CS440/ECE448: Intro to Artificial Intelligence

Lecture 27: Support vector machines

Prof. Julia Hockenmaier
juliahmr@illinois.edu

http://cs.illinois.edu/fa11/cs440

Class admin

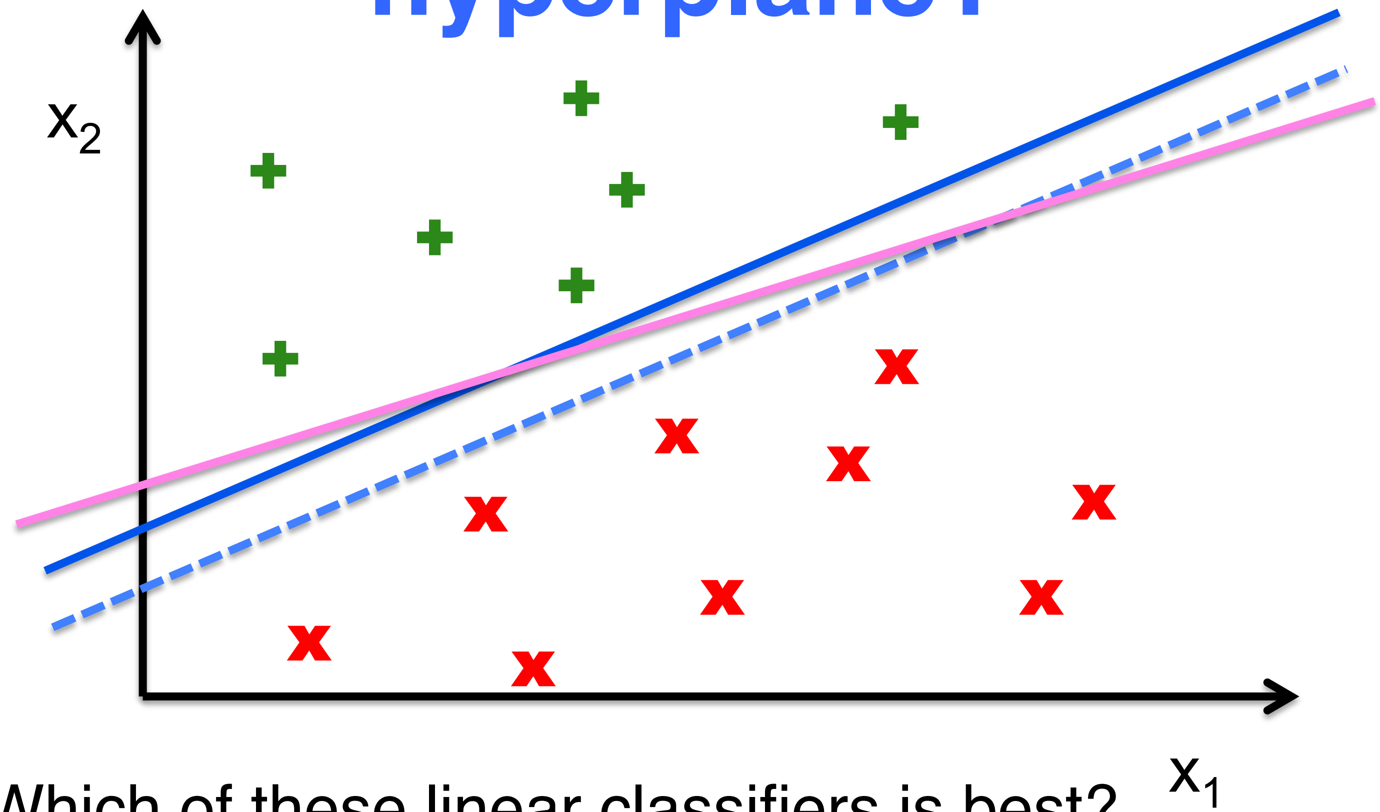
Final exam: Friday, May 13, 7pm.
Conflict exam: Thursday, May 12, 10am in 3401.

