CS57300
PURDUE UNIVERSITY

OCTOBER 4, 2021

DATA MINING

LOGISTIC REGRESSION

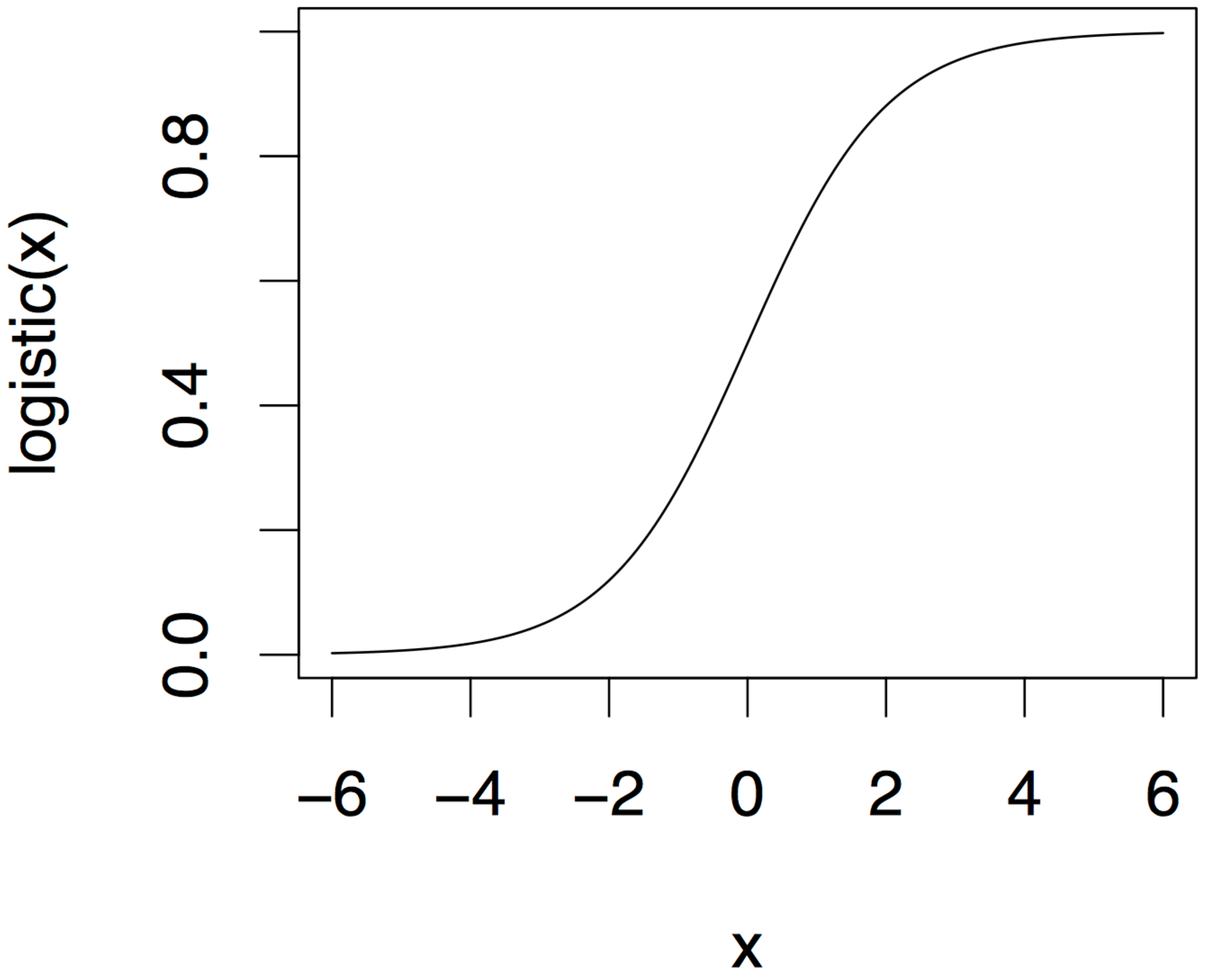
LOGISTIC REGRESSION

- Probabilistic classification
  - Output is the posterior (positive) class probability $P(y=1|\mathbf{x})$
  - Output is in the range [0, 1]
- Can we map the posterior class probability to another range that is easier to process?

LOGISTIC REGRESSION KNOWLEDGE REPRESENTATION

- p= $P(y=1|\mathbf{x}) = \dfrac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}+w_0)}}$
- Logistic function:

$$\text{logistic}(x) := \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$



LOGISTIC REGRESSION: LEARNING

- Model space: parametric model with the parameters being all possible [$\boldsymbol{w}$, w_0]
- Scoring function: Likelihood function

$$L(\mathbf{w}) = \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i)$$

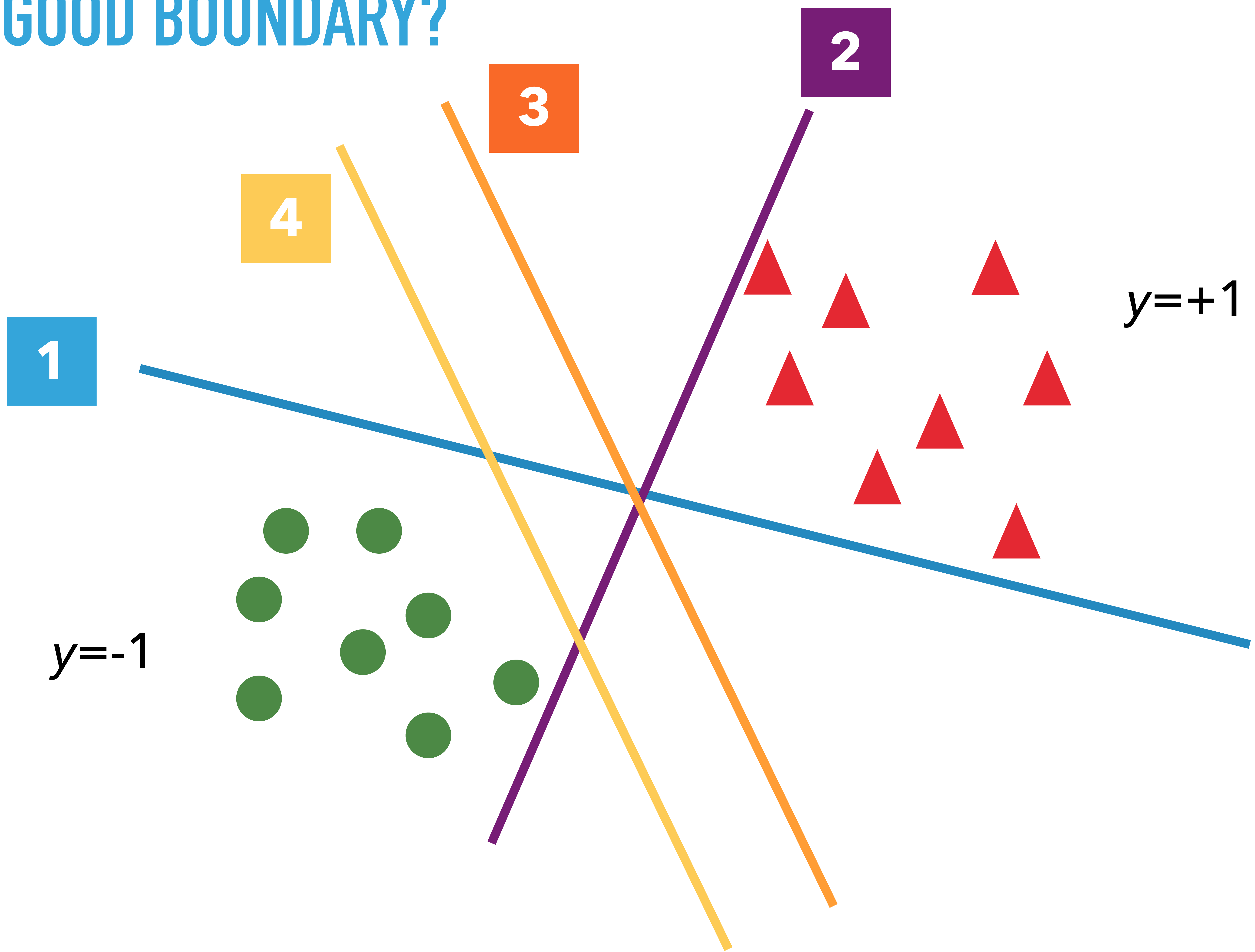
- Search
  - Take derivative respect to **w**
  - Concave function but can not get a closed form solution for the optimal parameters
  - Need new optimization methods!
  - More on this next week

