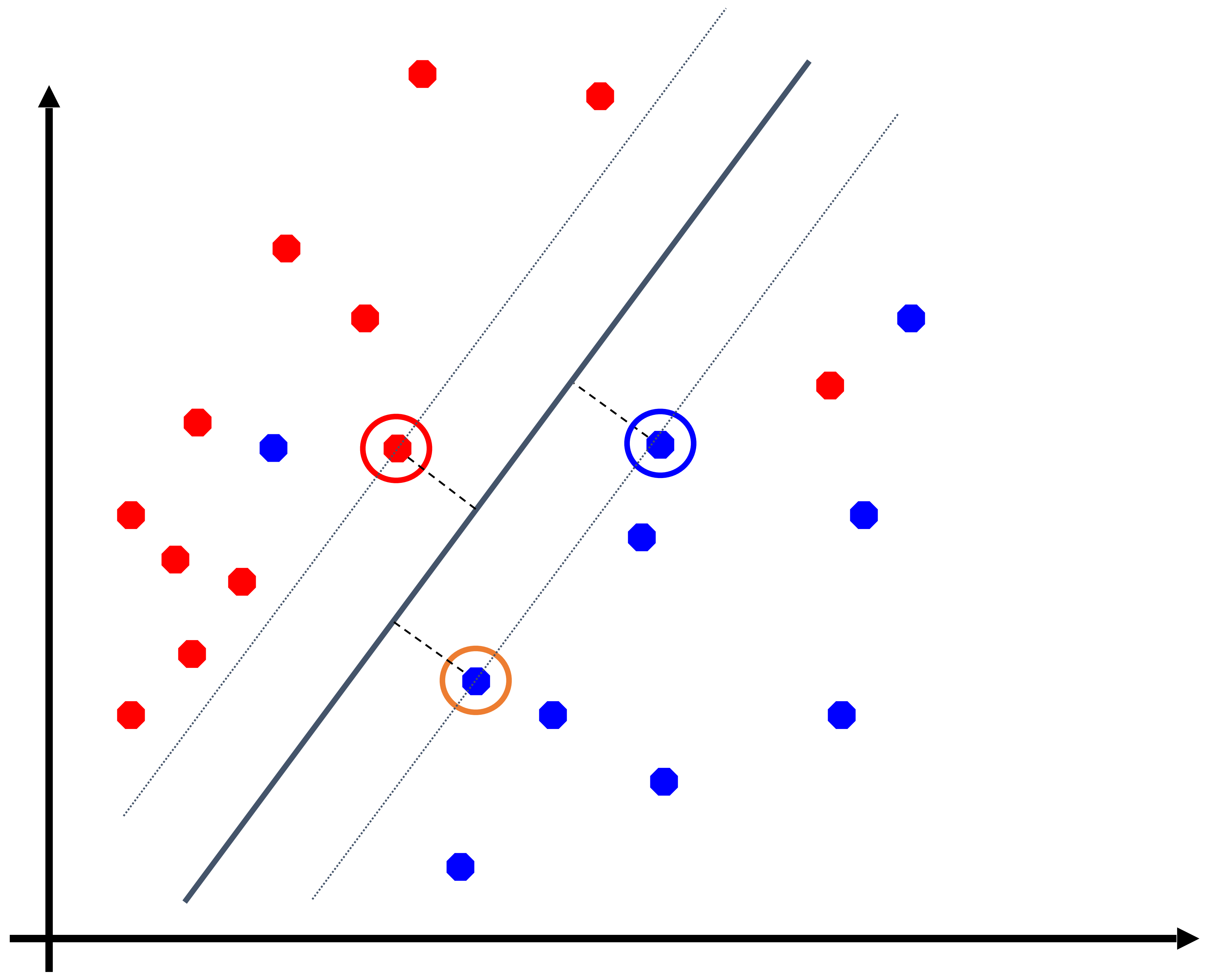
Soft Margin Classification

- **What if the training set is not linearly separable?**
- *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



