CS 453X: Class 12

Jacob Whitehill

Quadratic programming

Quadratic programming

- Quadratic programming is *not* a kind of computer programming.

- **Quadratic programming (QP)** problems are a kind of mathematical optimization problem:

  - Quadratic objective function (which we want to minimize or maximize).

  - Linear equality and/or inequality constraints.

- Same vein as linear programming, dynamic programming.

Quadratic programming

- Nonetheless, quadratic programs are typically *solved* using computer programs.

- As part of homework 4, you will use an off-the-shelf Python-based quadratic programming solver (**`cvxopt`**).

cvxopt

`cvxopt.solvers.` `qp` (*P*, *q* [, *G*, *h* [, *A*, *b* [, *solver* [, *initvals*]]]])

Solves the pair of primal and dual convex quadratic programs

$$\begin{array}{ll} \text{minimize} & (1/2)x^T Px + q^T x \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array}$$

http://cvxopt.org/userguide/coneprog.html#quadratic-programming

cvxopt

Quadratic objective

Linear objective

`cvxopt.solvers.qp(P, q [, G, h [, A, b [, solver [, initvals ] ] ] ])`

Solves the pair of primal and dual convex quadratic programs

$$\begin{array}{ll} \text{minimize} & (1/2)x^T Px + q^T x \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array}$$

Inequality constraints

Equality constraints

- To train an SVM using a QP, we need to define the appropriate matrices from our training data.

- The **x** comprises both the **w** (hyperplane) and *b* (bias) (similar to how we implemented bias in linear regression).

http://cvxopt.org/userguide/coneprog.html#quadratic-programming

cvxopt

Quadratic objective

Linear objective

`cvxopt.solvers.qp(P, q [, G, h [, A, b [, solver [, initvals ] ] ] ])`

Solves the pair of primal and dual convex quadratic programs

$$\begin{array}{ll} \text{minimize} & (1/2)x^T Px + q^T x \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array}$$

Inequality constraints

Equality constraints

- **q** will just be 0 (we have no linear objective function).
- **P**: part of homework 4.

http://cvxopt.org/userguide/coneprog.html#quadratic-programming

cvxopt

Quadratic objective

Linear objective

`cvxopt.solvers.qp(P, q [, G, h [, A, b [, solver [, initvals ] ] ] ])`

Solves the pair of primal and dual convex quadratic programs

$$\begin{array}{ll} \text{minimize} & (1/2)x^T Px + q^T x \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array}$$

Inequality constraints

Equality constraints

- **G** and **h** need to encode the linear inequality constraints.
- We will not use **A** or **b** (optional parameters) since we have no equality constraints.

http://cvxopt.org/userguide/coneprog.html#quadratic-programming

Defining *G* and *h*

- Suppose you have just two optimization variables — x_1 and x_2 — as well as the following constraints:
  - $2x_1 - 3x_2 \leq 2$
  - $x_1 + x_2 \geq 0$

- We need to express these both as linear inequality constraints (≤ 0).

Defining *G* and *h*

- Suppose you have just two optimization variables — x_1 and x_2 — as well as the following constraints:
  - $2x_1 - 3x_2 \leq 2$
  - $x_1 + x_2 \geq 0 \quad \Rightarrow \quad -x_1 - x_2 \leq 0$

- We need to express these both as linear inequality constraints (≤ 0).

Defining *G* and *h*

- Suppose you have just two optimization variables — x_1 and x_2 — as well as the following constraints:
  - $2x_1 - 3x_2 \leq 2$
  - $x_1 + x_2 \geq 0 \quad \Rightarrow \quad -x_1 - x_2 \leq 0$

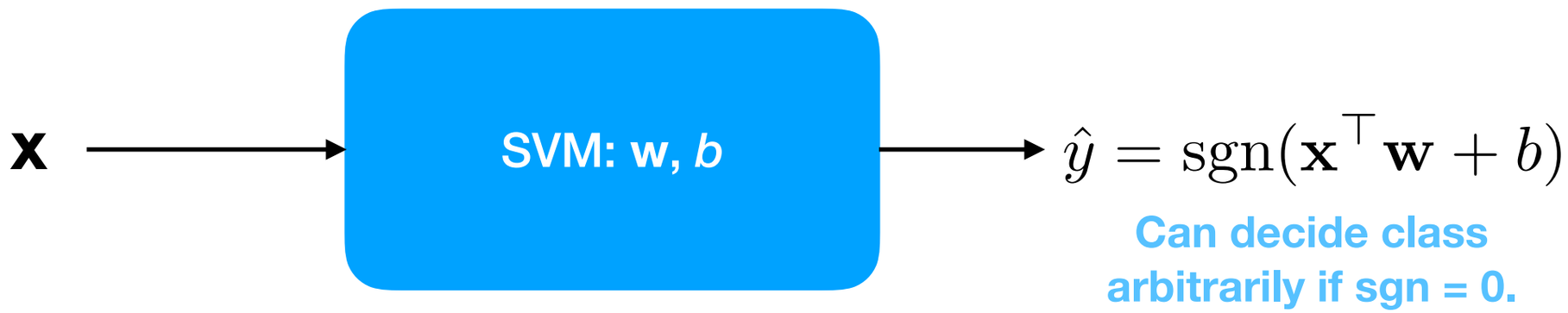
- We need to express these both as linear inequality constraints (≤ 0).

$$\underbrace{\begin{bmatrix} 2 & -3 \\ -1 & -1 \end{bmatrix}}_{\mathbf{G}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\mathbf{x}} \leq \underbrace{\begin{bmatrix} 2 \\ 0 \end{bmatrix}}_{\mathbf{h}}$$

SVM: classification

SVM: classification

- Here's how an SVM classifies a new example:

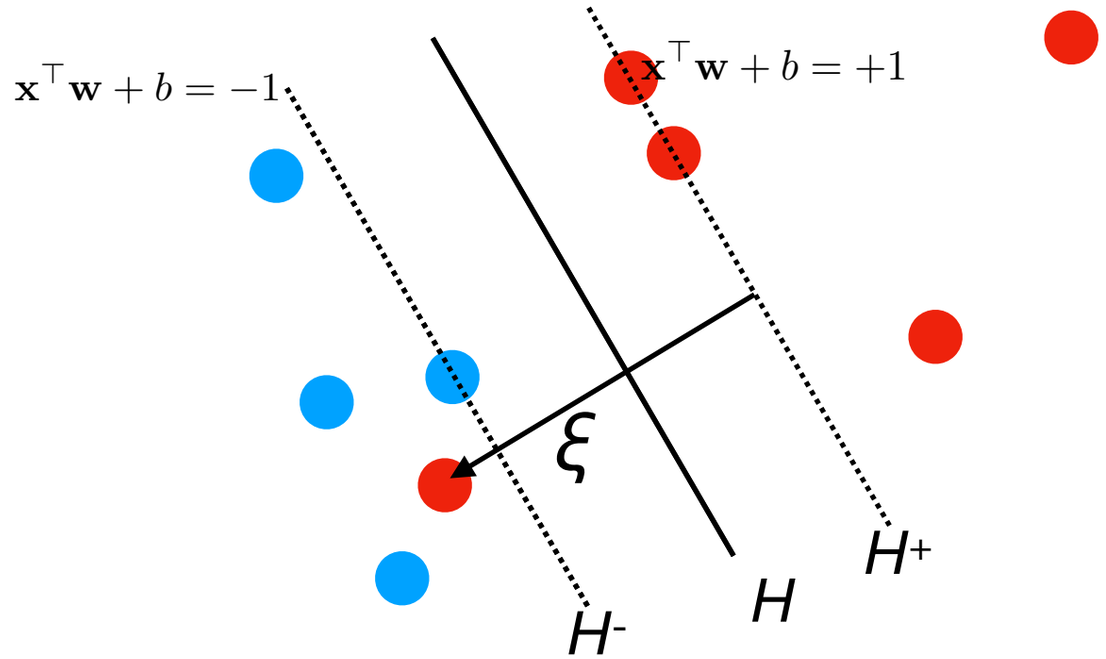


Soft-margin SVMs

Soft vs hard SVM margin

- The SVM defined so far is a **hard margin** SVM:
    - The hyperplane must perfectly separate all the + from the - examples.
- In many settings, this is unrealistic because the data are **linearly inseparable** — no separating hyperplane exists.
- To support such datasets, a **soft margin** SVM has also been formulated that allows for small “infractions” of the constraints.

Soft vs hard SVM margin



- We can “soften” the SVM constraint by allowing for **slack** in the position of each data point *i* w.r.t. the hyperplane *H*.

Soft margin SVM

- With a soft-margin SVM, we loosen the constraint on each data point $\mathbf{x}^{(i)}$ by giving it a **slack variable** $\xi^{(i)}$.