The last lecture is on Thursday
(review session)

I will offer additional review sessions on Tuesday and Thursday before the final

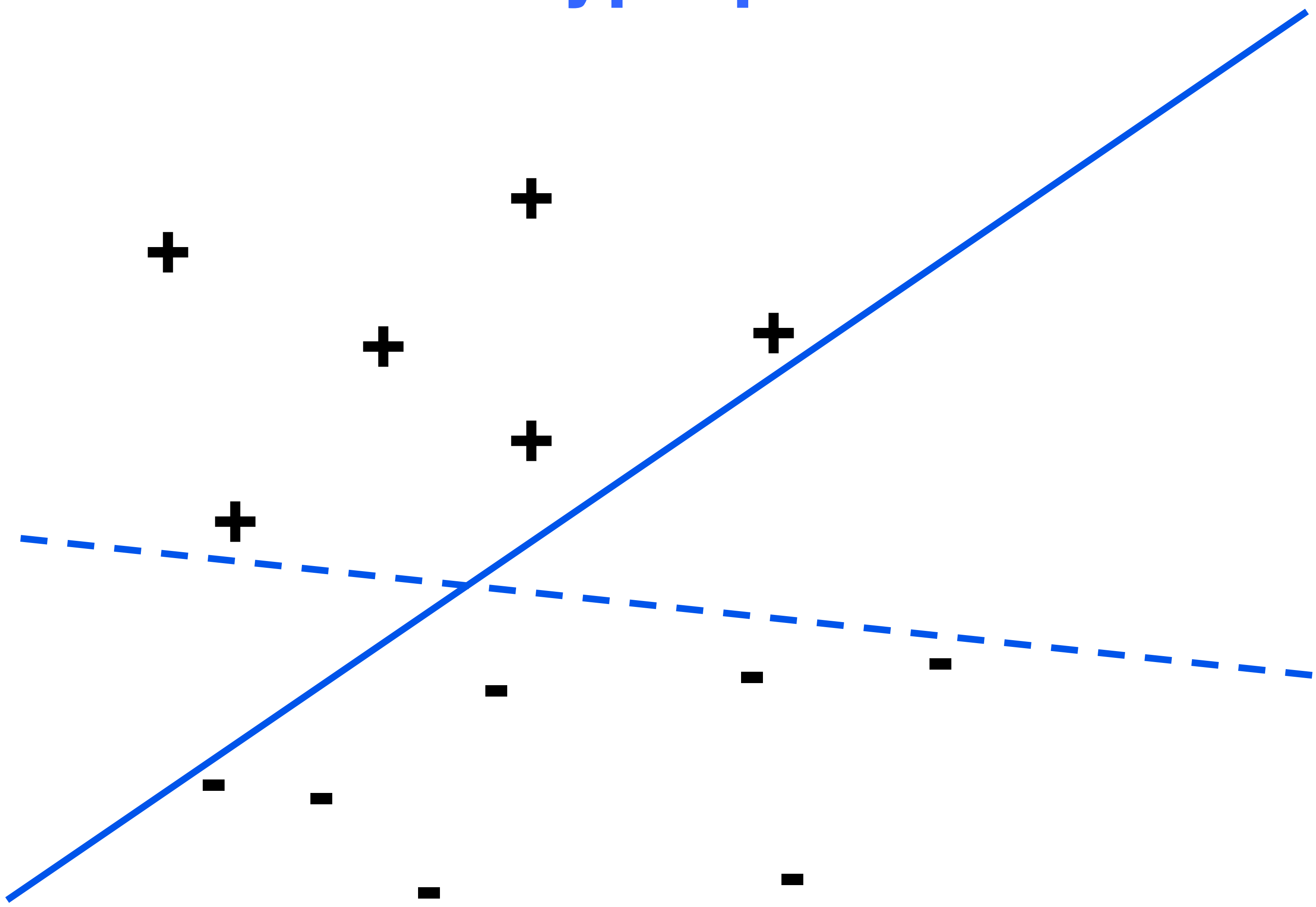
Large margin classifiers

What is the best separating hyperplane?