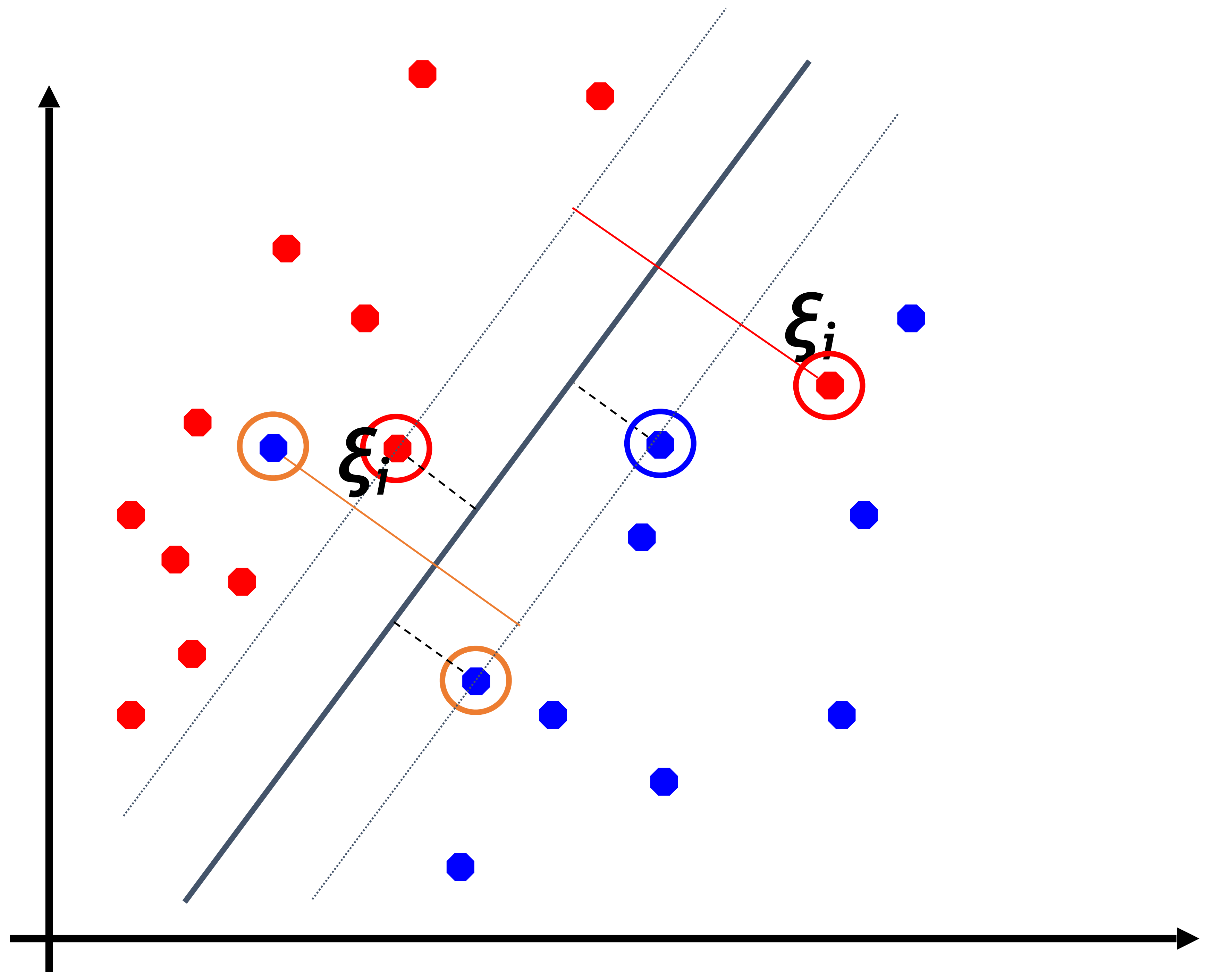
Soft Margin Classification

- **What if the training set is not linearly separable?**
- *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