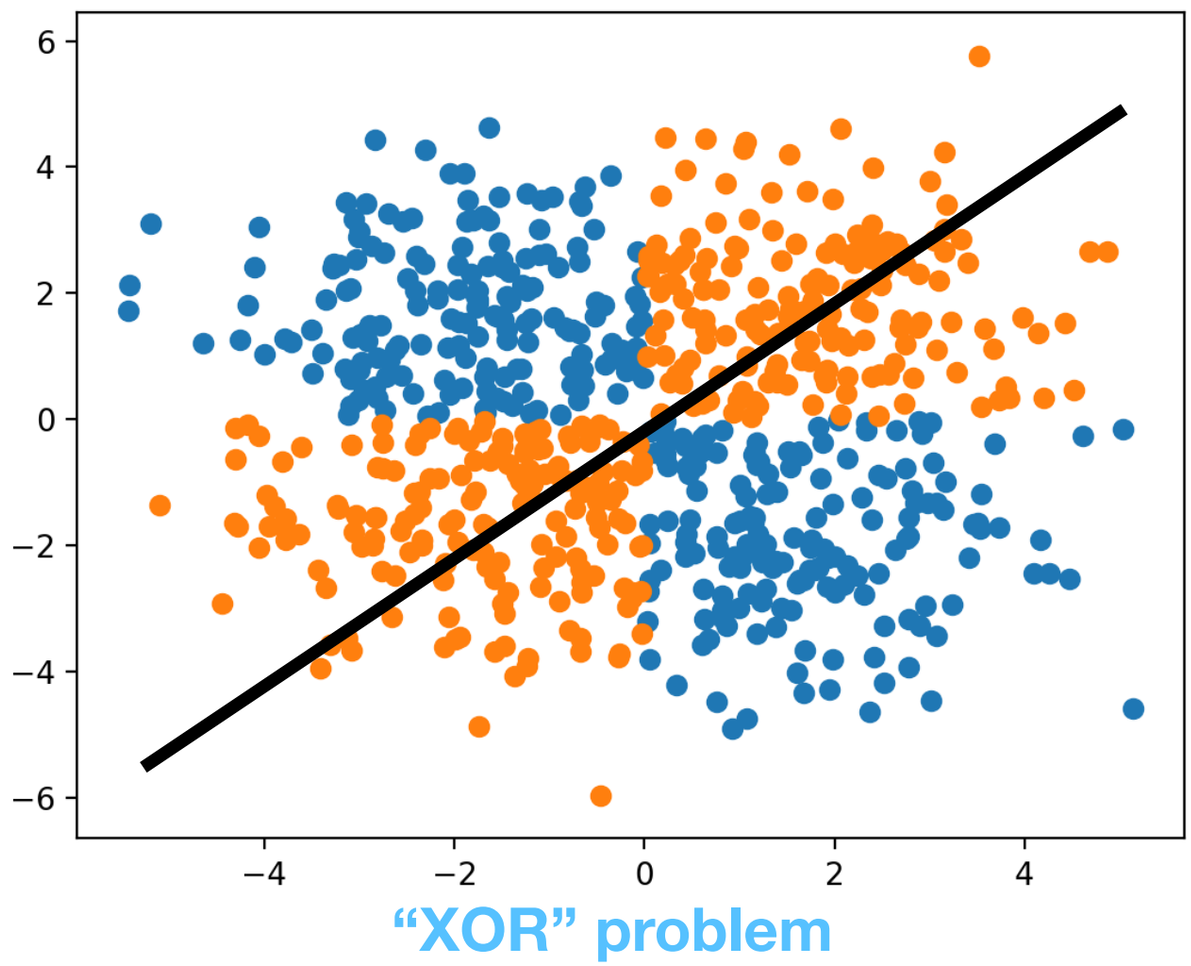
- We penalize large slack variables using a penalty parameter *C*.

- The new optimization problem becomes:
  - Minimize: $\frac{1}{2}\mathbf{w}^\top \mathbf{w} + C\sum_{i=1}^{n} \xi^{(i)}$ (Error penalty)
  - Subject to: $y^{(i)}\left(\mathbf{x}^{(i)\top}\mathbf{w} + b\right) \geq 1 - \xi^{(i)}$ (Slack)

Feature transformations

Linearly inseparable data

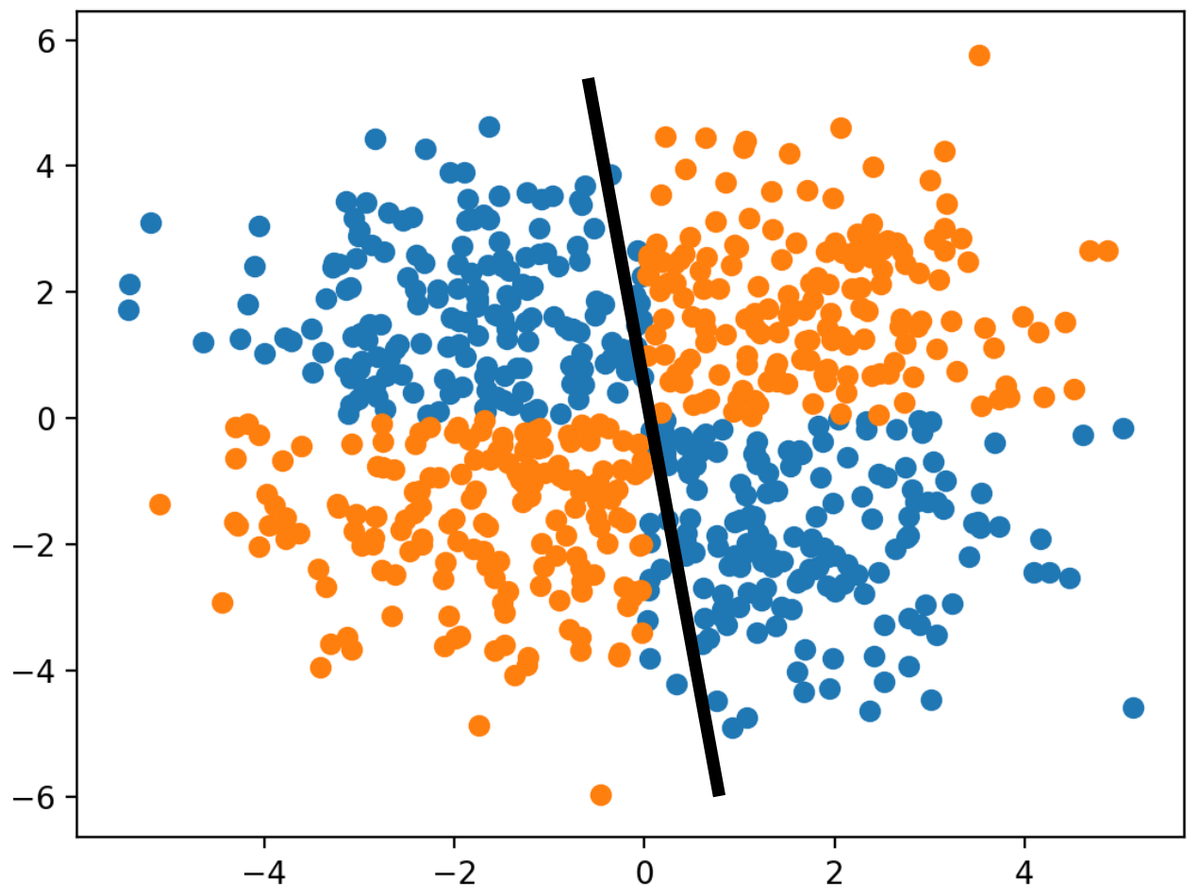
- SVMs use a hyperplane to separate data in two classes.
- But what if the data are **linearly inseparable**, e.g.:
- No matter what **w**, *b* we choose, the SVM will never do a good job of classifying the data.



“XOR” problem

Linearly inseparable data

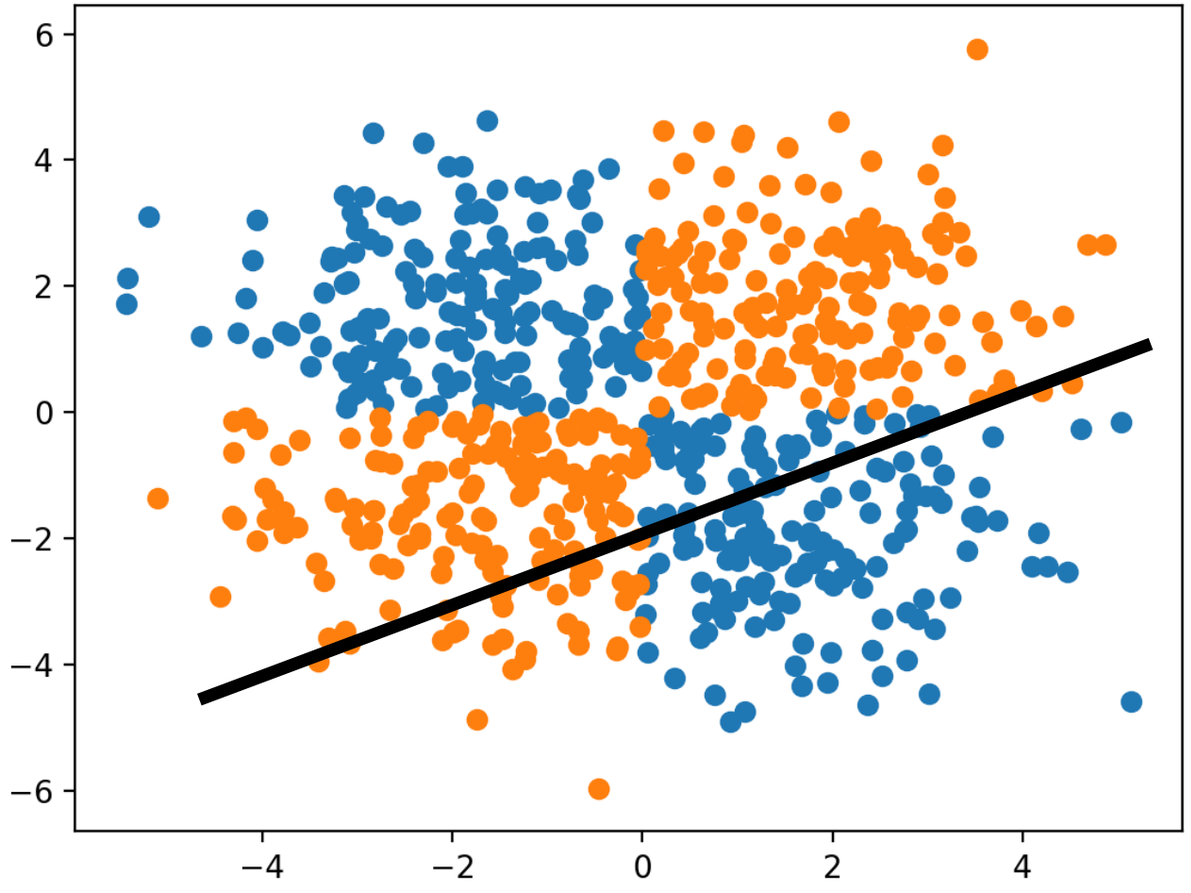
- SVMs use a hyperplane to separate data in two classes.
- But what if the data are **linearly inseparable**, e.g.:
- No matter what **w**, *b* we choose, the SVM will never do a good job of classifying the data.



“XOR” problem

Linearly inseparable data

- SVMs use a hyperplane to separate data in two classes.
- But what if the data are **linearly inseparable**, e.g.:
- No matter what **w**, *b* we choose, the SVM will never do a good job of classifying the data.