SUPPORT VECTOR MACHINES

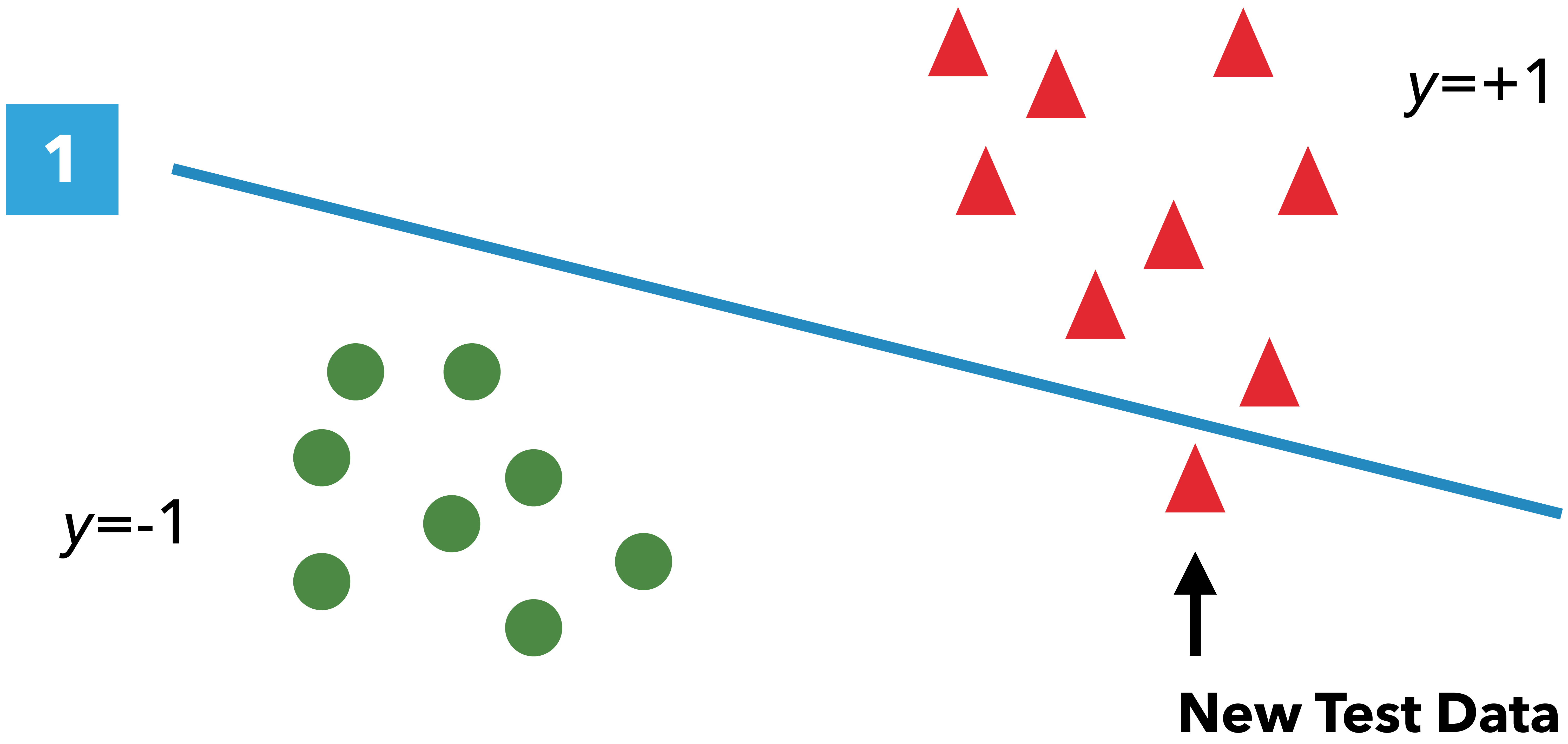
SUPPORT VECTOR MACHINES

- Discriminative classification
  - Output is the class label
  - Directly model the decision boundary
- Linear SVM
  - Parametric form: $y = sign\left[\sum_{i=1}^{m} w_i x_i + b\right]$
  - Decision boundaries are **hyperplanes** in the p-D space
  - Model space: different parameter values for ***w*** and *b*

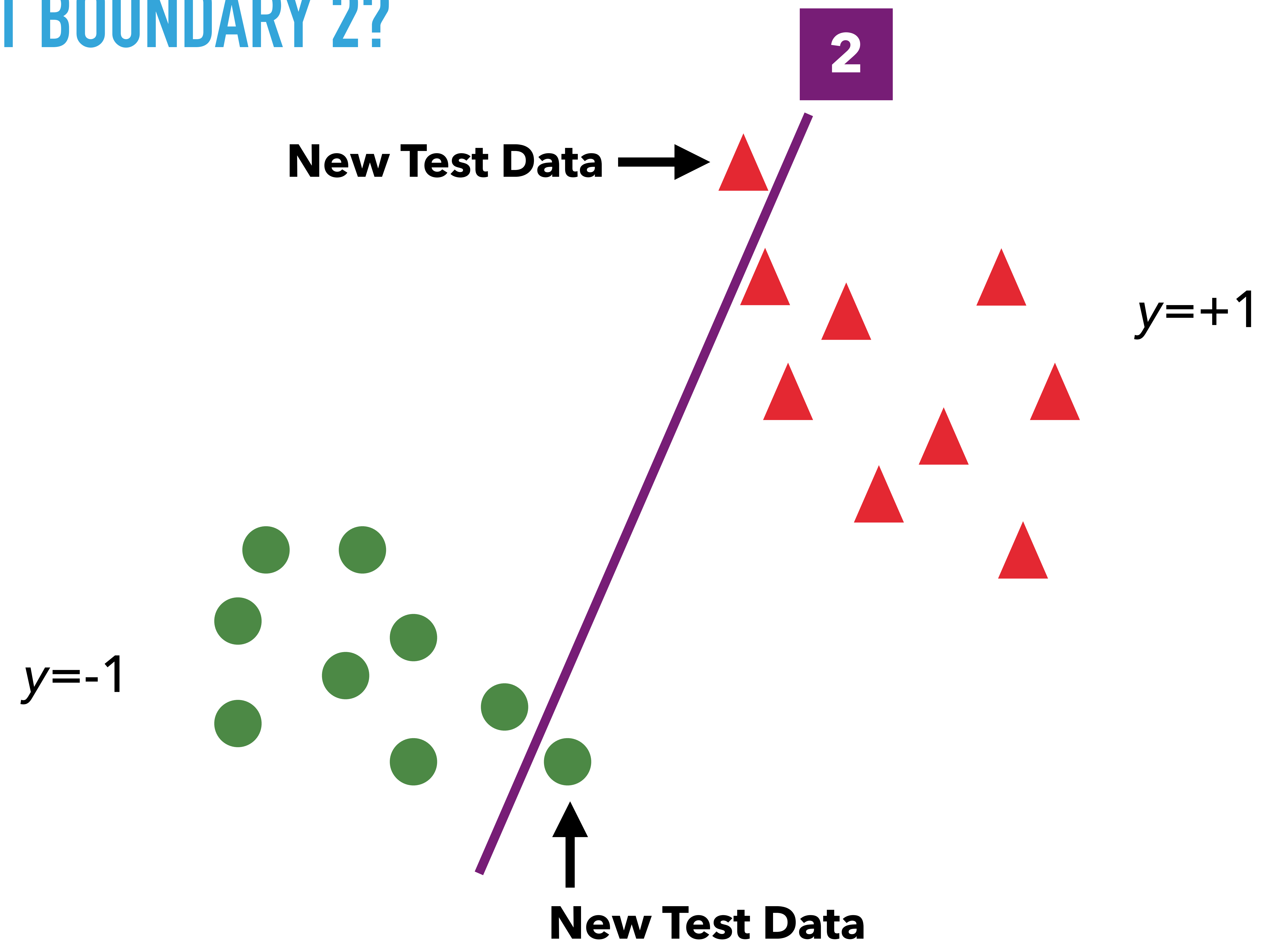
WHAT IS A GOOD BOUNDARY?