Soft Margin Classification

- **What if the training set is not linearly separable?**
- *Slack variables ξ_i* can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



Soft Margin Classification Mathematically

- The old formulation:

$$\begin{array}{c}\textbf{Find w and b such that}\\ \Phi(\mathbf{w}) = \mathbf{w}^T\mathbf{w} \ \text{ is minimized}\\ \textbf{and for all } (\mathbf{x}_i, y_i),\ i=1..n: \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1\end{array}$$

- Modified formulation incorporates slack variables:

$$\begin{array}{c}\textbf{Find w and b such that}\\ \Phi(\mathbf{w}) = \mathbf{w}^T\mathbf{w} + C\Sigma\xi_i \ \text{ is minimized}\\ \textbf{and for all } (\mathbf{x}_i, y_i),\ i=1..n: \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0\end{array}$$

- Parameter C can be viewed as a way to control overfitting: it "trades off" the relative importance of maximizing the margin and fitting the training data.

Soft Margin Classification – Solution

- Dual problem is identical to separable case (would *not* be identical if the 2-norm penalty for slack variables $C\Sigma\xi_i^2$ was used in primal objective, we would need additional Lagrange multipliers for slack variables):

$$\boxed{\begin{array}{c}\textbf{Find } \boldsymbol{a_1 \ldots a_N} \textbf{ such that} \\ \mathbf{Q}(\alpha) = \Sigma a_i - \tfrac{1}{2}\Sigma\Sigma a_i a_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \text{ is maximized and} \\ \textbf{(1)}\ \ \Sigma a_i y_i = 0 \\ \textbf{(2)}\ \ 0 \le a_i \le C \textbf{ for all } a_i \end{array}}$$

- Again, x_i with non-zero a_i will be support vectors.
- Solution to the dual problem is:

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

$$b = y_k(1 - \xi_k) - \Sigma a_i y_i \mathbf{x}_i^T \mathbf{x}_k \quad \text{for any } k \text{ s.t. } a_k > 0$$

Again, we don't need to compute w explicitly for classification:

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

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 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 inner products:

Find $a_1 \ldots a_N$ such that

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

(1) $\Sigma a_i y_i = 0$

(2) $0 \leq a_i \leq C$ for all a_i

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

Solving the Optimization Problem
Solution 1: SGD and PEGASOS

- We can train the primal form of SVM using the SGD technique
- SVM objective if formulated as an L1-penalty unconstrained minimization

$$\min_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{(\mathbf{x},y)\in S} \ell(\mathbf{w};(\mathbf{x},y)) \ ,$$

$$\ell(\mathbf{w};(\mathbf{x},y)) \ = \ \max\{0, 1 - y\,\langle \mathbf{w}, \mathbf{x}\rangle\} \ ,$$

$$S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m, \text{ where } \mathbf{x}_i \in \mathbb{R}^n \text{ and } y_i \in \{+1, -1\}$$

- L1 penalty acts as the inclusion of constraint in the minization problem

Solving the Optimization Problem
Solution 1: SGD and PEGASOS

- We can minimize the unconstrained L1 penalty objective by taking derivatives w.r.t. w in a gradient descent approach
- The derivatives involve derivation of L1 ABS operator not differentiable.
- Make use of a Sub-Gradient technique
- Consider one sample i at each iteration t

$$f(\mathbf{w}; i_t) \;=\; \frac{\lambda}{2}\|\mathbf{w}\|^2 + \ell(\mathbf{w}; (\mathbf{x}_{i_t}, y_{i_t}))$$

- Sub gradient w.r.t. w is

$$\nabla_t = \lambda\,\mathbf{w}_t - \mathbb{1}[y_{i_t}\,\langle\mathbf{w}_t, \mathbf{x}_{i_t}\rangle < 1]\,y_{i_t}\mathbf{x}_{i_t}$$

- Update Rule

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla_t$$

- with learning rate

$$\mathbf{w}_{t+1} \leftarrow (1 - \frac{1}{t})\mathbf{w}_t + \eta_t \mathbb{1}[y_{i_t}\,\langle\mathbf{w}_t, \mathbf{x}_{i_t}\rangle < 1]\,y_{i_t}\mathbf{x}_{i_t}$$

PEGASOS FULL ALGORITHM

Shalev-Shwartz, S., Singer, Y., Srebro, N. et al. Pegasos: primal estimated sub-gradient solver for SVM Math. Program. (2011) 127: 3.