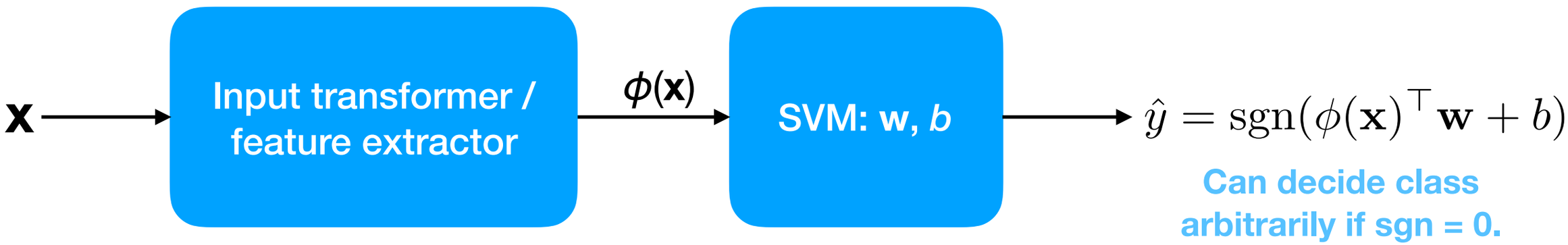


“XOR” problem

Feature transformations

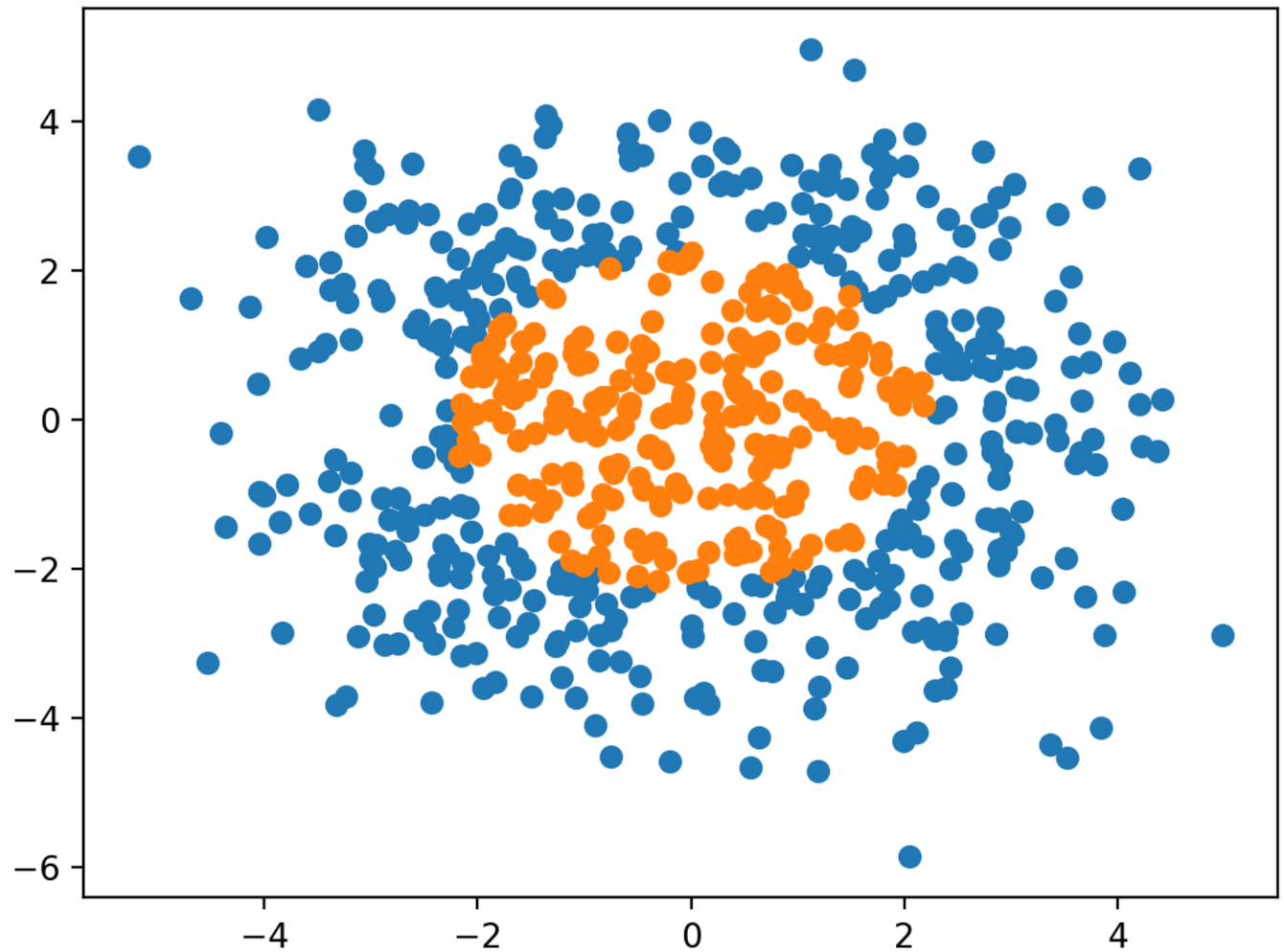
- But what if we somehow transformed the raw input **x** into some (possibly higher-dimensional) representation $\phi(\mathbf{x})$?

- Might the classes become linearly separable then?



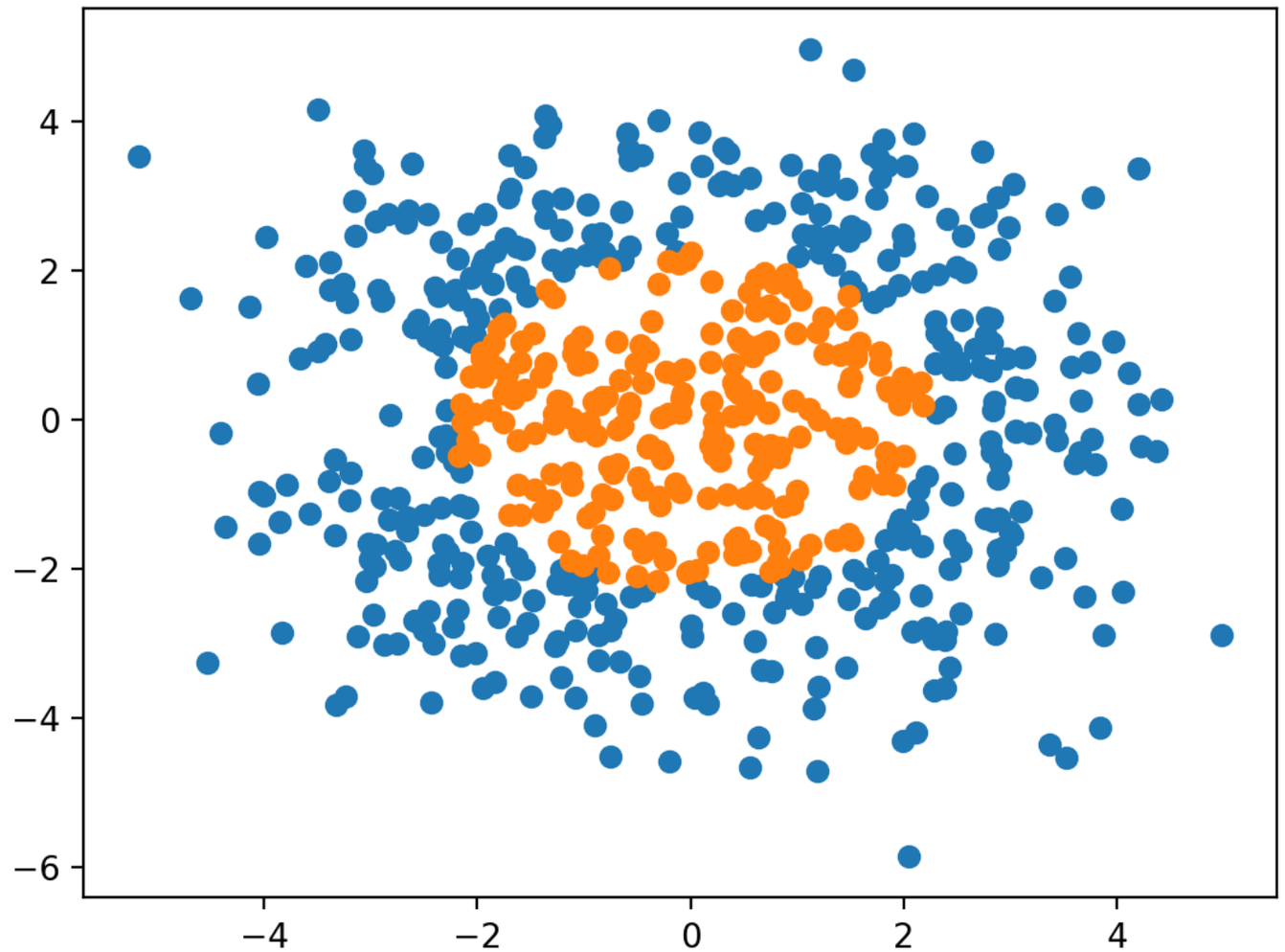
Example

- The data shown below are not linearly separable.
- What is the essential difference between the classes?