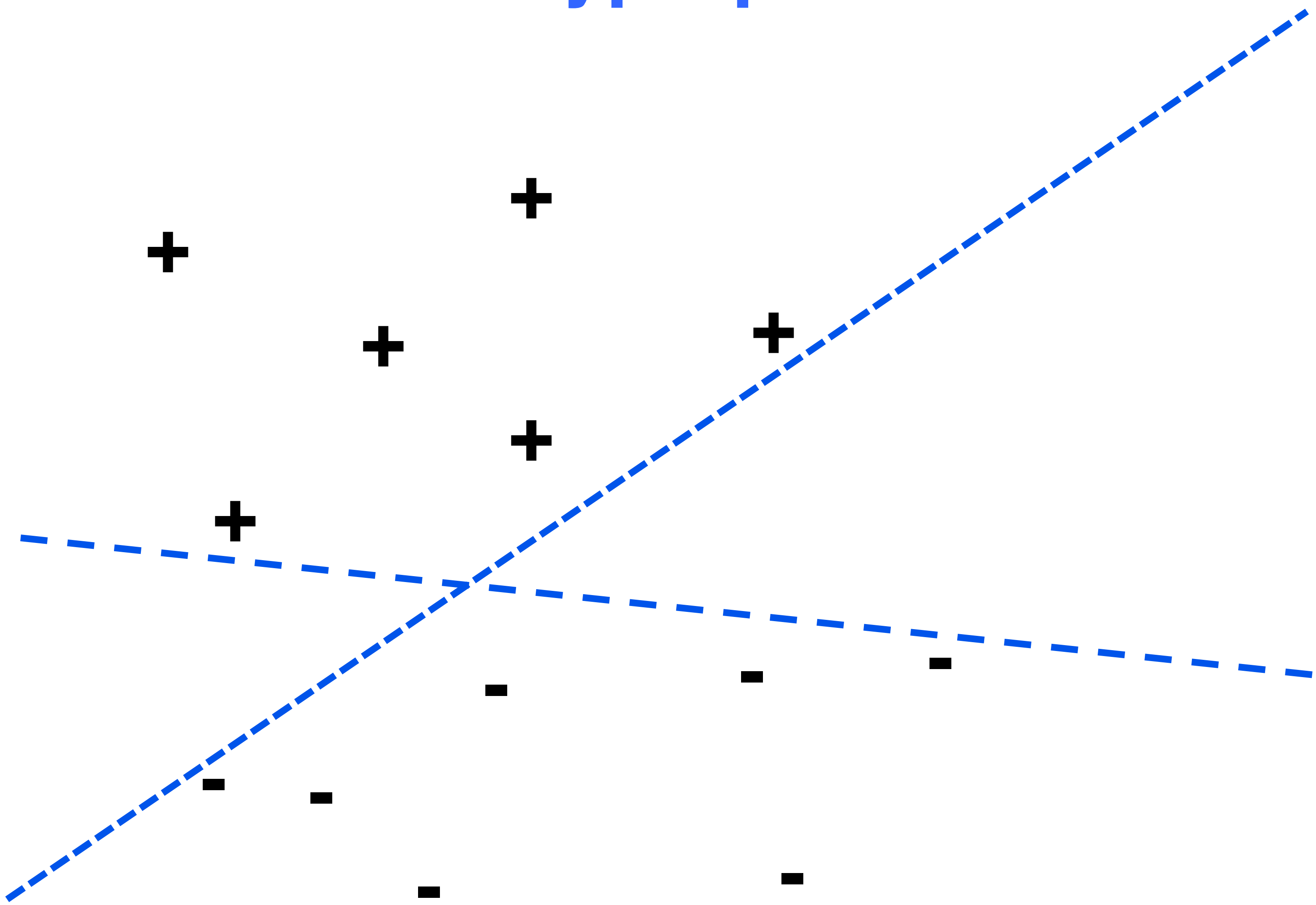


Which of these linear classifiers is best?
How can we choose?