INPUT: S, λ, T

INITIALIZE: Set $\mathbf{w}_1 = 0$

FOR $t = 1, 2, \ldots, T$

  Choose $i_t \in \{1, \ldots, |S|\}$ uniformly at random.

  Set $\eta_t = \frac{1}{\lambda t}$

  If $y_{i_t} \langle \mathbf{w}_t, \mathbf{x}_{i_t} \rangle < 1$, then:

    Set $\mathbf{w}_{t+1} \leftarrow (1 - \eta_t \lambda)\mathbf{w}_t + \eta_t y_{i_t} \mathbf{x}_{i_t}$

  Else (if $y_{i_t} \langle \mathbf{w}_t, \mathbf{x}_{i_t} \rangle \geq 1$):

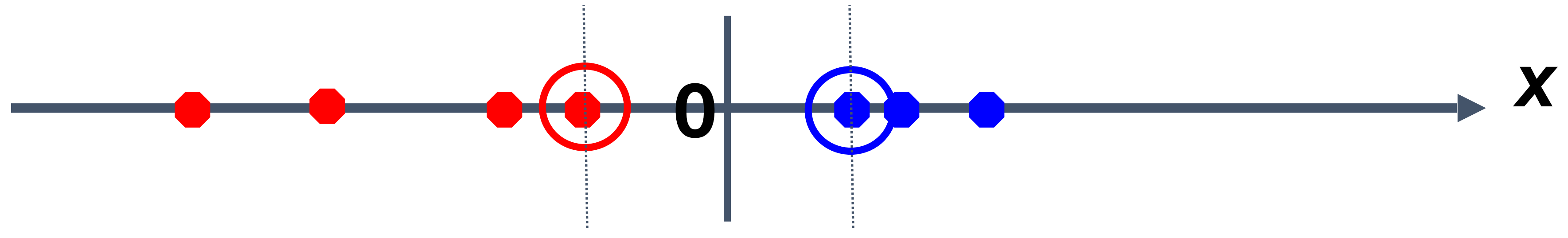
    Set $\mathbf{w}_{t+1} \leftarrow (1 - \eta_t \lambda)\mathbf{w}_t$

  [Optional: $\mathbf{w}_{t+1} \leftarrow \min \left\{1, \frac{1/\sqrt{\lambda}}{\|\mathbf{w}_{t+1}\|}\right\} \mathbf{w}_{t+1}$]