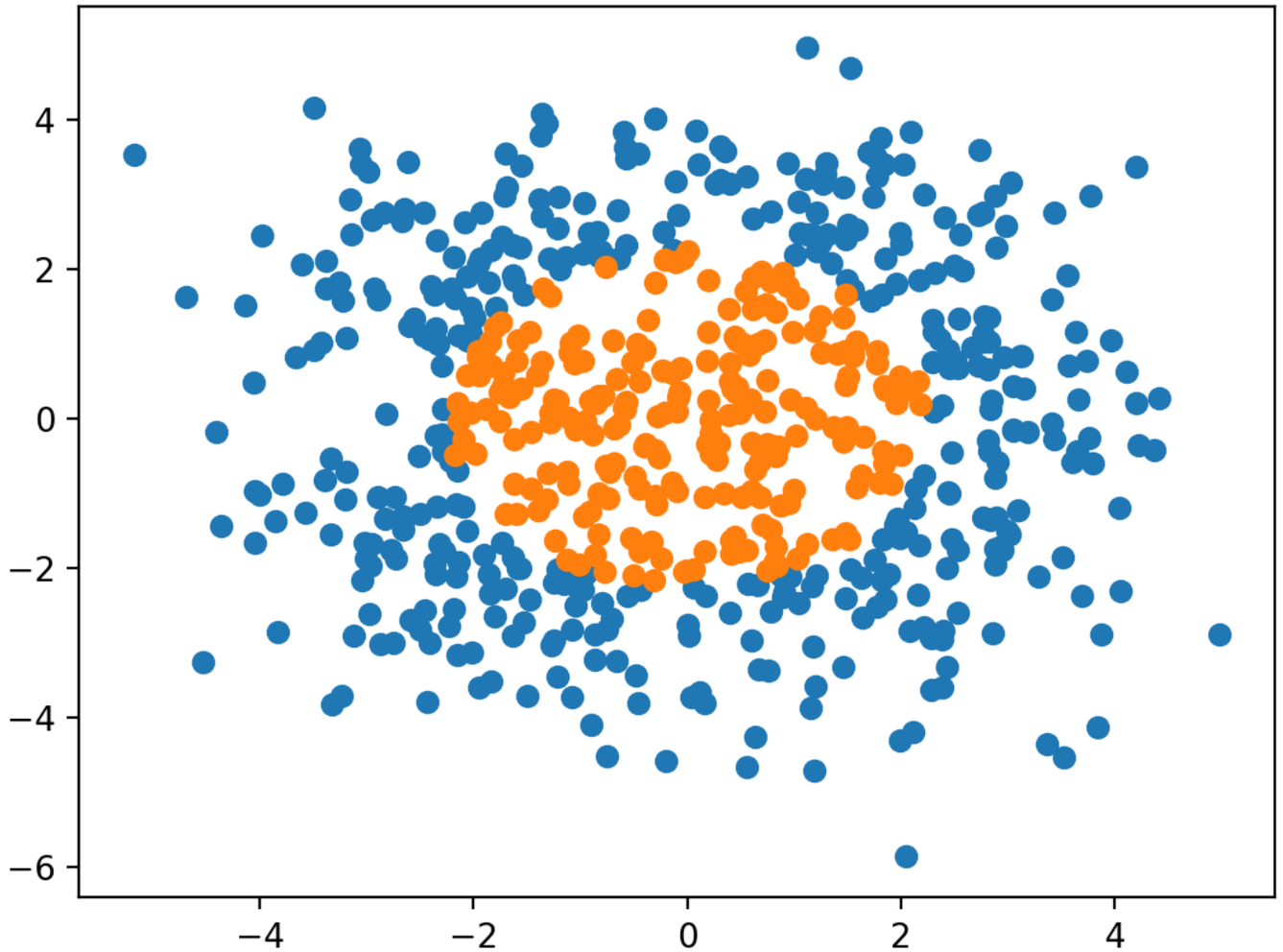
Example

- The data shown below are not linearly separable.
- What is the essential difference between the classes?
- The blue points are farther from the origin than the orange points.
- How could we measure distance?



Example

- The data shown below are not linearly separable.
- What is the essential difference between the classes?
- The blue points are farther from the origin than the orange points.
- How could we measure distance?
  - $x^2 + y^2$

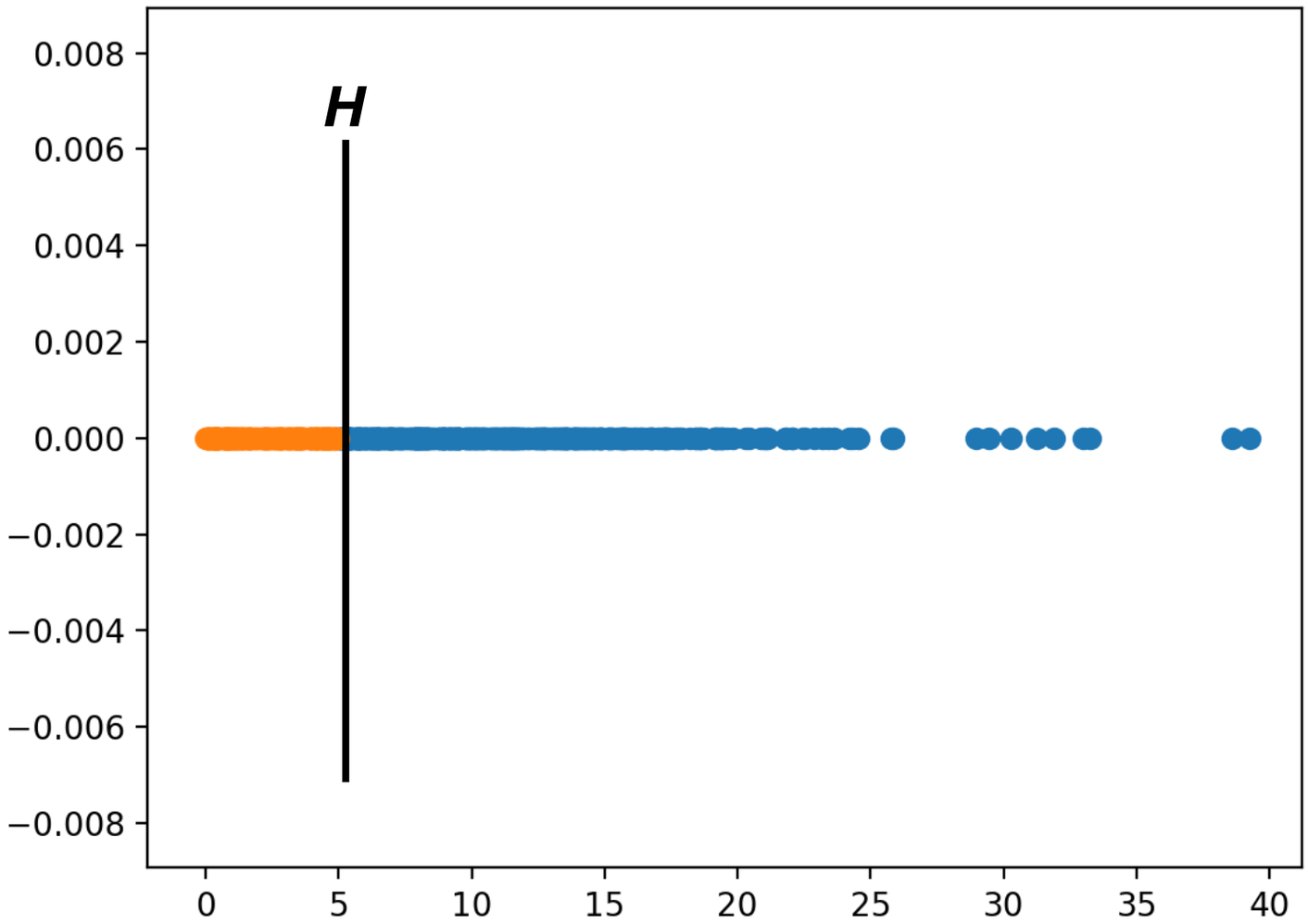


Example

- We can render these two classes linearly separable by first transforming each point (*x*,*y*) into:

$$\phi(x, y) = \begin{bmatrix} x^2 + y^2 \\ 0 \end{bmatrix}$$

The *x* coordinate will already reveal the class label; hence, the *y*-coordinate doesn't matter.



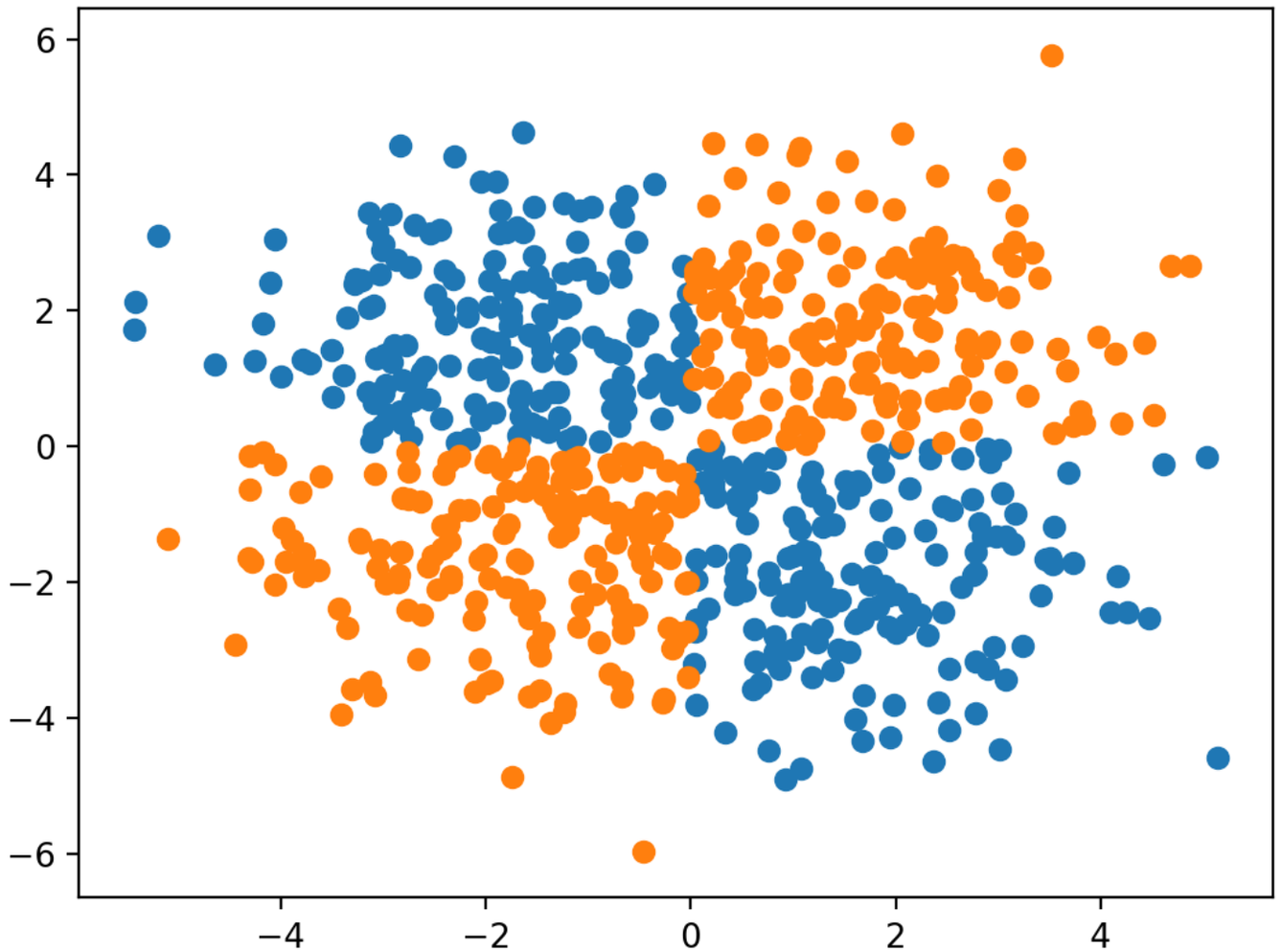
Exercise 1

- Which of the following transformation(s) will make these data linearly separable?