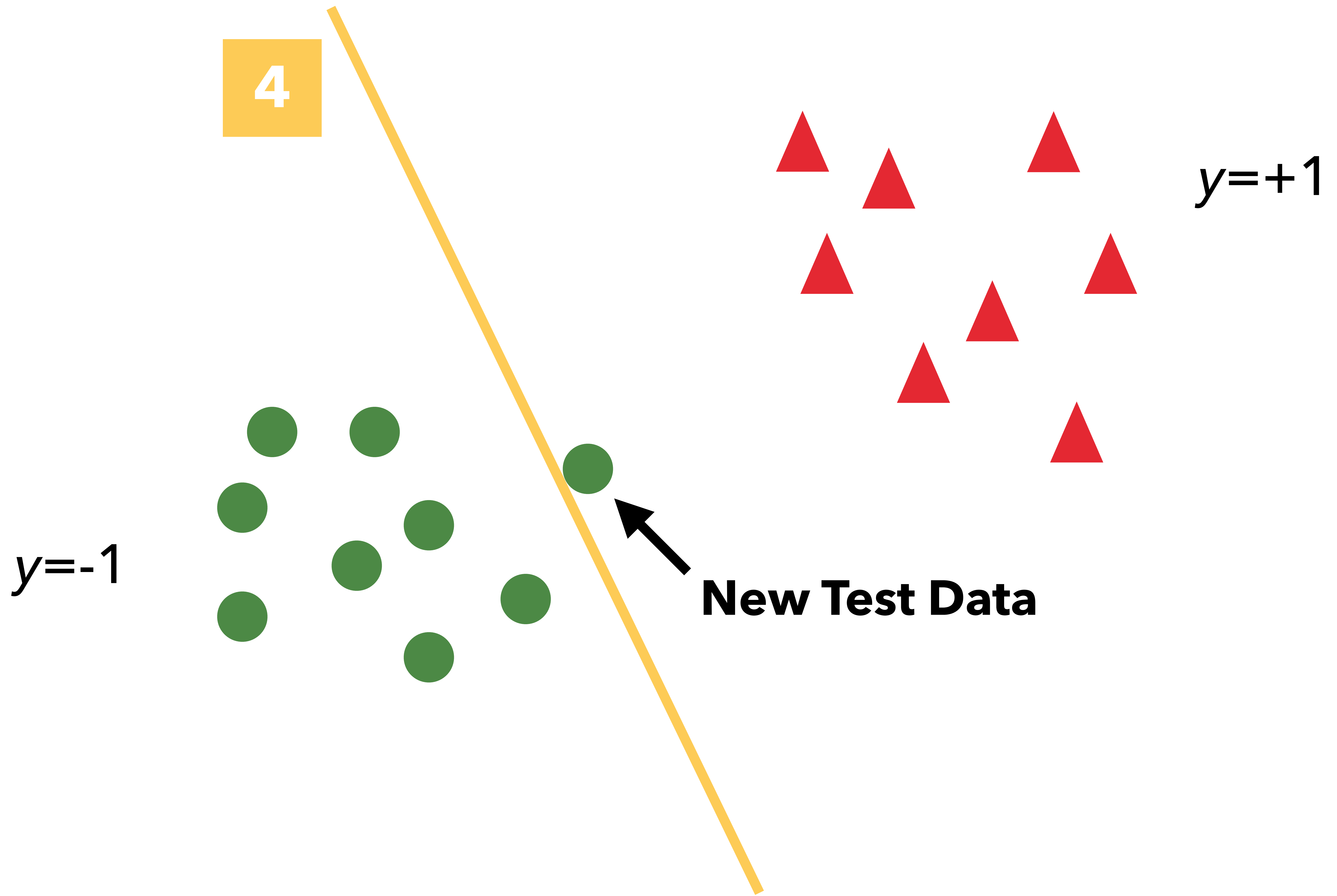
WHAT ABOUT BOUNDARY 1?



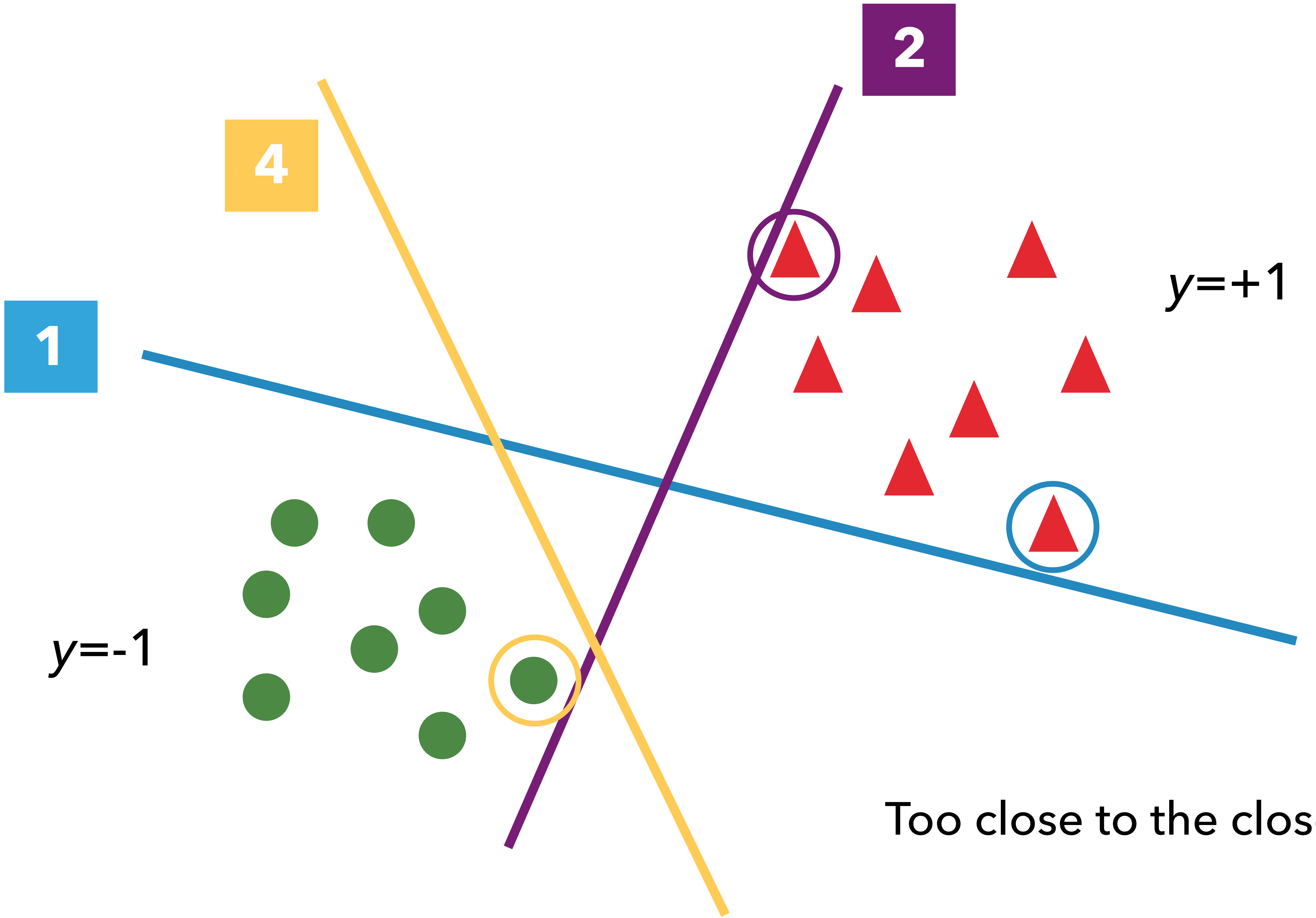
WHAT ABOUT BOUNDARY 2?