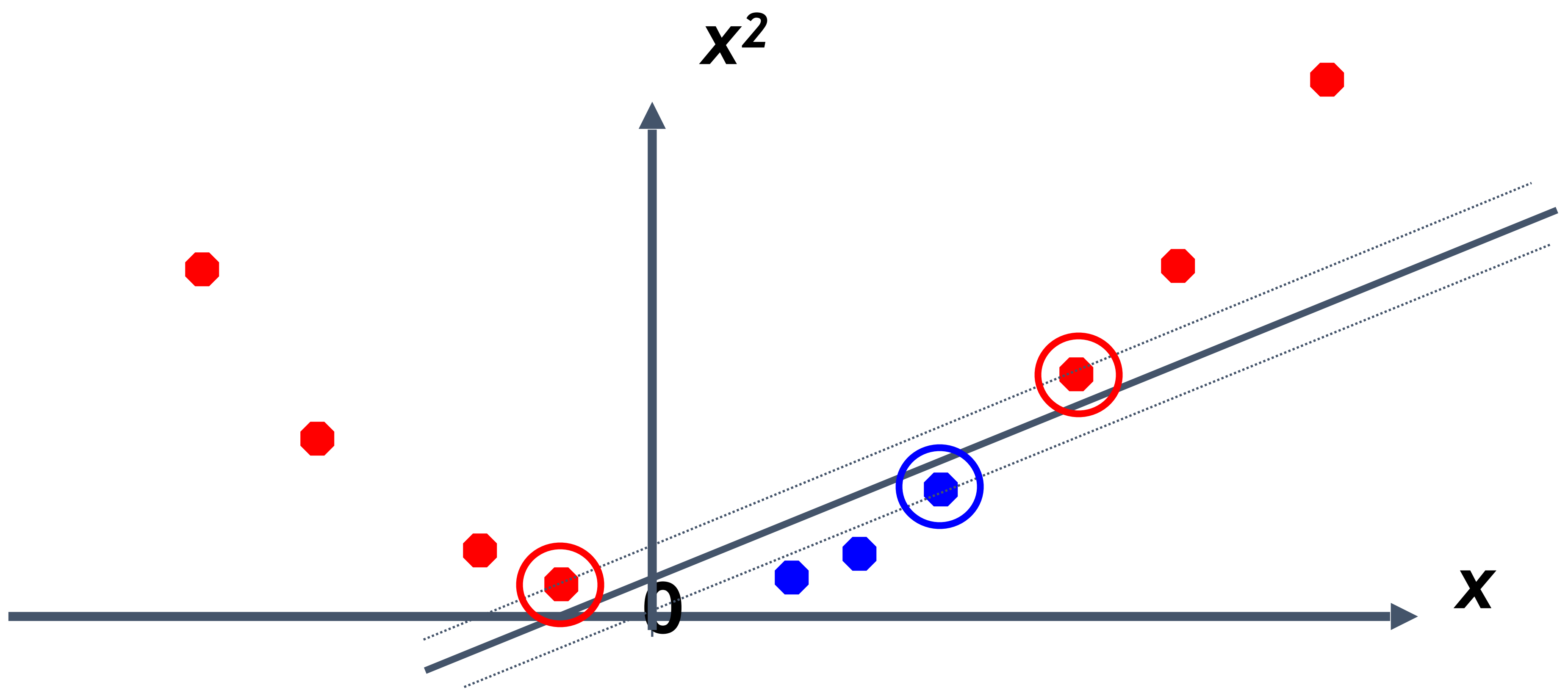
OUTPUT: $\mathbf{w}_{T+1}$

Non-linear SVMs

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

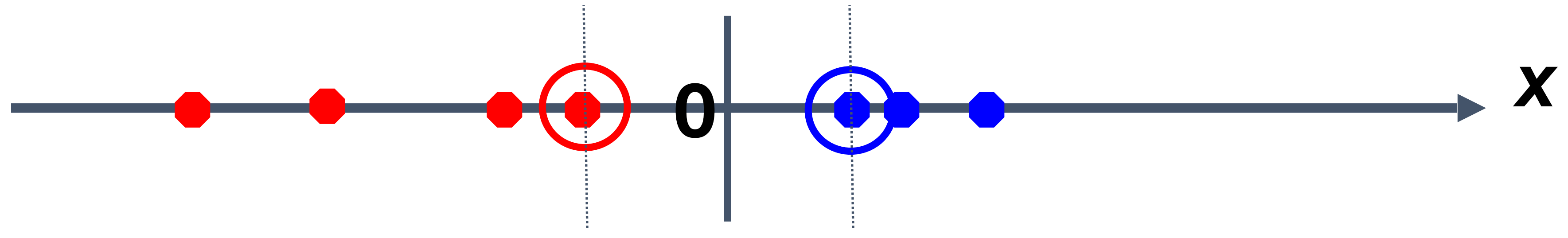


- But what are we going to do if the dataset is just too hard?