1. $\phi(x, y) = \begin{bmatrix} x^2 \\ y^2 \end{bmatrix}$

2. $\phi(x, y) = \begin{bmatrix} x \\ x^2 + y^2 \end{bmatrix}$

3. $\phi(x, y) = \begin{bmatrix} x \\ xy \end{bmatrix}$

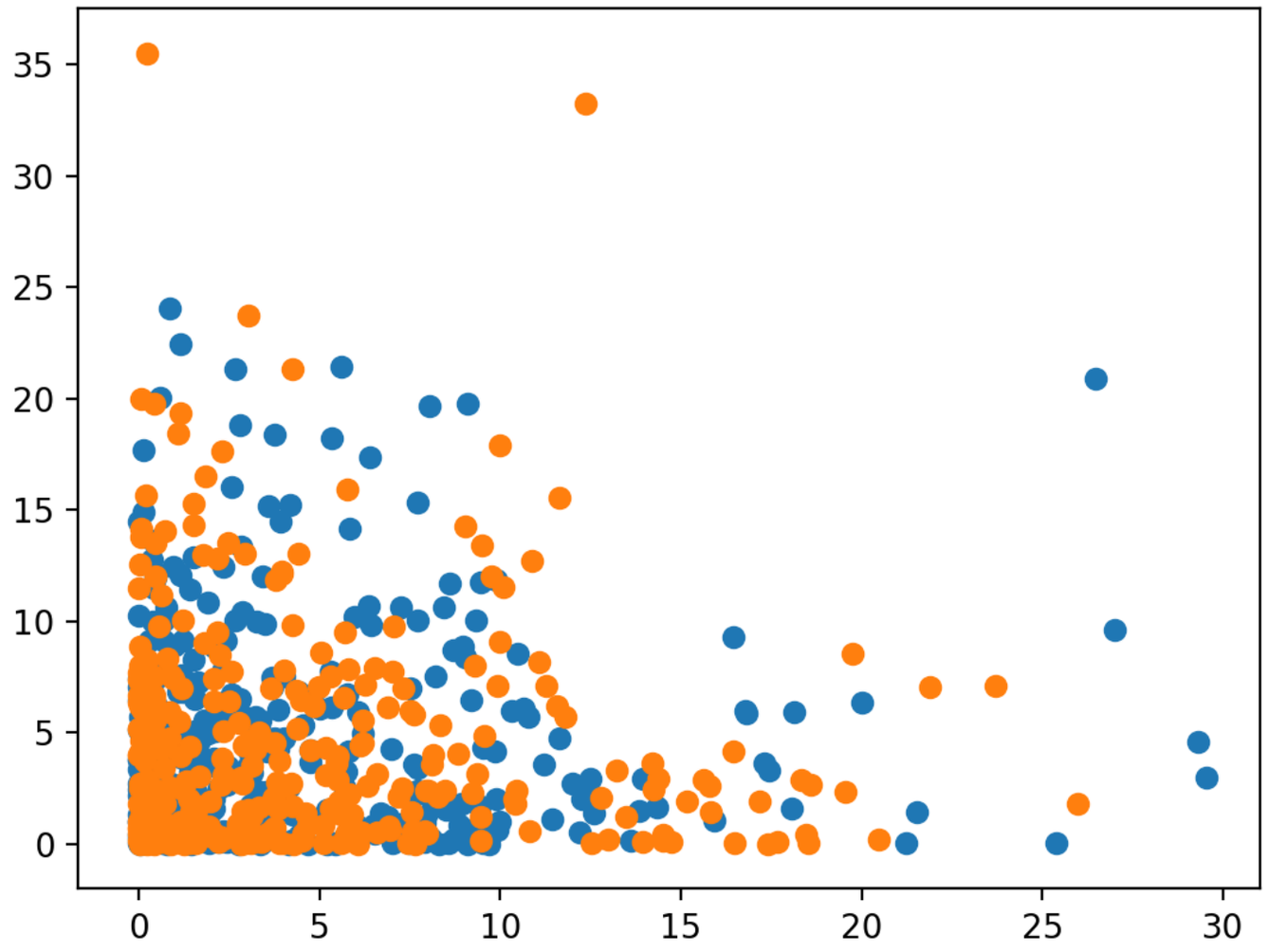


Exercise 1

- Which of the following transformation(s) will make these data linearly separable?

1. $\phi(x, y) = \begin{bmatrix} x^2 \\ y^2 \end{bmatrix}$

This collapses across both the left-right and up-down half-spaces, but does not render the two classes linearly separable.

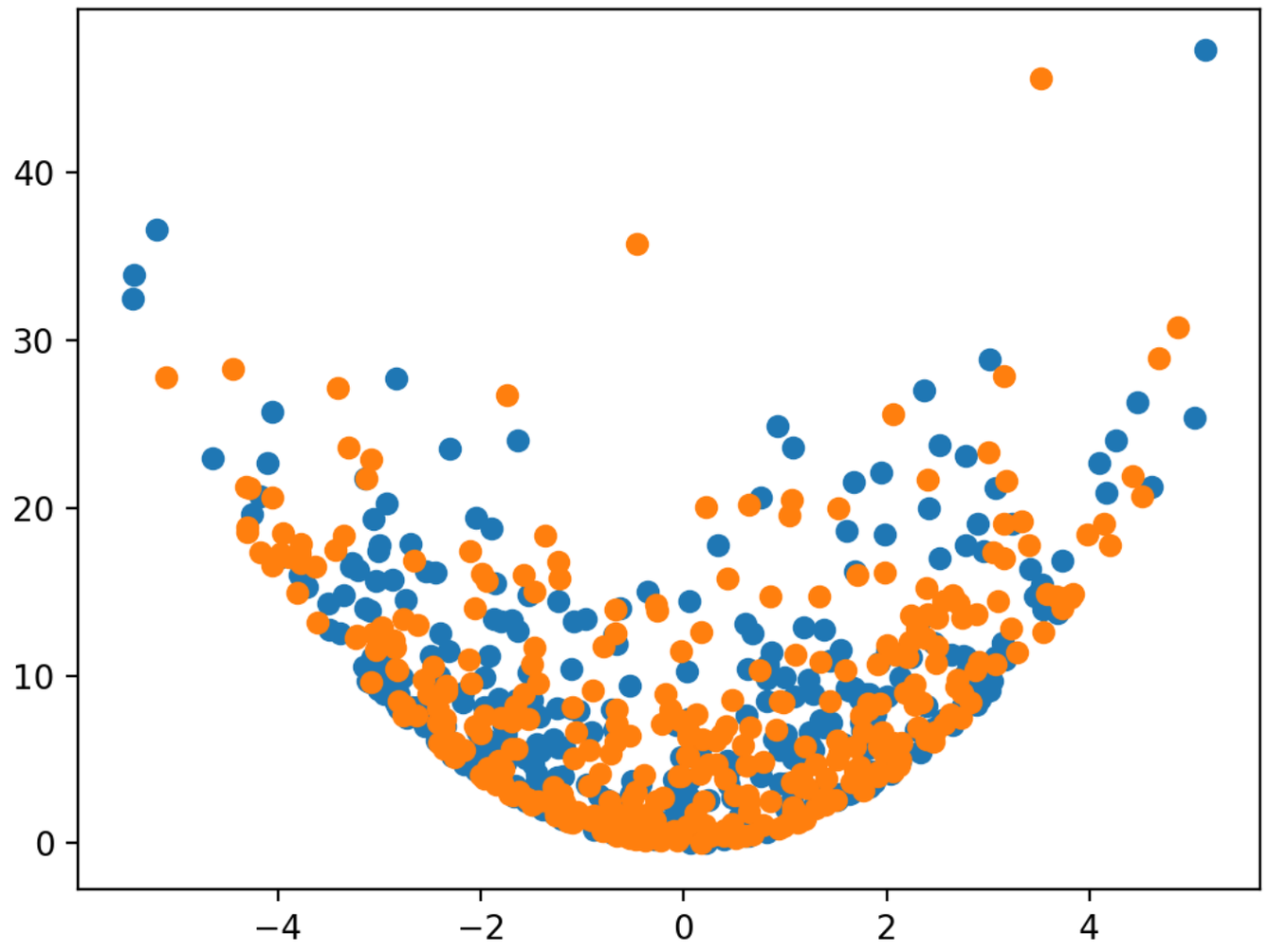


Exercise 1

- Which of the following transformation(s) will make these data linearly separable?

2. $\phi(x, y) = \begin{bmatrix} x \\ x^2 + y^2 \end{bmatrix}$

$x^2 + y^2$ computes the distance from the origin, which is not related to the class label in this problem.