WHAT ABOUT BOUNDARY 4?



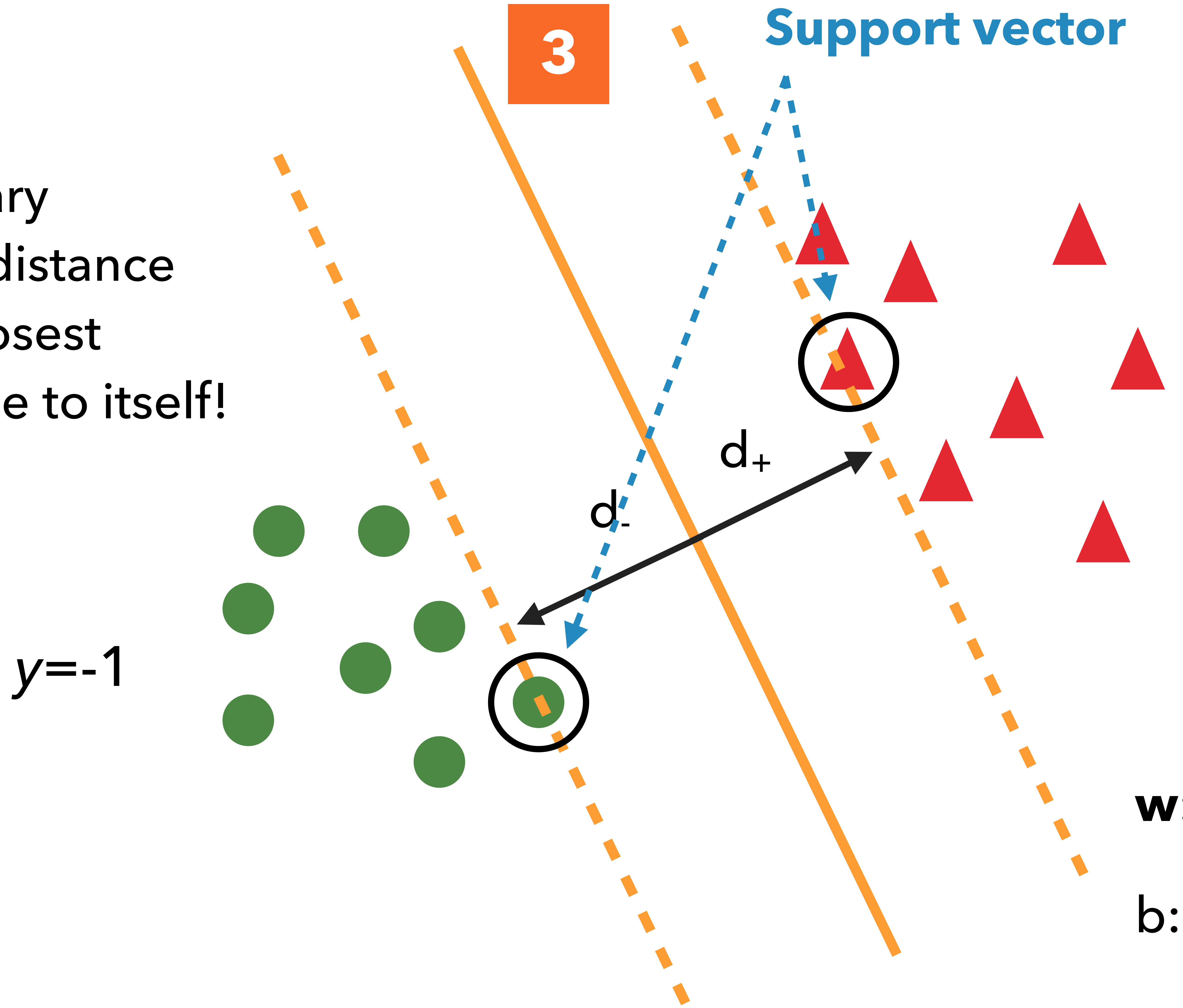
WHAT DOES BOUNDARY 1, 2, 4 HAVE IN COMMON?



Too close to the closest training example!