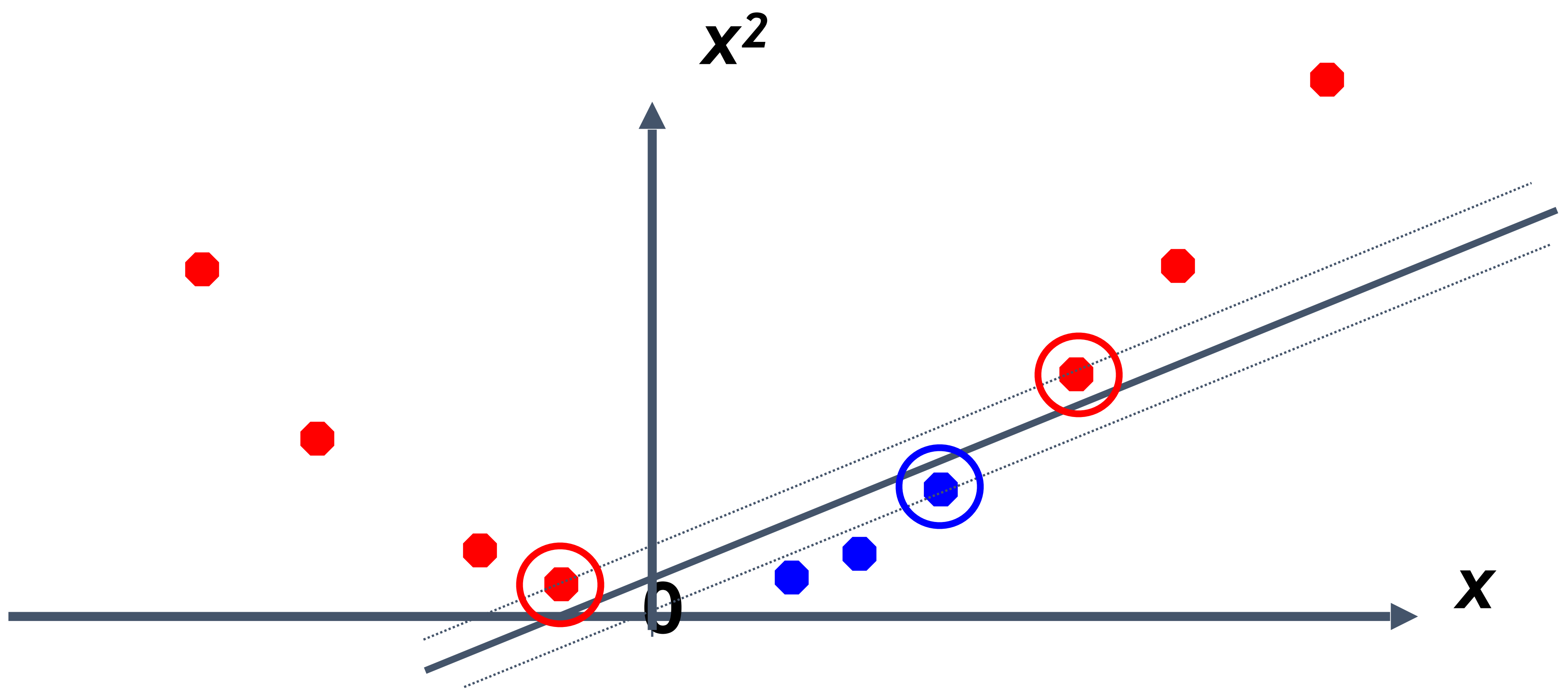
Non-linear SVMs

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



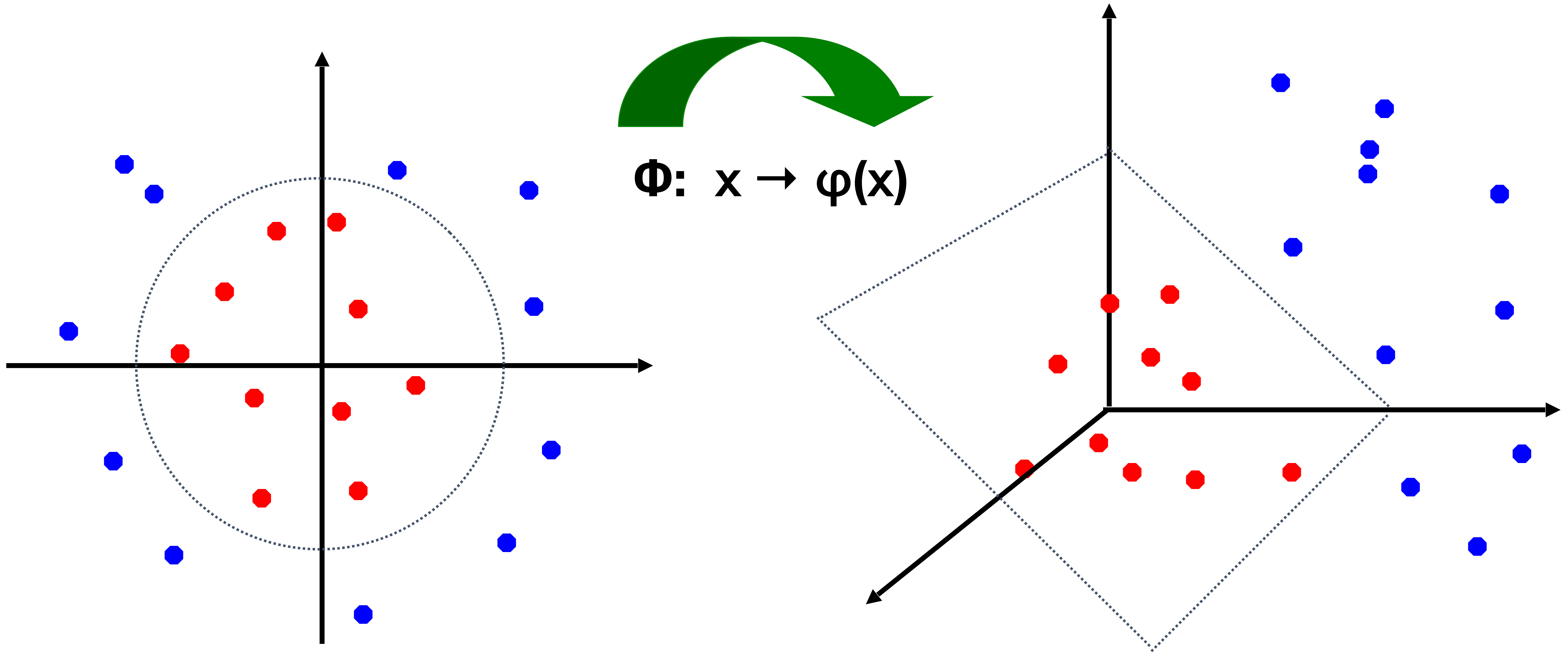
- But what are we going to do if the dataset is just too hard?

- 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 inner product between vectors $K(\mathrm{x}_i,\mathrm{x}_j)=\mathrm{x}_i^{\mathrm{T}}\mathrm{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathrm{x} \rightarrow \phi(\mathrm{x})$, the inner product becomes:

$K(\mathrm{x}_i,\mathrm{x}_j)= \phi(\mathrm{x}_i)^{\mathrm{T}}\phi(\mathrm{x}_j)$

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

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

  Need to show that $K(\mathrm{x}_i,\mathrm{x}_j)= \phi(\mathrm{x}_i)^{\mathrm{T}}\phi(\mathrm{x}_j)$:

  $K(\mathrm{x}_i,\mathrm{x}_j)=(1 + \mathrm{x}_i^{\mathrm{T}}\mathrm{x}_j)^2,= 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$

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

  $= \phi(\mathrm{x}_i)^{\mathrm{T}}\phi(\mathrm{x}_j)$, where $\phi(\mathrm{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$

- Thus, a kernel function *implicitly* maps data to a high-dimensional space (without the need to compute each $\phi(\mathrm{x})$ explicitly).