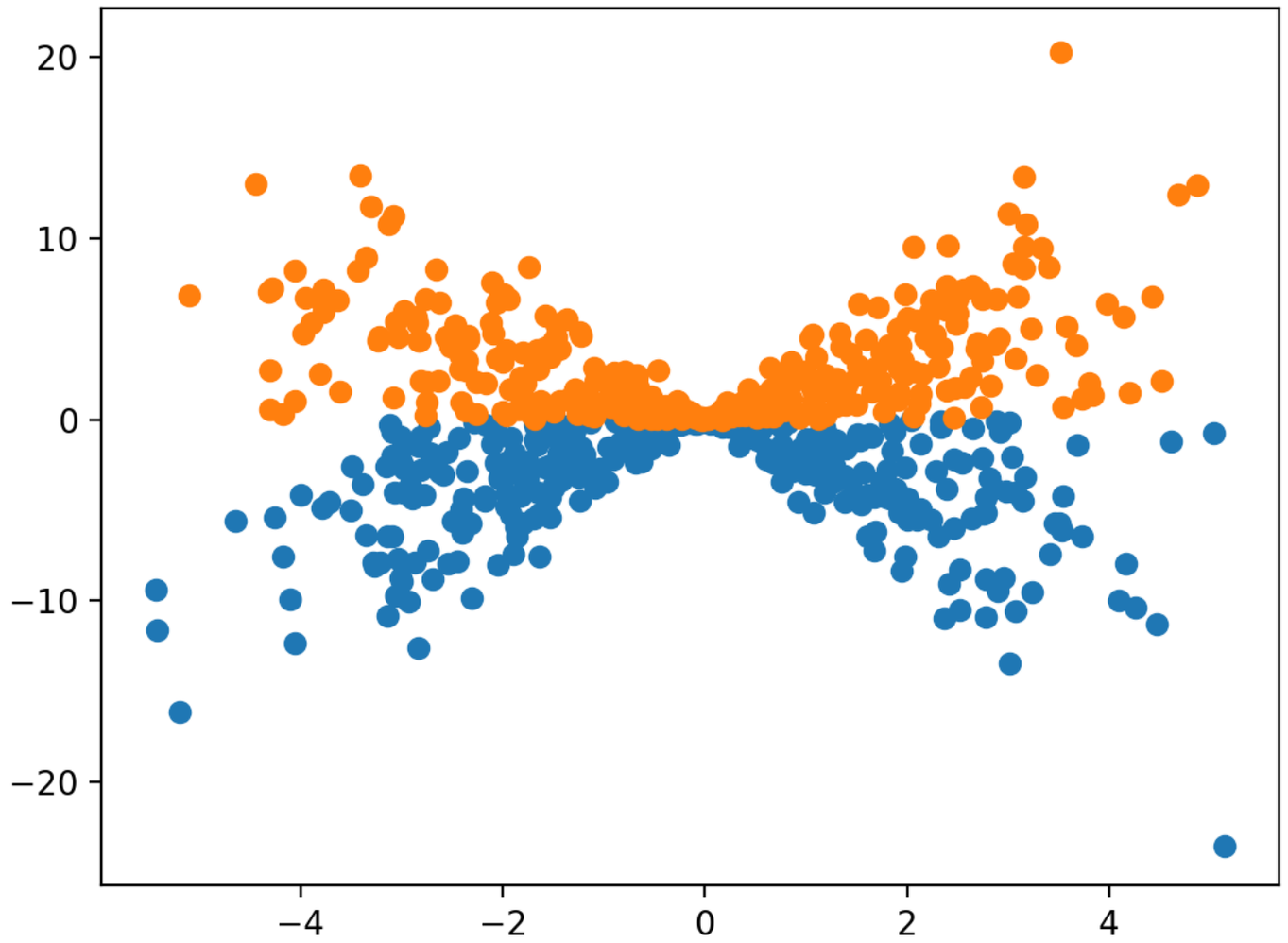


Exercise 1

- Which of the following transformation(s) will make these data linearly separable?

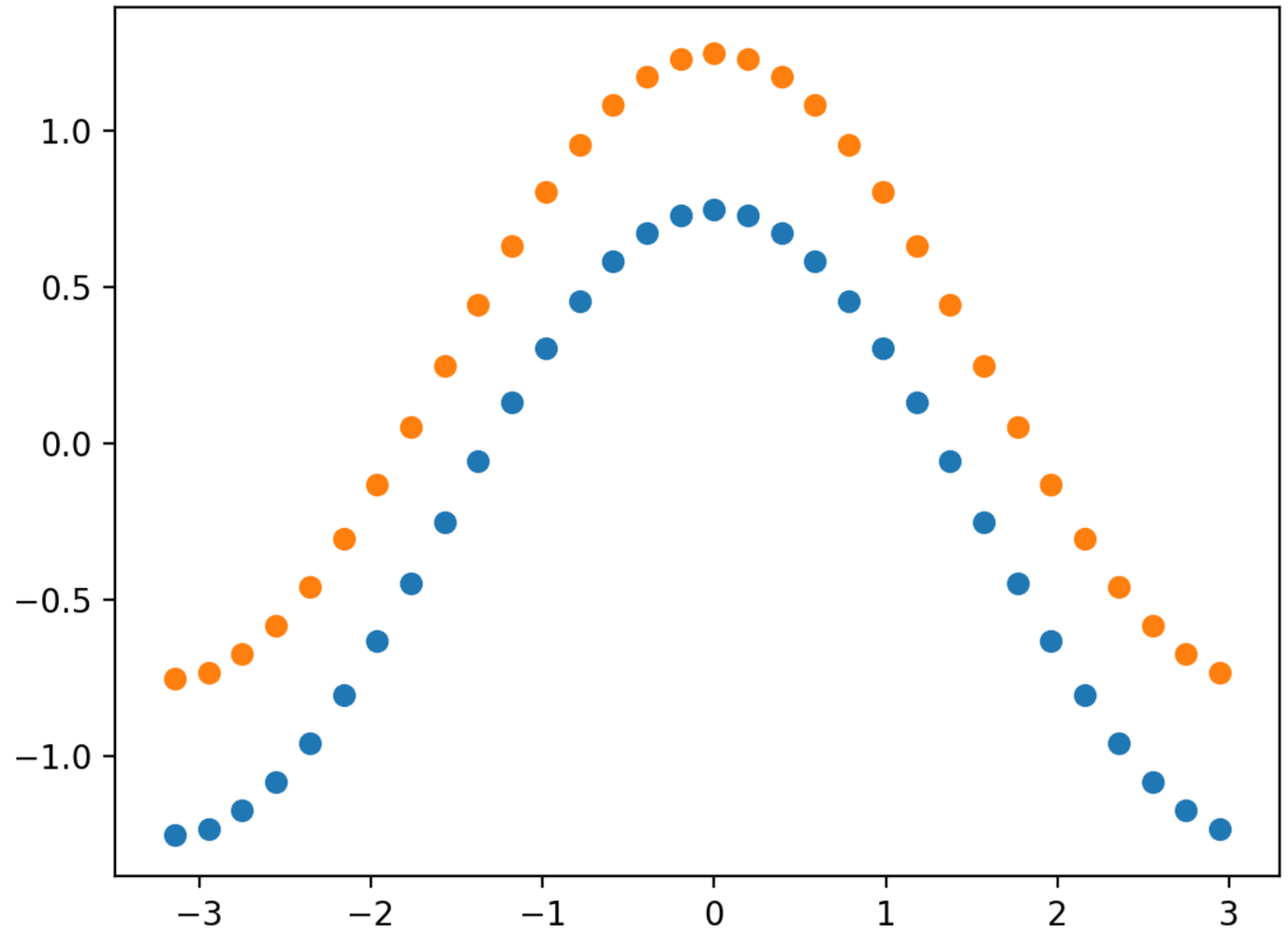
xy actually determines the class label in this problem; hence, this transformation makes the classes separable.

3. $\phi(x, y) = \begin{bmatrix} x \\ xy \end{bmatrix}$



Exercise 2

- Which of the following transformation(s) will make these data linearly separable?



1. $\phi(x, y) = \begin{bmatrix} x + y \\ x - y \end{bmatrix}$

2. $\phi(x, y) = \begin{bmatrix} \cos(x) \\ y \end{bmatrix}$

3. $\phi(x, y) = \begin{bmatrix} |x| \\ y \end{bmatrix}$

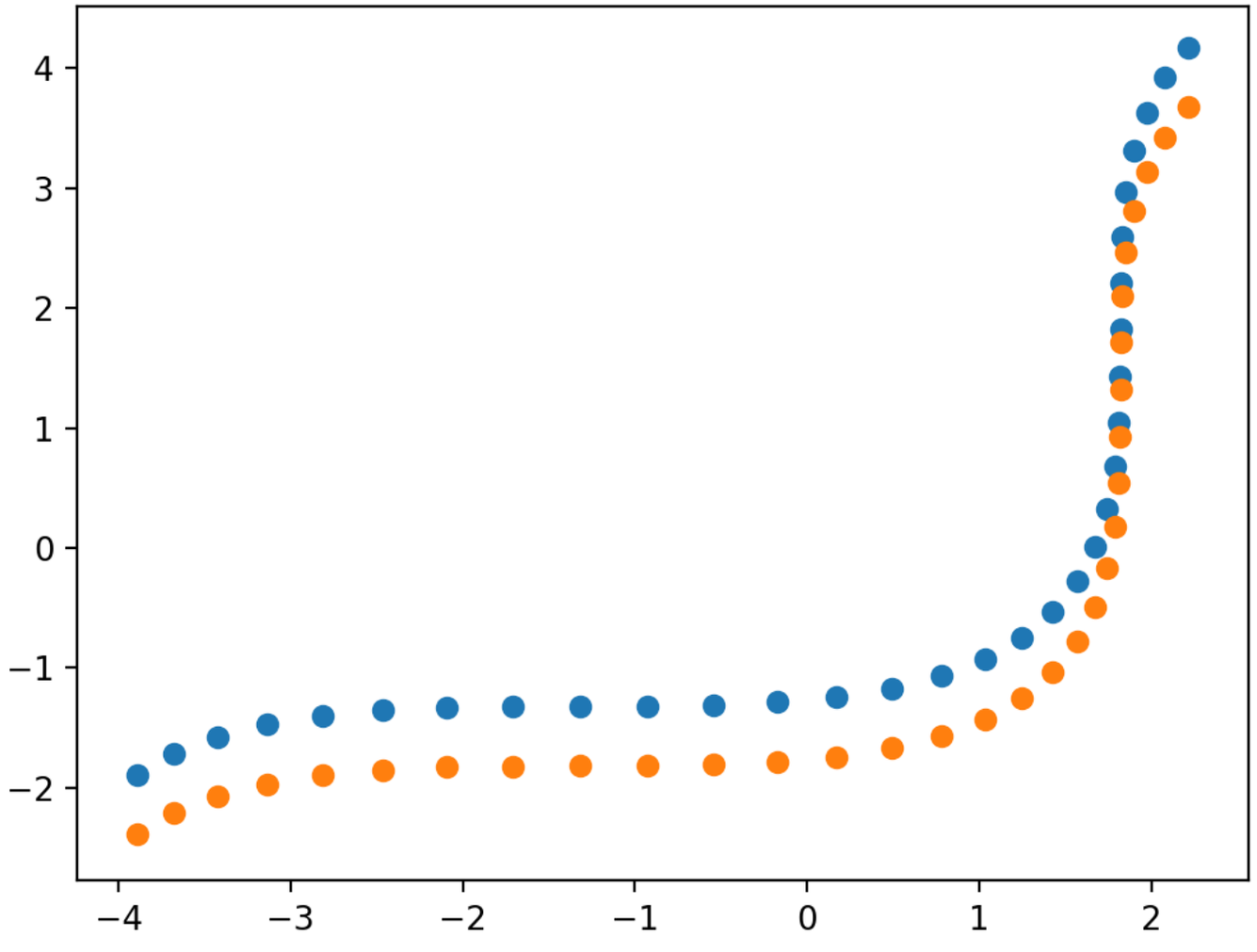
Exercise 2

- Which of the following transformation(s) will make these data linearly separable?