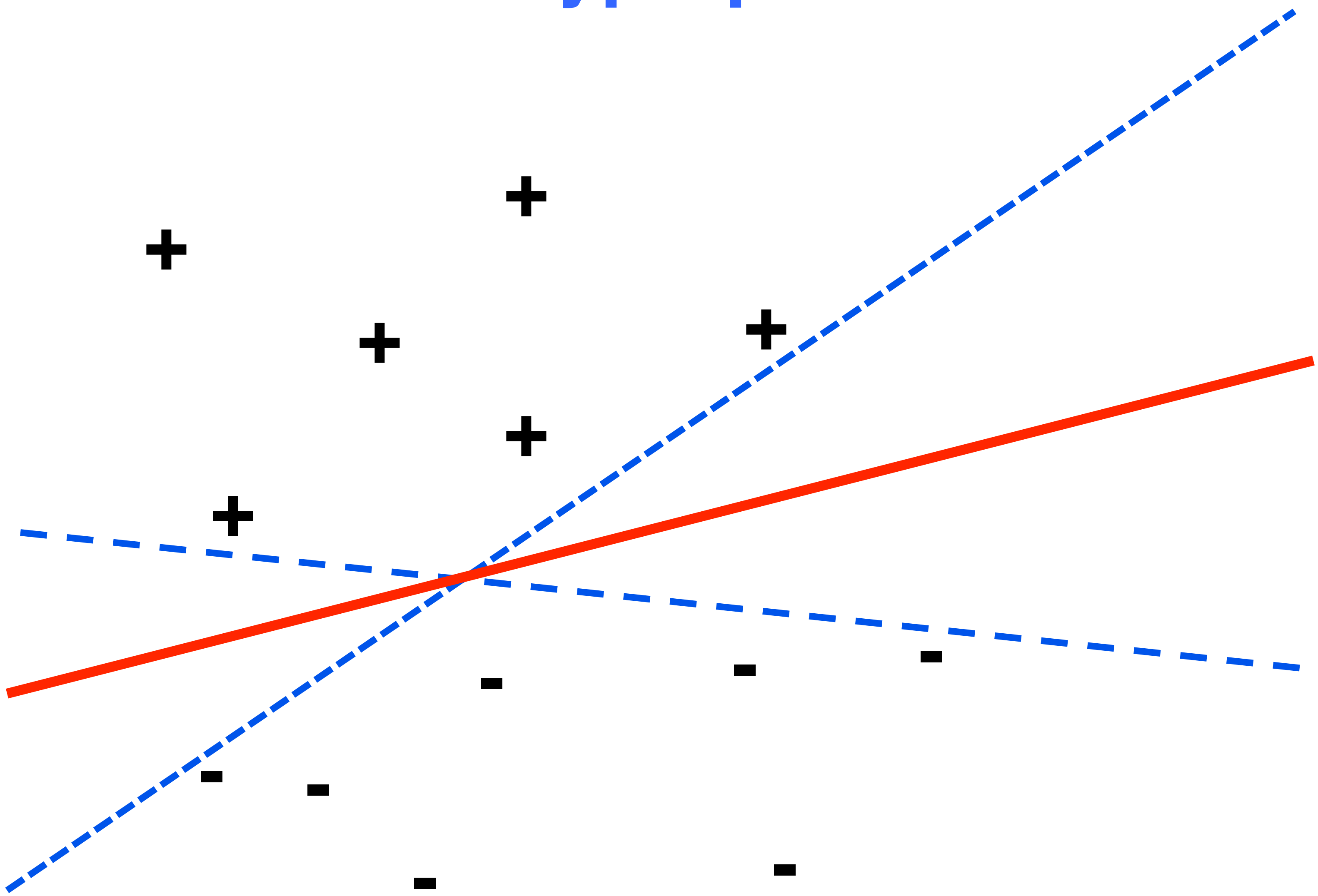
What's the Best Separating Hyperplane?