What Functions are Kernels?

- For some functions $K(x_i,x_j)$ checking that $K(x_i,x_j)= \phi(x_i)^T\phi(x_j)$ 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:

$K(\mathbf{x}_1,\mathbf{x}_1)$	$K(\mathbf{x}_1,\mathbf{x}_2)$	$K(\mathbf{x}_1,\mathbf{x}_3)$	…	$K(\mathbf{x}_1,\mathbf{x}_n)$
$K(\mathbf{x}_2,\mathbf{x}_1)$	$K(\mathbf{x}_2,\mathbf{x}_2)$	$K(\mathbf{x}_2,\mathbf{x}_3)$		$K(\mathbf{x}_2,\mathbf{x}_n)$
…	…	…	…	…
$K(\mathbf{x}_n,\mathbf{x}_1)$	$K(\mathbf{x}_n,\mathbf{x}_2)$	$K(\mathbf{x}_n,\mathbf{x}_3)$	…	$K(\mathbf{x}_n,\mathbf{x}_n)$

Examples of Kernel Functions

- Linear: $K(\mathrm{x}_i,\mathrm{x}_j)= \mathrm{x}_i^{\mathrm{T}}\mathrm{x}_j$
  - Mapping Φ: x → φ(x), where φ(x) is x itself

- Polynomial of power p: $K(\mathrm{x}_i,\mathrm{x}_j)= (1+ \mathrm{x}_i^{\mathrm{T}}\mathrm{x}_j)^p$
  - Mapping Φ: x → φ(x), where φ(x) has $\binom{d+p}{p}$ dimensions

- Gaussian (radial-basis function): $K(\mathrm{x}_i,\mathrm{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$
  - Mapping Φ: x → φ(x), where φ(x) is *infinite-dimensional*: every point is mapped to *a function* (a Gaussian); combination of functions for support vectors is the separator.

- Higher-dimensional space still has *intrinsic* dimensionality *d* (the mapping is not *onto*), but linear separators in it correspond to *non-linear* separators in original space.

Non-linear SVMs Mathematically

- Dual problem formulation:

$$\boxed{\begin{array}{c}\textbf{Find } a_1 \ldots a_n \textbf{ such that} \\ Q(\alpha) = \Sigma a_i - \tfrac{1}{2}\Sigma\Sigma a_i a_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \text{ is maximized and} \\ (1)\ \ \Sigma a_i y_i = 0 \\ (2)\ a_i \geq 0 \text{ for all } a_i\end{array}}$$

- The solution is:

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

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

Pegasos and NON linear SVM

EMBEDDING + LINEAR

- Use kernels to create an Affinity Matrix
- Use Dimensionality Reduction to obtain an Euclidean Embedding according to kernels affinity
- Train a linear SVM in the Embedded space with Pegasos.

- DRAWBACK if you use LLE or Laplacian EigenMap:
- Need to embed also test elements in the embedded representation

NON LINEAR PEGASOS

Check the paper Section 4 ☺️

SVM applications

- SVMs were originally proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in late 1990s.
- SVMs are currently among the best performers for a number of classification tasks ranging from text to genomic data.
- SVMs can be applied to complex data types beyond feature vectors (e.g. graphs, sequences, relational data) by designing kernel functions for such data.
- SVM techniques have been extended to a number of tasks such as regression [Vapnik *et al.* '97], principal component analysis [Schölkopf *et al.* '99], etc.
- Most popular optimization algorithms for SVMs use *decomposition* to hill-climb over a subset of α_i's at a time, e.g. SMO [Platt '99] and [Joachims '99]
- Tuning SVMs remains a black art: selecting a specific kernel and parameters is usually done in a try-and-see manner.