1. $\phi(x, y) = \begin{bmatrix} x + y \\ x - y \end{bmatrix}$

This transformation is linear because it can be written:

$$\phi(x, y) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



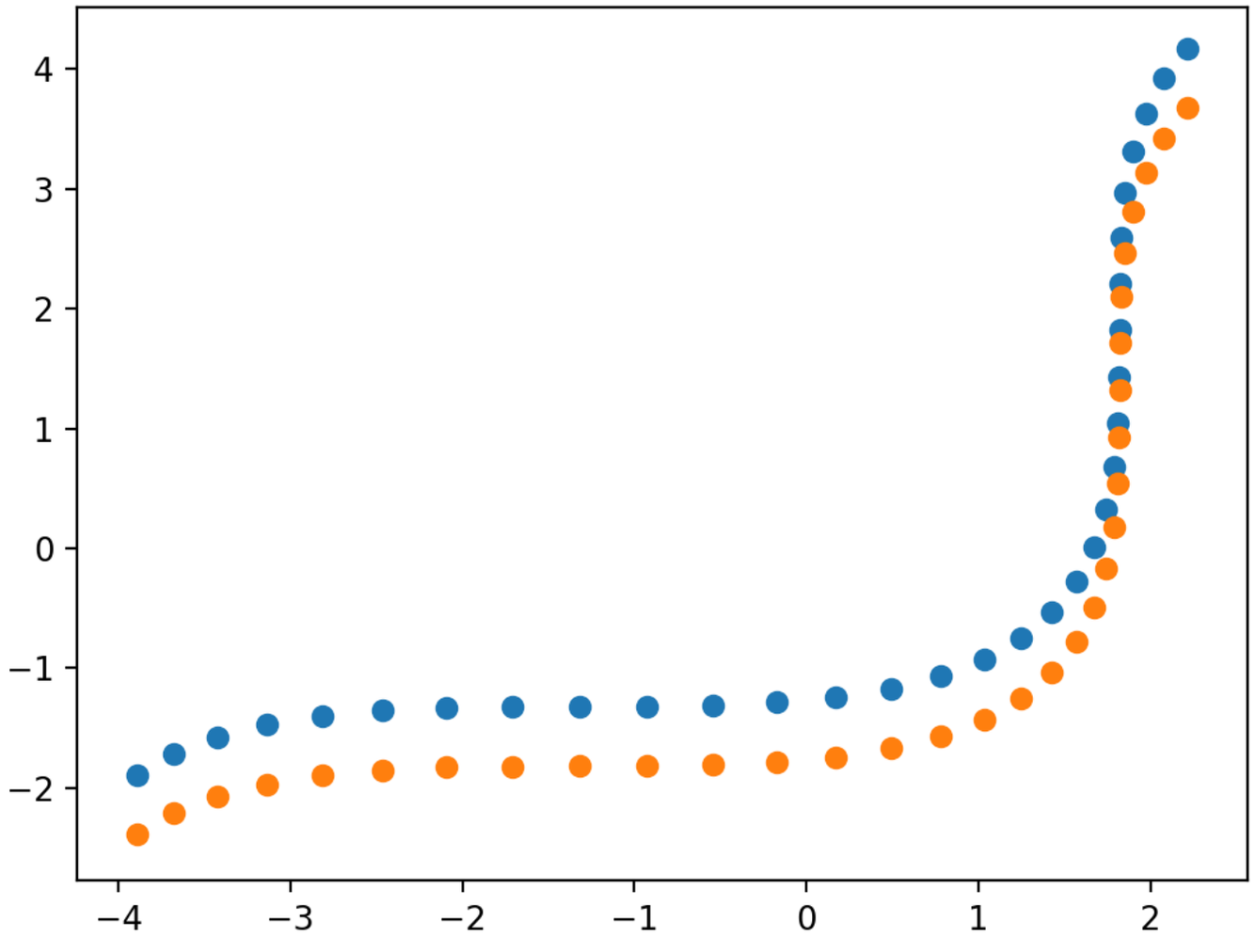
Exercise 2

- Which of the following transformation(s) will make these data linearly separable?

1. $\phi(x, y) = \begin{bmatrix} x + y \\ x - y \end{bmatrix}$

Linear transformations can only rotate, scale, and shear the input data.

No linear transformation can make linearly inseparable data linearly separable.

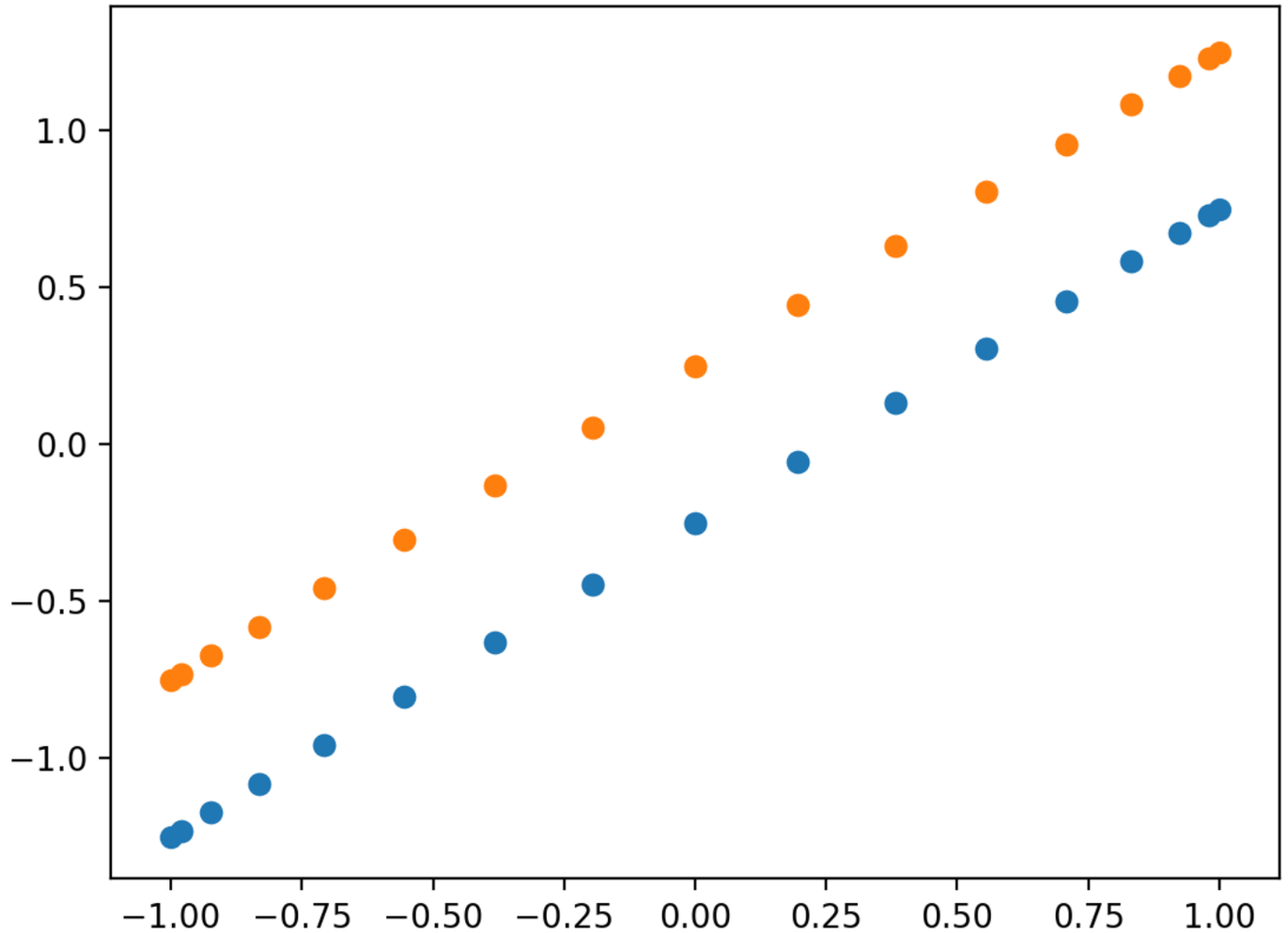


Exercise 2

- Which of the following transformation(s) will make these data linearly separable?

Since *y* is already ~cos(*x*), transforming *x* to cos(*x*) puts the two axes on the same scale.

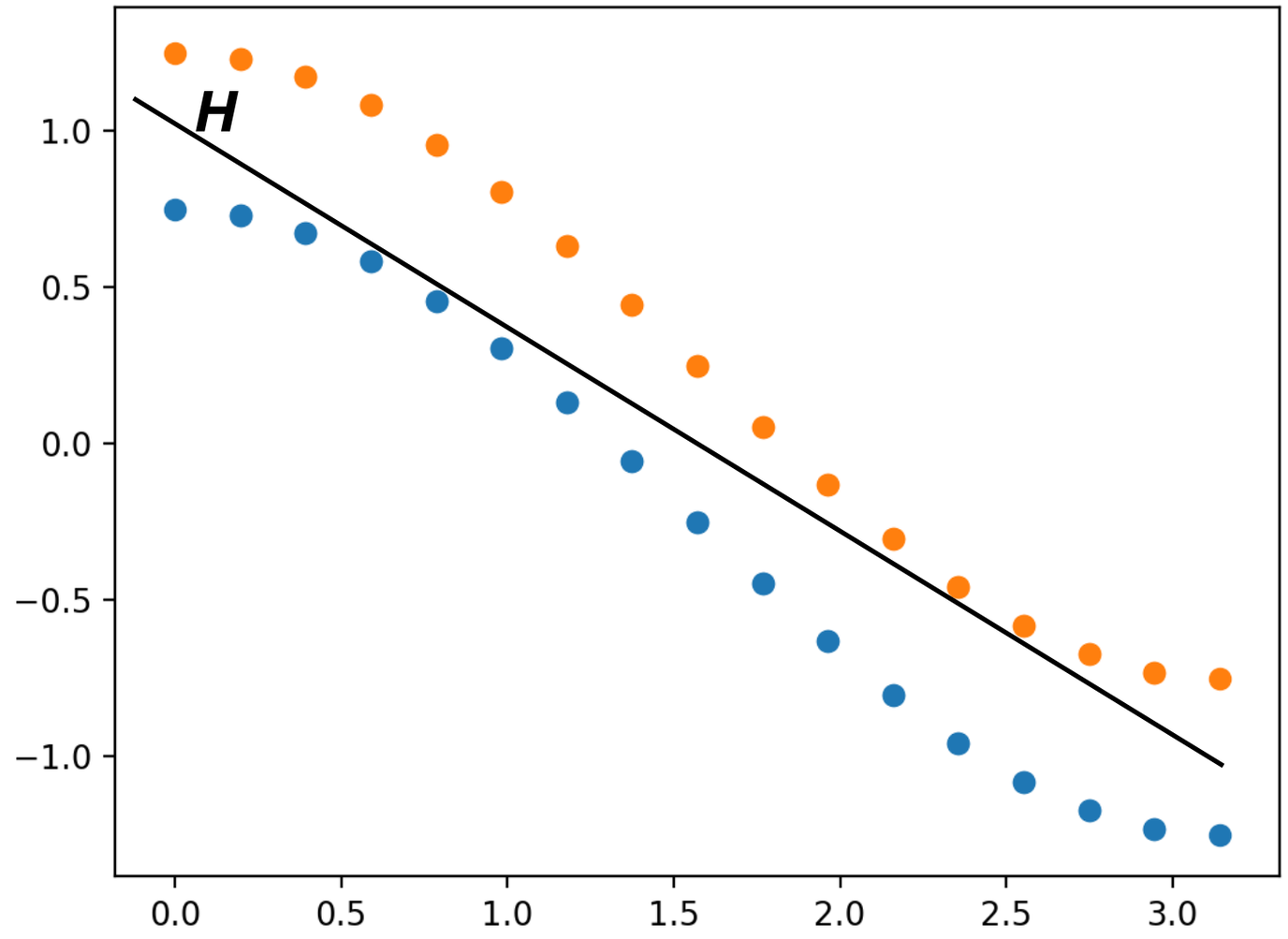
2. $\phi(x, y) = \begin{bmatrix} \cos(x) \\ y \end{bmatrix}$



Exercise 2

- Which of the following transformation(s) will make these data linearly separable?