MOST ROBUST BOUNDARY

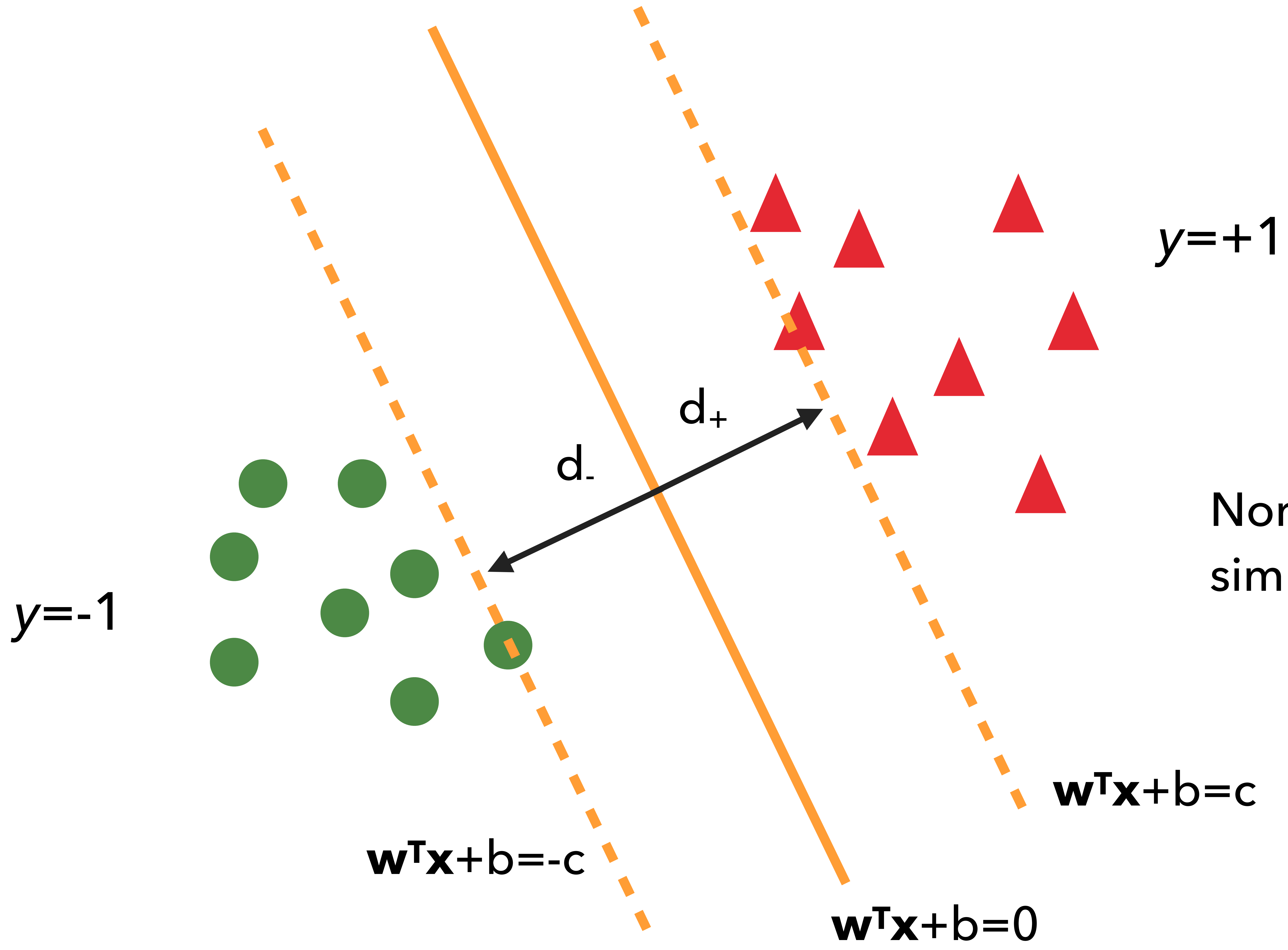
A good boundary maximizes the distance between the closest training example to itself!



w: maximizes the margin (d_++ d_-)

b: makes d_+= d_-

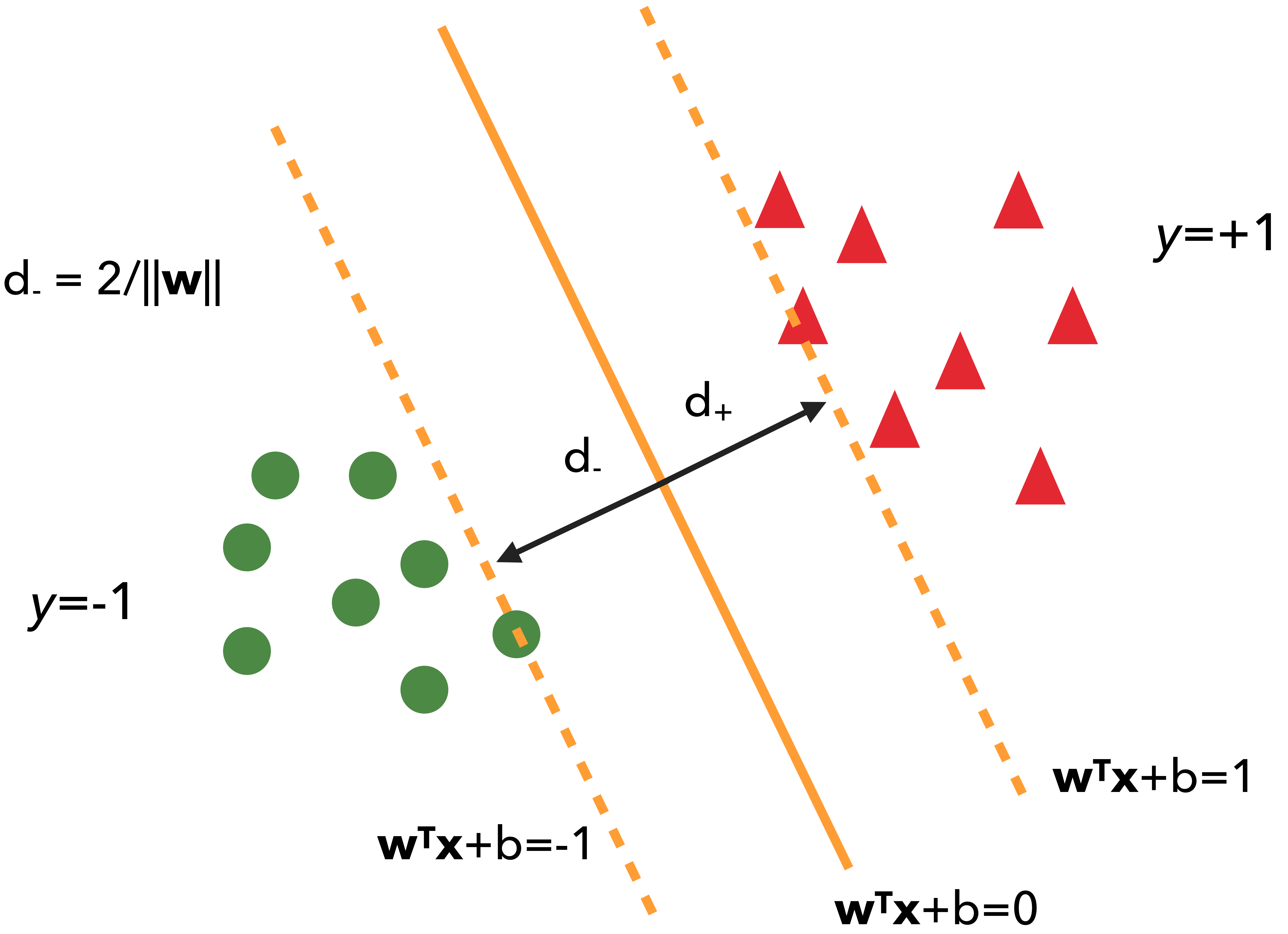
NORMALIZATION



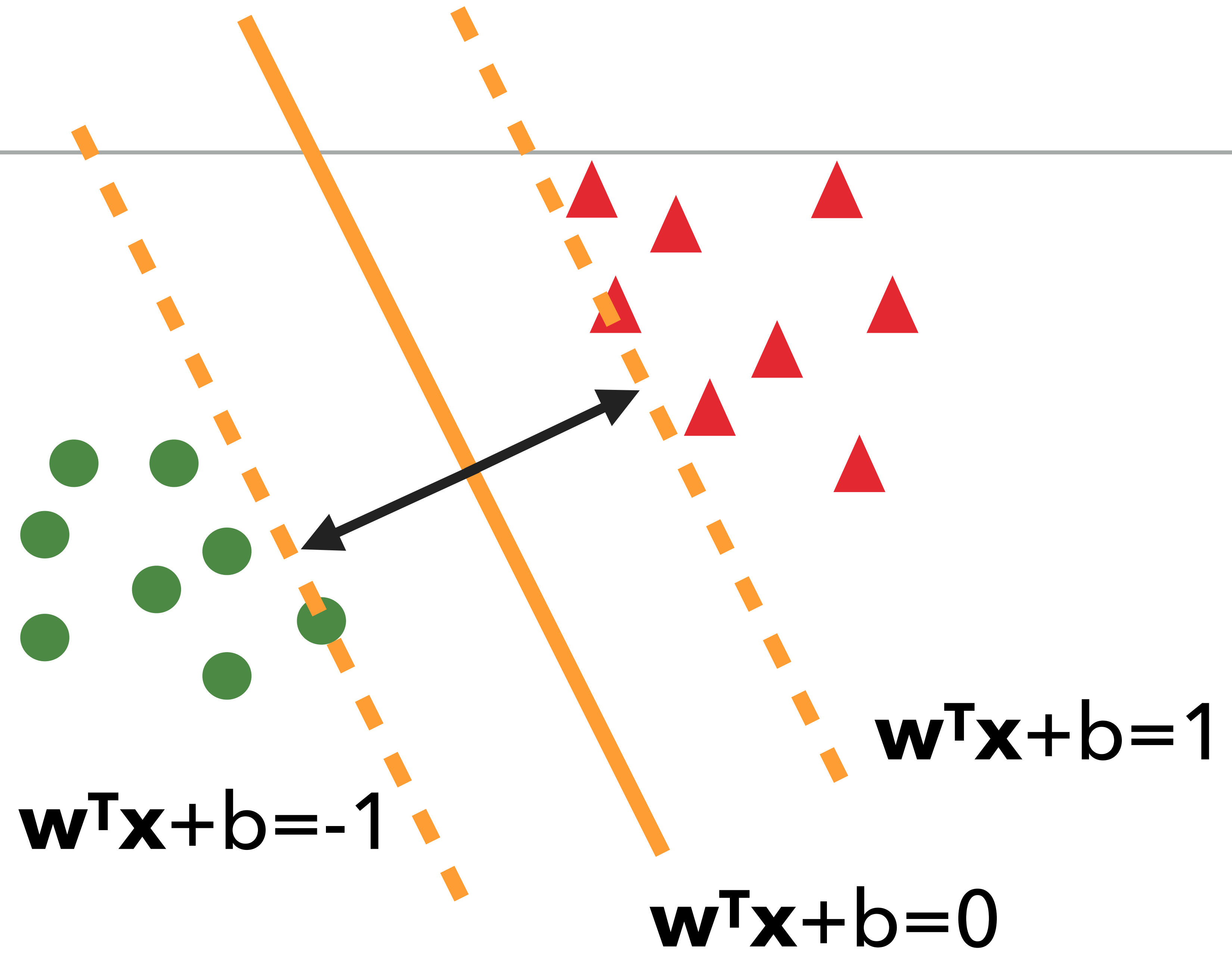
Normalize the scale for simplicity: **w**=**w**/c, b=b/c

HOW LARGE IS THE MARGIN?

Margin = $d_+ + d_- = 2/||\mathbf{w}||$



SVM LEARNING SCORING FUNCTION



- Maximize margin, i.e., max $2/||\mathbf{w}||$
- Subject to constraints!
  - Margin is defined by the closet positive/negative examples to the boundary
  - Constraint 1: $\mathbf{w}^T\mathbf{x} + b \geq 1, \forall y_i = +1$
  - Constraint 2: $\mathbf{w}^T\mathbf{x} + b \leq -1, \forall y_i = -1$
  - Combine constraints 1 and 2: $y_i(\mathbf{w}^T\mathbf{x} + b) \geq 1, \forall i \in \{1,2,\ldots,N\}$
- Search: solve this optimization problem…more in the next class!

OPTIMIZATION

OPTIMIZATION IN MODEL LEARNING

- Consider a **space** of possible models $M=\{M_1, M_2, \ldots, M_k\}$ with parameters θ
- Search over model structures or parameters, e.g.:
  - **Parameters**: In a logistic regression model, what are regression coefficients (**w**) that maximize log likelihood on the training data?
  - **Model structure**: In a decision trees, what is the tree structure that minimizes 0/1 loss on the training data?
- Find the best model structure or parameter values that **optimize** scoring function value on the training dataset

COMBINATORIAL OPTIMIZATION VS. SMOOTH OPTIMIZATION

- **Combinatorial** optimization:
  - The model space is a finite or countably infinite set (i.e., the scoring function is discrete)
  - Systematically search through the model space, often using heuristics
  - Example: Search the best decision tree structure
- **Smooth** optimization:
  - The model space is an uncountable set (i.e., the scoring function is continuous)
  - Gradient-based optimization
  - Example: Find parameter values for Naive Bayes Classifier

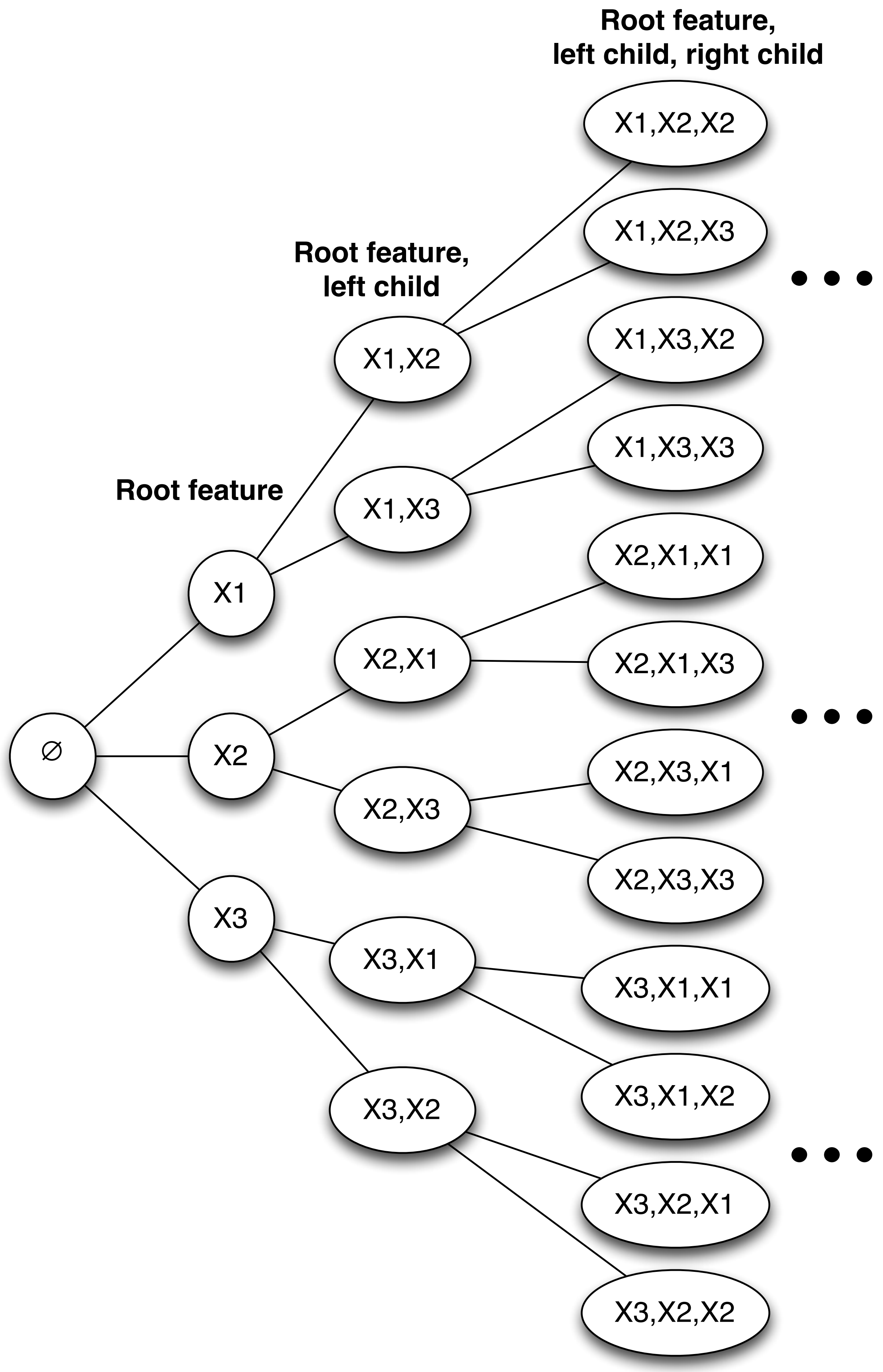
COMBINATORIAL OPTIMIZATION

COMBINATORIAL OPTIMIZATION: STATE SPACE

- *S*: state; the set of all possible models
- *Action*(*s*): the set of all possible actions that can be performed at state *s*
- *Result*(*s*, *a*): the result of performing action *a* on state *s*, which is another state
- *Score*(*s*): the scoring function value of state *s* (i.e., for the model represented by state *s*)
- State space: representing each state as a node, and two nodes *s* and *s′* are connected by an edge if *s′*=Result(*s*, *a*) for some *a* in Action(*s*)

STATE SPACE EXAMPLE

- Constructing the state space of decision trees where each data point has three binary variables X_1, X_2, X_3



SEARCH THROUGH THE STATE SPACE

- Start from a particular state (i.e., model)
- Evaluate the score of the current state
- If the current state is not the goal state (e.g., model with maximum score), expand the current state by applying all possible actions to the current state and generate successor states
- Pick one of the successor state, repeat, and backtrack
- Exhaustive search: systematic search through all possible states in the state space
  - e.g., depth-first search, breadth-first search, etc.

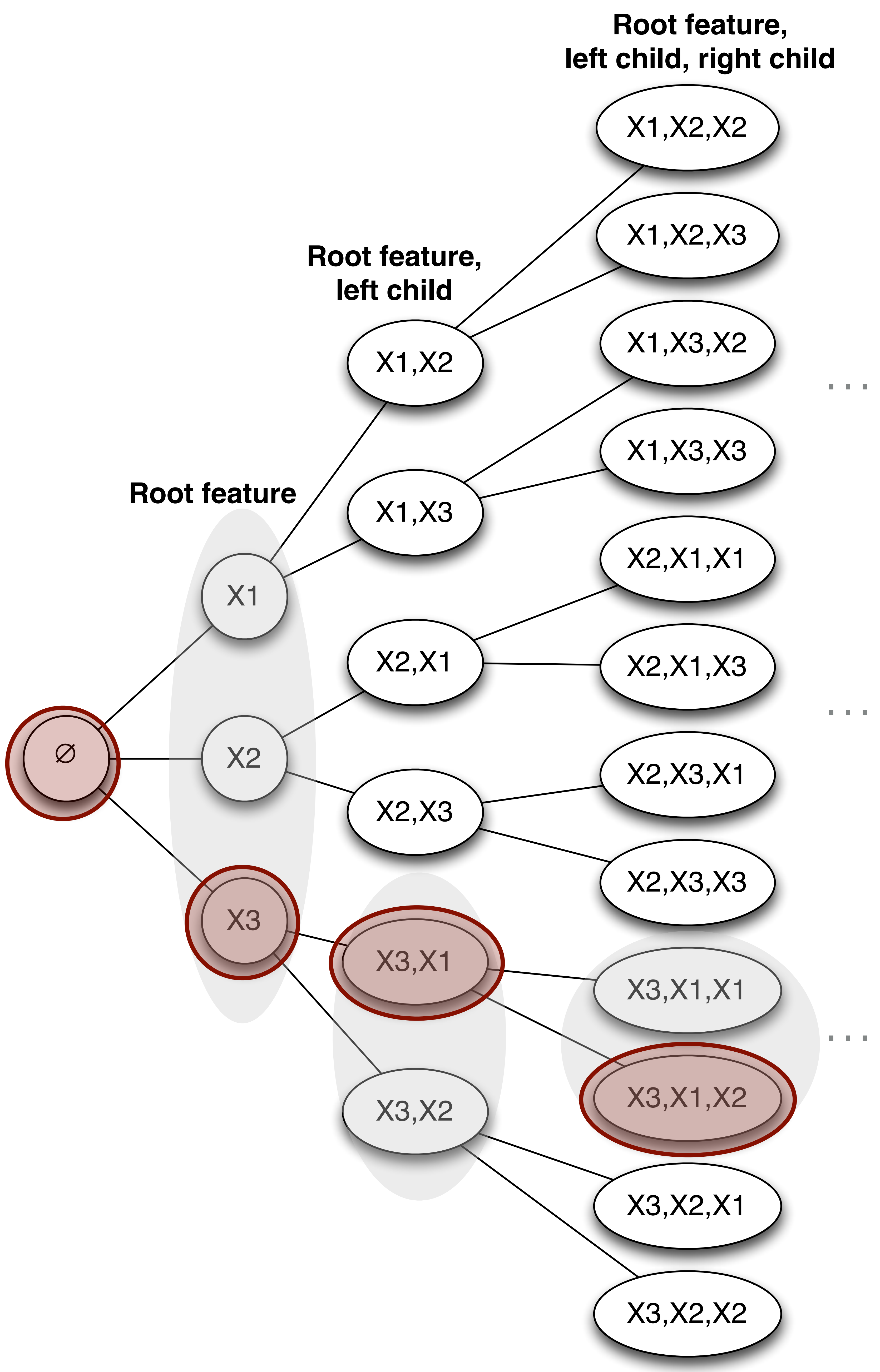
HEURISTIC SEARCH

- Typically, there is an exponential number of models in the model space, making it intractable to exhaustively search the space
  - Thus, it is generally impossible to return a model that is **guaranteed** to have the best score
- Instead, we have to resort to **heuristic** search techniques
  - Methods are evaluated experimentally and shown to have good performance on average
  - **Greedy** search: Given a current model M, look for the successor of M and move to the best of these (if any have a score better than M)

GREEDY SEARCH

- Choose an initial state M^0 corresponding to a particular model structure (e.g., an empty tree)
- Let M^i be the model considered at the i-th iteration
- For each iteration i
  - Construct all possible models $\{M^{j1}, \ldots, M^{jk}\}$ adjacent to M^i (as defined by action operators)
  - Evaluate scores for all models $\{M^{j1}, \ldots, M^{jk}\}$
  - Choose to move to the adjacent model with best score: $M^{i+1} = M^{j.best}$
  - Repeat until there is no possible further improvement in the score

Which states does greedy search consider?



SMOOTH OPTIMIZATION

SMOOTH OPTIMIZATION

- **Smooth** scoring functions:
  - If a function is *smooth*, it is differentiable and the derivatives are continuous, then we can use gradient-based optimization
    - If function is *convex*, we can often solve the minimization problem using **convex optimization** or **gradient descent**
    - If function is smooth but non-linear, we can use iterative search over the surface of S to find a local minimum (e.g., hill-climbing)

CONVEX OPTIMIZATION PROBLEMS

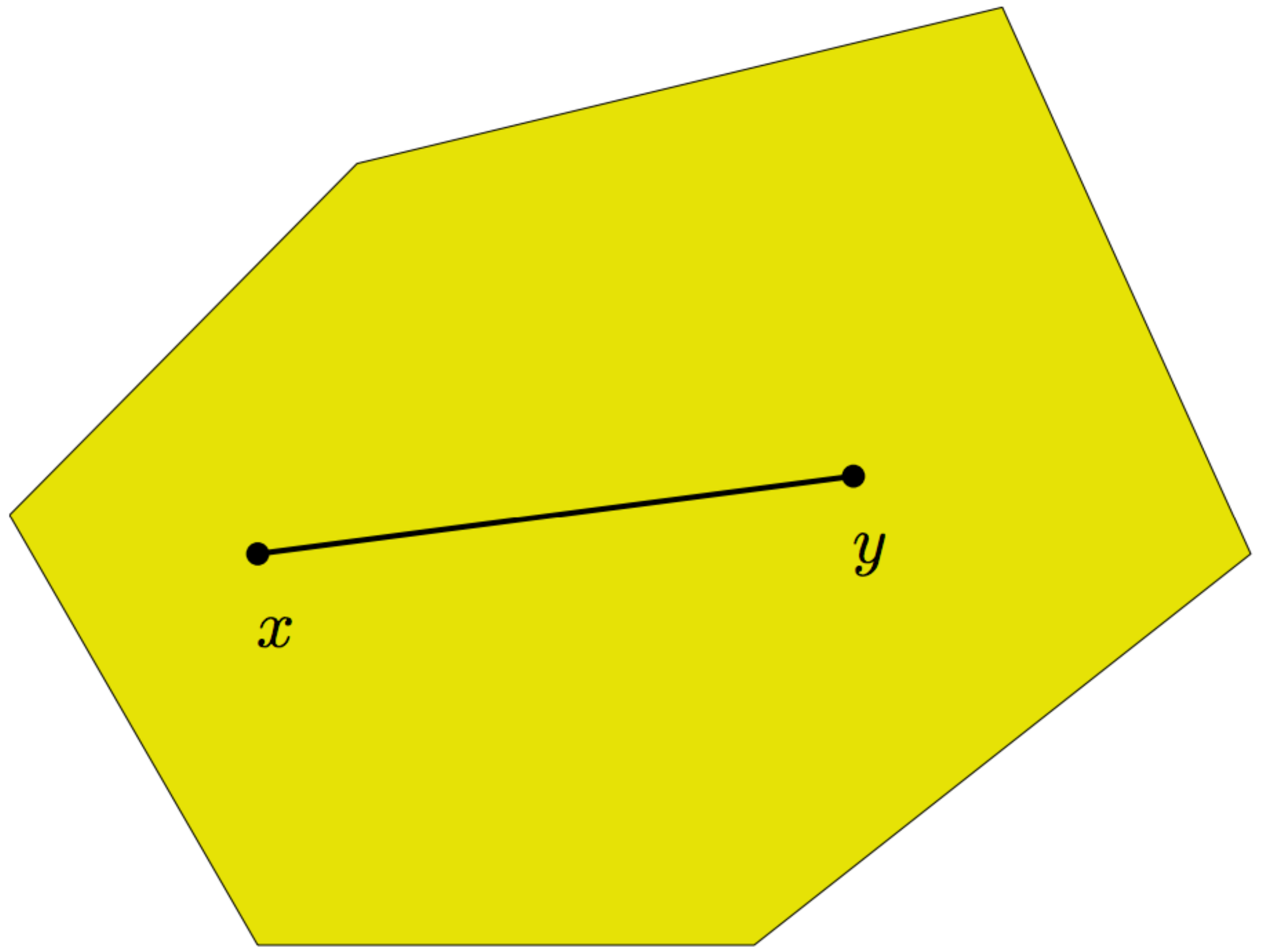
$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in C \end{array}$$

- x is the optimization variable (*e.g., model parameters*)
  f (*e.g., score function*) is a **convex function**
  C is a **convex set** (*e.g., constraints on model parameters*)
- For convex optimization problems, all locally optimal points are globally optimal

CONVEX SET

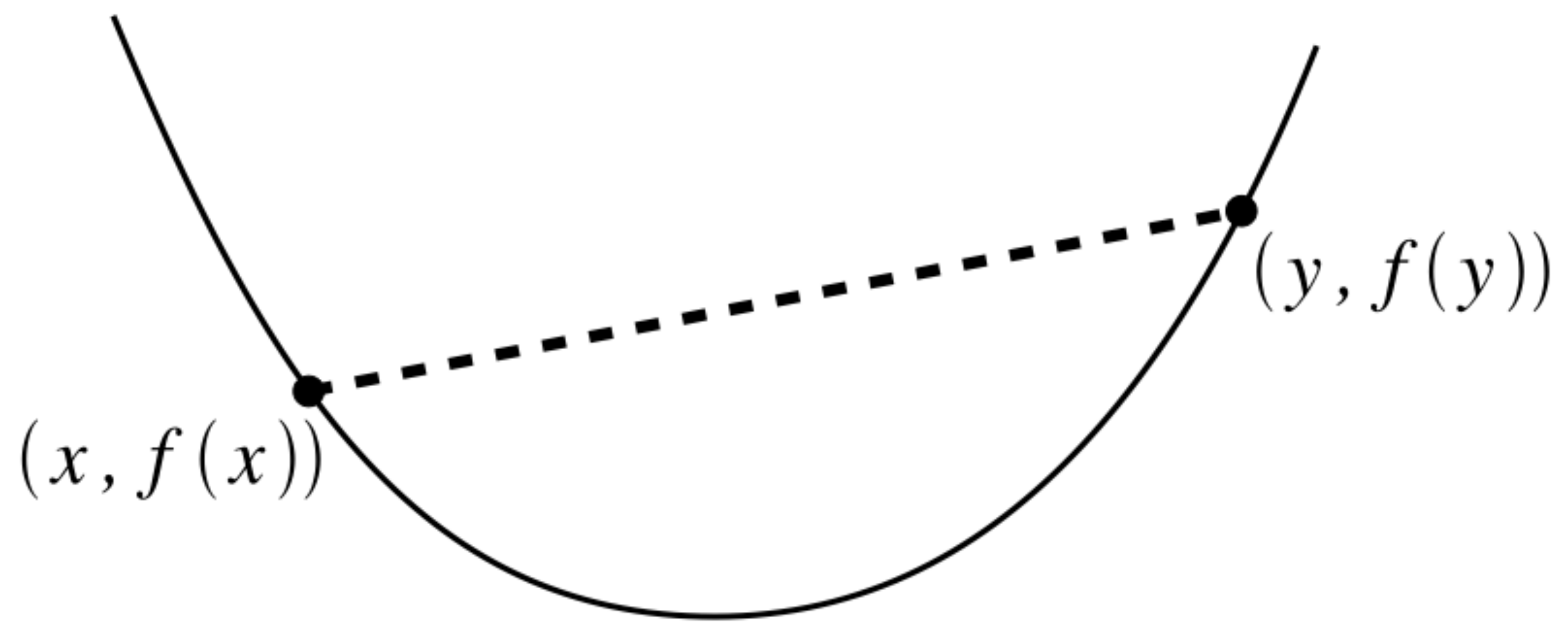
- A set C is convex if for any $x, y \in C$ and any θ with $0 \leq \theta \leq 1$ we have

$$\theta x + (1 - \theta) y \in C$$



CONVEX FUNCTIONS

- In graph of convex function *f*, the line connecting two points must lie above the function: $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$ for all $0 \leq \alpha \leq 1$
- Practical test for convexity: a twice differentiable function *f* of a variable *x* is convex on an interval if an only if for any *x* in the interval: $f''(x) \geq 0$
  - Strictly convex if $f''(x) > 0$



- Sum of convex functions is convex; max of convex functions is convex