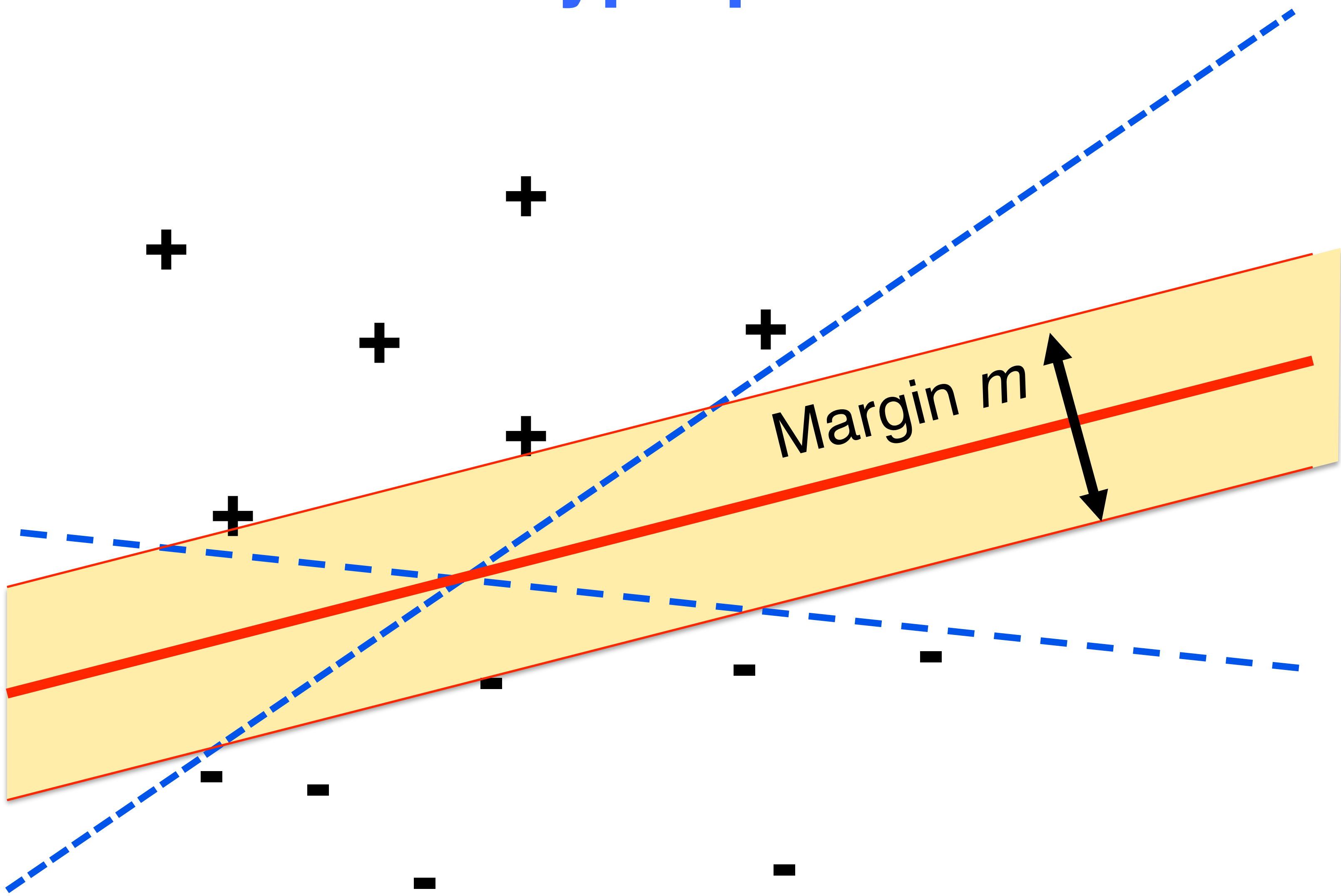
What's the Best Separating Hyperplane?



What's the Best Separating Hyperplane?



What's the Best Separating Hyperplane?



Maximum margin classifier

We want to find the classifier whose decision boundary is furthest away from any data point. (this classifier has the largest margin).

This additional requirement (*bias)* reduces the *variance* (i.e. reduces overfitting).

Perceptrons and SVMs: differences in notation

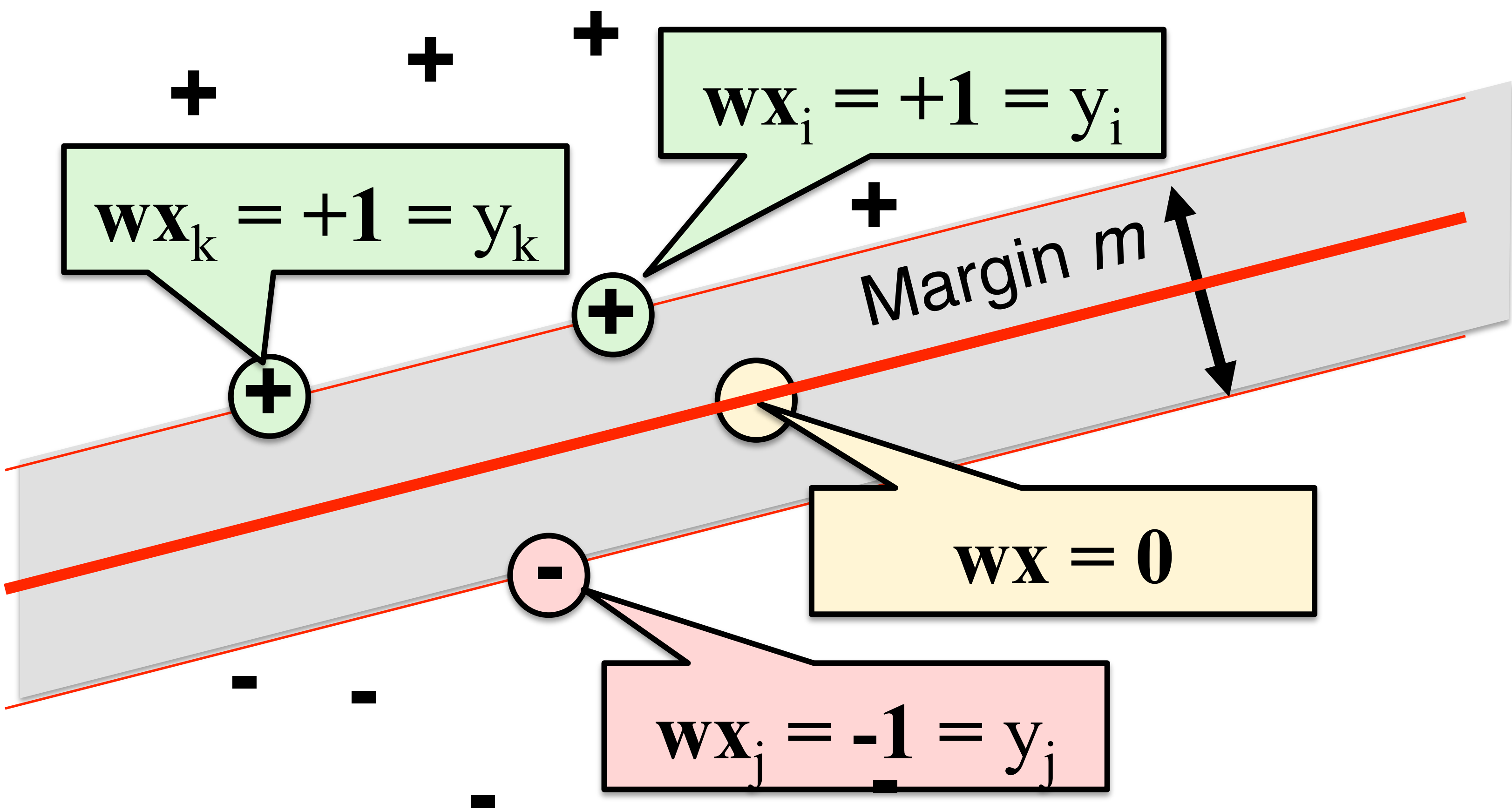
Perceptrons:

- Positive examples: $y_j = +1$; negative examples: $y_j = 0$
- Weight vector has bias term w_0 (x_0 = dummy value 1)
- Decision boundary: $\mathbf{wx} = 0$

SVMs/Large Margin classifiers:

- Positive examples: $y_j = +1$; negative examples: $y_j = -1$
- Explicit bias term b; weight vector $\mathbf{w} = (w_1 \ldots w_n)$
- Decision boundary $\mathbf{wx} + b = 0$

The maximum margin decision boundary



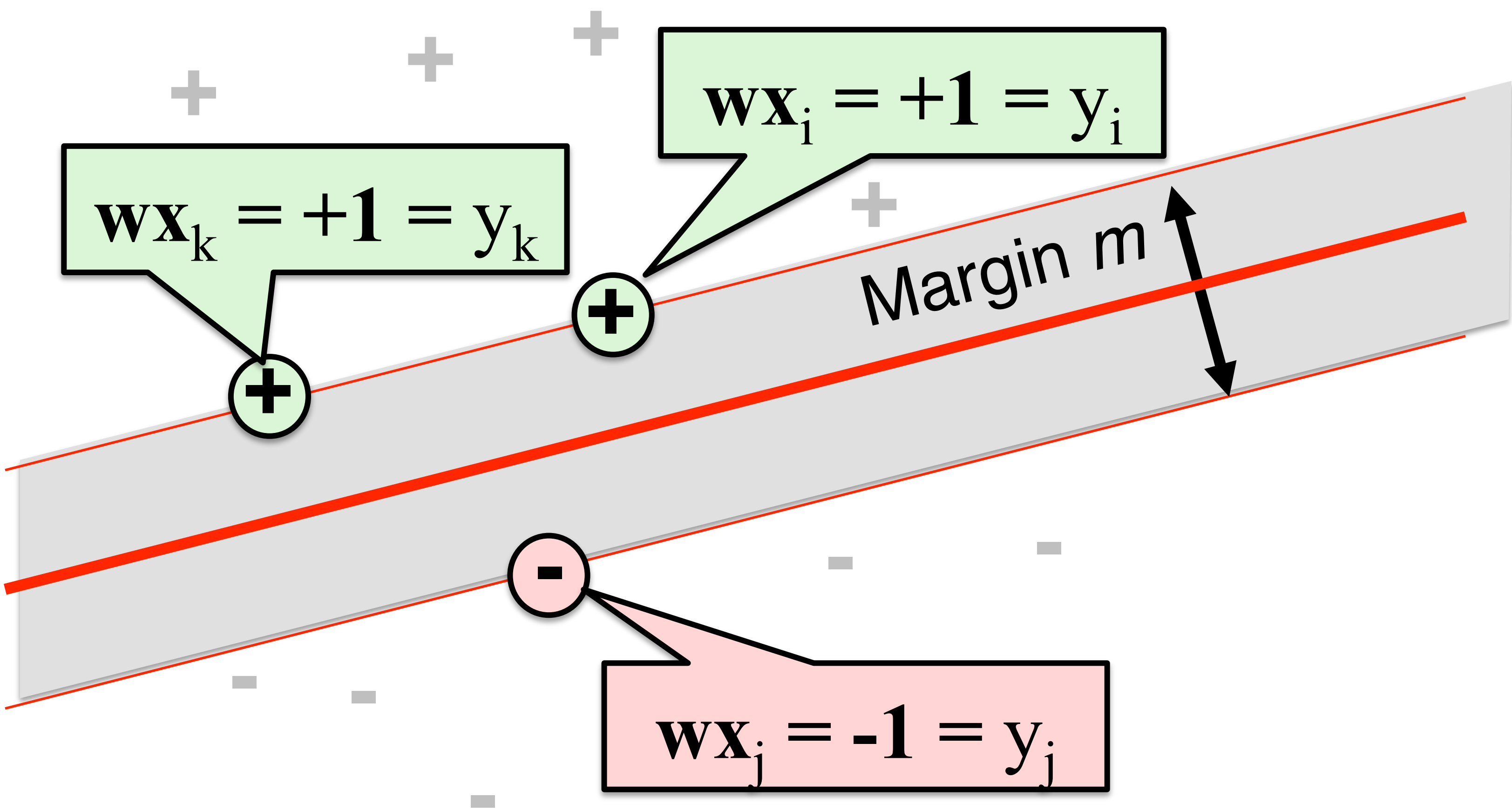
The maximum margin decision boundary…

… is defined by two parallel hyperplanes:

- one that goes through the **positive** data points $(y_j = +1)$ that are closest to the decision boundary, and

- one that goes through the **negative** data points $(y_j = -1)$ that are closest to the decision boundary.

Support vectors



Support vectors

We can express the separating hyperplane in terms of the data points $\mathbf{x}_j$ that are closest to the decision boundary.

These data points are called the **support vectors.**

The primal representation

The data items $\mathbf{x} = (x_1 \ldots x_n)$ have *n* features
The weight vector $\mathbf{w} = (w_1 \ldots w_n)$ has *n* elements

Learning:
Find a weight w_j for each feature x_j

Classification:
Evaluate $\mathbf{wx}$

The dual representation

Equivalently, we can represent $\mathbf{w}$ as a linear combination of the items in the training data:

$$\mathbf{w} = \sum_j \alpha_j \mathbf{x}_j$$

Learning:

Find a weight α_j (≥ 0) for each data point $\mathbf{x}_\mathrm{j}$

This requires computing the inner product $\mathbf{x}_\mathrm{i}\mathbf{x}_\mathrm{j} = \langle \mathbf{x}_\mathrm{i}\, \mathbf{x}_\mathrm{j} \rangle$ between all data items $\mathbf{x}_\mathrm{i}\, \mathbf{x}_\mathrm{j}$

Support vectors

= the set of data points $\mathbf{x}_\mathrm{j}$ with non-zero weights α_j

Classifying test data

In the primal:

Compute inner product between weight vector and test item

$$\mathbf{wx} = \langle \mathbf{w}, \mathbf{x} \rangle$$

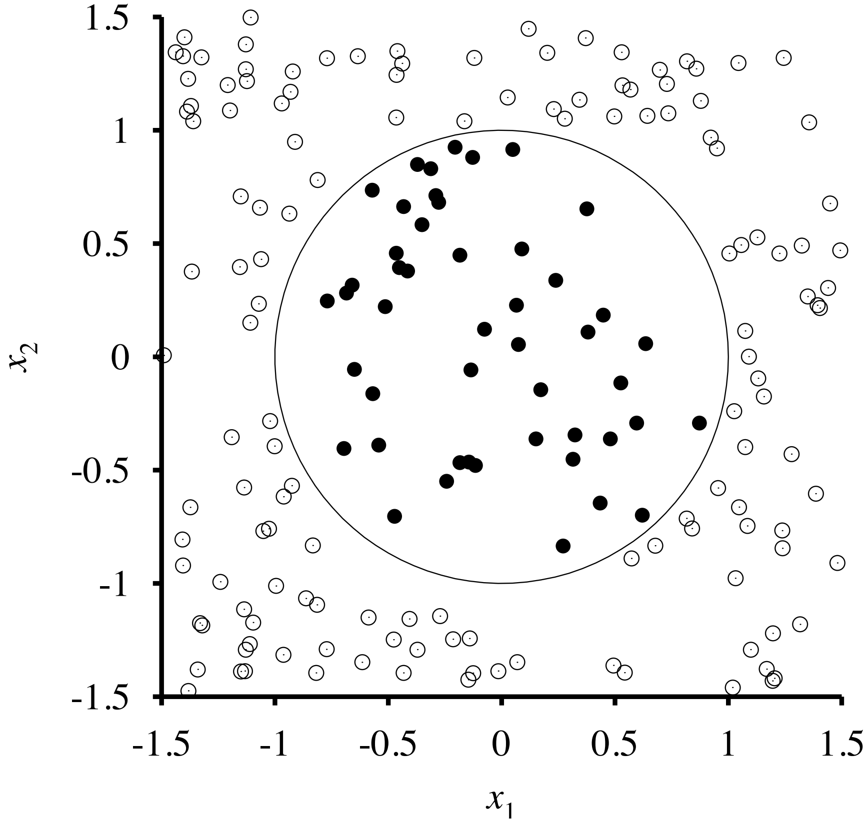
In the dual:

Compute inner product between support vectors and test item

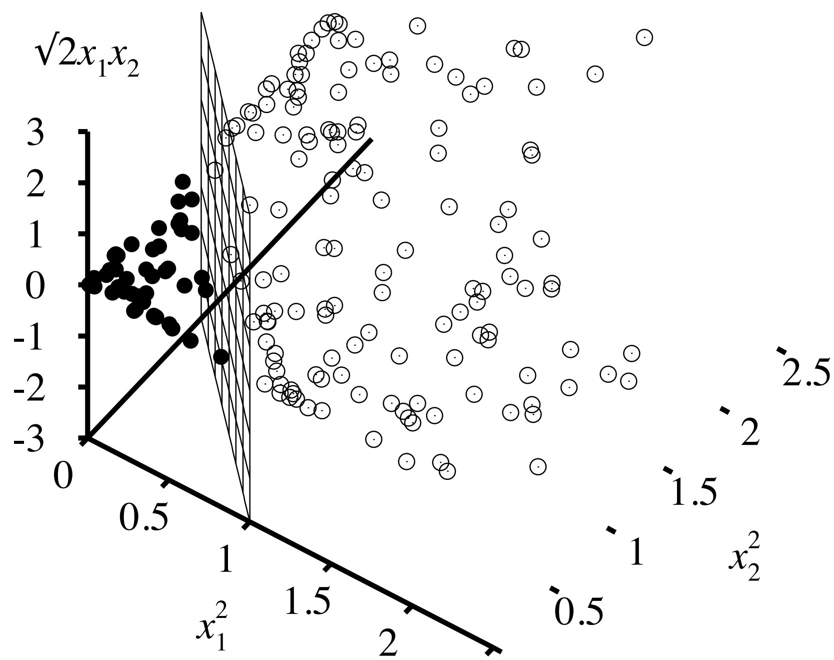
$$\mathbf{wx} = \langle \mathbf{w}, \mathbf{x} \rangle = \langle \sum_j \alpha_j x_j, \mathbf{x} \rangle = \sum_j \alpha_j \langle x_j, \mathbf{x} \rangle$$

The kernel trick

Linear separability violated…



Mapping to $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$



The kernel trick

N (independent) data points will always be linearly separable in *N-1* dimensions.

Insight 1: if we map each **x** to a point F(**x**) in a higher-dimensional feature space, the data become linearly separable

Insight 2: in the dual, we compute $\langle F(\mathbf{x}_i)\ F(\mathbf{x}_j) \rangle$. This can often be computed directly as a 'kernel' function $K(\mathbf{x}_i, \mathbf{x}_j)$

The kernel trick

What is K(x,y) ?

Anything we want; often polynomial kernels:

$(x \cdot y)^d$ Homogeneous polynomials

$(x \cdot y + 1)^d$ Complete polynomials

Condition: the kernel matrix **K** with $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ is positive semi-definite

In the dual, we now compute

$$\sum_{i=1}^{m} \alpha_i K(\mathbf{x}_i, \mathbf{x})$$