Because of the gap between the two curves, collapsing across the two left and right half-spaces (via $|x|$) renders the classes linearly separable.



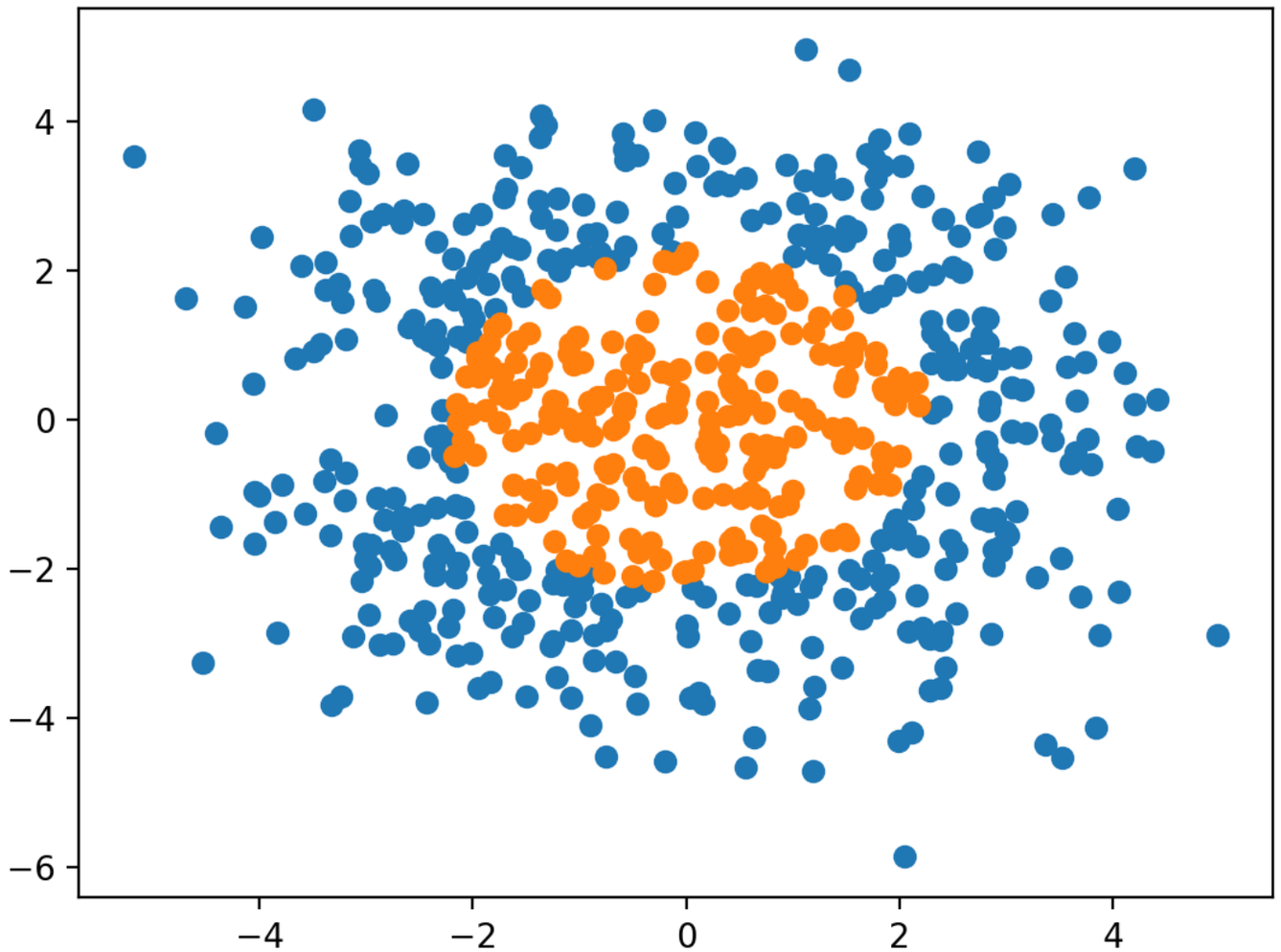
3. $\phi(x, y) = \begin{bmatrix} |x| \\ y \end{bmatrix}$

Example

- Let’s re-visit this set of data…

- Feature transformations are usually applied to map the input data into a *higher* dimensional space.

- With higher dimensions, there is a greater opportunity for the classes to separate.



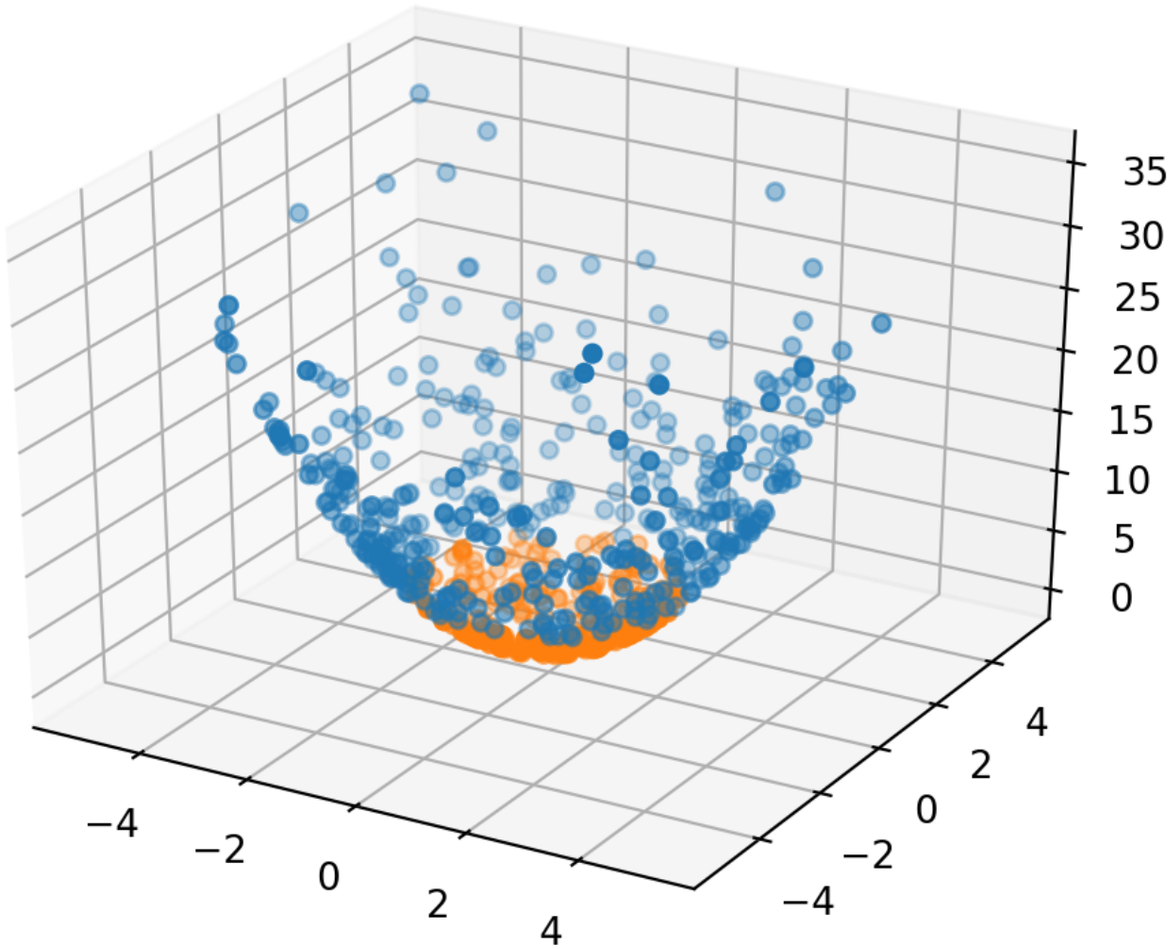
Example

- For example, we might apply the transformation:

$$\phi(x, y) = \begin{bmatrix} x \\ y \\ x^2 + y^2 \end{bmatrix}$$

Here, we transform each 2-D x into a 3-D ϕ(x).

Now, a hyperplane perpendicular to the *z* axis perfectly separates the two classes.



Feature engineering

- Deciding on a suitable transformation of the raw input space to make the data more amenable to classification is sometimes called **feature engineering**.

- Traditionally, this has been performed by hand using domain knowledge of the application domain.

- More recently (with deep neural networks), this is performed implicitly by the training process itself (more on this later).

Feature engineering: example 1

- Suppose you are forecasting stock prices based on historical data.

- One useful predictor might be the volatility of the stock during the past month.

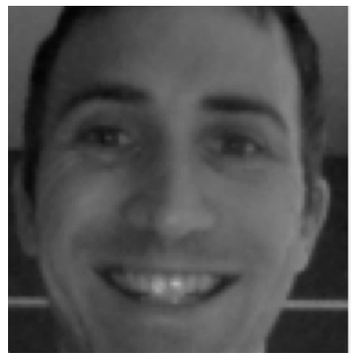
- We can measure the change of the stock price relative to the previous day's price with variable $\Delta t = x_t - x_{t-1}$.

- Because we care more about the absolute change than the sign of the change, we use $(\Delta t)^2$ as a feature rather than the "raw" value Δt.

Feature engineering: example 2

- For classifying facial expression, it can be useful to focus on “edges” in the image, e.g., due to dimples, wrinkles, eyebrows, etc.

- Instead of classifying the raw image…



- … we can instead classify a *filtered* image: