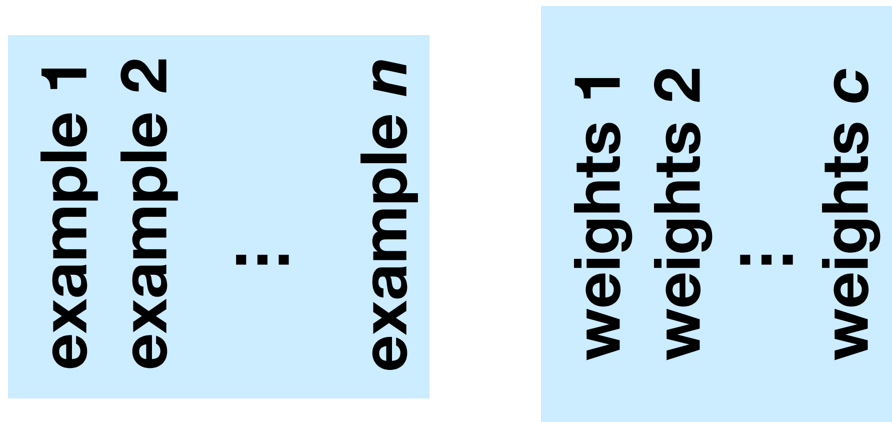
CS 4342: Class 14

Jacob Whitehill

Exercise

- Given are matrices **A** and **B** below:



A **B**

1. What are the ways to compute pre-activation scores?

   $\mathbf{C}=\mathbf{AB}$ $\mathbf{C}=\mathbf{BA}$

   $\mathbf{C}=\mathbf{A}^T\mathbf{B}$ $\mathbf{C}=\mathbf{B}^T\mathbf{A}$

   $\mathbf{C}=\mathbf{AB}^T$ $\mathbf{C}=\mathbf{BA}^T$

   $\mathbf{C}=\mathbf{A}^T\mathbf{B}^T$ $\mathbf{C}=\mathbf{B}^T\mathbf{A}^T$

2. How should exp(**C**) be normalized (row-wise or column-wise) to implement softmax for each of your choices?

Support vectors

SVM optimization problem

- Putting the parts together, we wish to:
  - Minimize: $\frac{1}{2}\mathbf{w}^\top \mathbf{w}$
  - Subject to: $y^{(i)}(\mathbf{x}^{(i)^\top}\mathbf{w} + b) \geq 1 \quad \forall i$
- This is a **quadratic programming** problem: quadratic objective with linear inequality (and/or equality) constraints.
- There are many efficient solvers for quadratic programs.

SVM optimization problem

- We can get some intuition by doing some analytical simplification.

- Similar as with Lagrange multipliers, with KKT conditions we also define a function *L* of the optimization variables (**w**) and the dual variables (*α*):

$$L(\mathbf{w}, b, \alpha) \quad = \quad \underbrace{\frac{1}{2}\mathbf{w}^\top \mathbf{w}}_{\text{Objective}} - \underbrace{\sum_{i=1}^{n} \alpha^{(i)} \left(y^{(i)} \left(\mathbf{x}^{(i)\top} \mathbf{w} + b - 1 \right) \right)}_{\text{Inequality constraints}}$$

SVM optimization problem

- We can get some intuition by doing some analytical simplification.
- Similar as with Lagrange multipliers, with KKT conditions we also define a function *L* of the optimization variables (**w**) and the dual variables (*α*):

$$L(\mathbf{w}, b, \alpha) \quad = \quad \frac{1}{2}\mathbf{w}^\top\mathbf{w} - \sum_{i=1}^{n} \alpha^{(i)} \left(y^{(i)} \left(\mathbf{x}^{(i)}{}^\top \mathbf{w} + b - 1 \right) \right)$$

Objective Inequality constraints

- We then compute the gradient of *L*, set to 0, and solve (numerically)…

SVM optimization problem

- As shown below, an optimal **w** will always be a **linear combination** of the data points $\mathbf{x}^{(i)}$, weighted by the $\alpha^{(i)}$.

$$L(\mathbf{w}, b, \alpha) \quad = \quad \frac{1}{2}\mathbf{w}^\top \mathbf{w} - \sum_{i=1}^{n} \alpha^{(i)} \left(y^{(i)} \left(\mathbf{x}^{(i)\top} \mathbf{w} + b - 1 \right) \right)$$

SVM optimization problem

- As shown below, an optimal **w** will always be a **linear combination** of the data points $\mathbf{x}^{(i)}$, weighted by the $\alpha^{(i)}$.

$$
\begin{aligned}
L(\mathbf{w}, b, \alpha) &= \frac{1}{2}\mathbf{w}^\top \mathbf{w} - \sum_{i=1}^{n} \alpha^{(i)} \left(y^{(i)} \left(\mathbf{x}^{(i)\top} \mathbf{w} + b - 1 \right) \right) \\
\frac{\partial L}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^{n} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}
\end{aligned}
$$

SVM optimization problem

- As shown below, an optimal **w** will always be a **linear combination** of the data points $\mathbf{x}^{(i)}$, weighted by the $\alpha^{(i)}$.

$$
\begin{aligned}
L(\mathbf{w}, b, \alpha) &= \frac{1}{2}\mathbf{w}^\top\mathbf{w} - \sum_{i=1}^n \alpha^{(i)} \left(y^{(i)} \left(\mathbf{x}^{(i)\top}\mathbf{w} + b - 1 \right) \right) \\
\frac{\partial L}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^n \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)} \\
\implies \mathbf{w} &= \sum_{i=1}^n \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}
\end{aligned}
$$

SVM optimization problem

- As shown below, an optimal **w** will always be a **linear combination** of the data points $\mathbf{x}^{(i)}$, weighted by the $\alpha^{(i)}$.
- **w** is therefore a linear combination of the training data $\{ \mathbf{x}^{(i)} \}$.

$$
\begin{aligned}
L(\mathbf{w}, b, \alpha) &= \frac{1}{2}\mathbf{w}^\top\mathbf{w} - \sum_{i=1}^n \alpha^{(i)} \left(y^{(i)} \left(\mathbf{x}^{(i)\top}\mathbf{w} + b - 1 \right) \right) \\
\frac{\partial L}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^n \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)} \\
\implies \mathbf{w} &= \sum_{i=1}^n \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}
\end{aligned}
$$

SVM optimization problem

- As shown below, an optimal **w** will always be a **linear combination** of the data points $\mathbf{x}^{(i)}$, weighted by the $\alpha^{(i)}$.

- As mentioned earlier, only some of the *n* constraints will be active — for the others (inactive), $\alpha^{(i)} = 0$.

$$
\begin{aligned}
L(\mathbf{w}, b, \alpha) &= \frac{1}{2}\mathbf{w}^\top \mathbf{w} - \sum_{i=1}^n \alpha^{(i)} \left(y^{(i)} \left(\mathbf{x}^{(i)\top} \mathbf{w} + b - 1 \right) \right) \\
\frac{\partial L}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^n \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)} \\
\implies \mathbf{w} &= \sum_{i=1}^n \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}
\end{aligned}
$$

Karush-Kuhn-Tucker (KKT) conditions

- With KKT conditions we use a set of “multipliers” α (one for each constraint), sometimes known as **dual variables**.

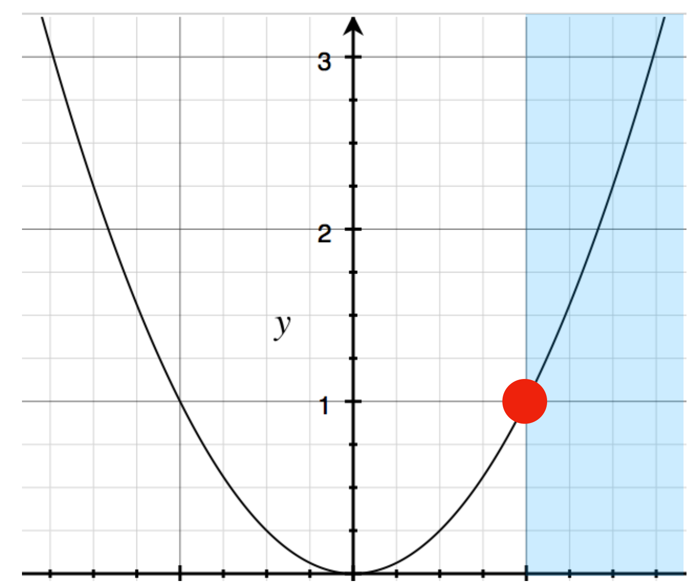
$$L(\mathbf{w}, \alpha) = f(\mathbf{w}) - \sum_{i=1}^{n} \alpha_i g_i(\mathbf{w})$$

- Key points:

1. With *inequality* constraints, we require that each $\alpha_i \geq 0$.

2. At optimal solution:

    - $\alpha_i > 0$ if the constraint is **active**.
    - $\alpha_i = 0$ if the constraint is inactive.



$g(x)=x$

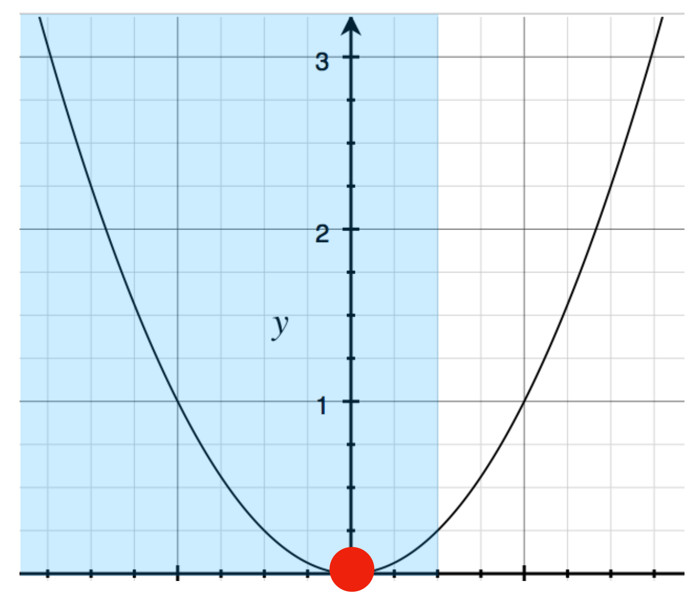
Karush-Kuhn-Tucker (KKT) conditions

- With KKT conditions we use a set of “multipliers” α (one for each constraint), sometimes known as **dual variables.**

$$L(\mathbf{w}, \alpha) = f(\mathbf{w}) - \sum_{i=1}^{n} \alpha_i g_i(\mathbf{w})$$

- Key points:

  1. With *inequality* constraints, we require that each $\alpha_i \geq 0$.
  2. At optimal solution:
     - $\alpha_i > 0$ if the constraint is active.
     - $\alpha_i = 0$ if the constraint is **inactive.**



$g(x)=x-1/2$

Jacob Whitehill, WPI

SVM optimization problem

- This means that **w** will actually only be a linear combination of a *subset* of the input vectors $\mathbf{x}^{(i)}$.
  - The data $\mathbf{x}^{(i)}$ for which $\alpha^{(i)} > 0$ are called **support vectors.**

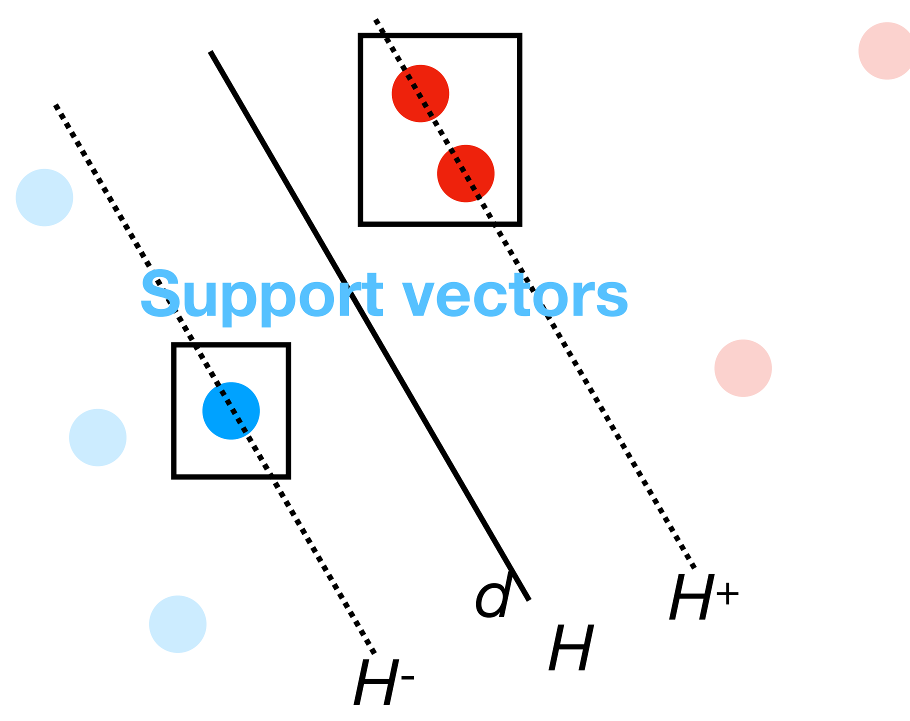
- The other data (for which $\alpha^{(i)} = 0$) are essentially *irrelevant — they do not influence the location or orientation of the hyperplane.*

$$
\begin{aligned}
L(\mathbf{w}, b, \alpha) &= \frac{1}{2}\mathbf{w}^\top \mathbf{w} - \sum_{i=1}^{n} \alpha^{(i)} \left(y^{(i)} \left(\mathbf{x}^{(i)^\top} \mathbf{w} + b - 1 \right) \right) \\
\frac{\partial L}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^{n} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)} \\
\Longrightarrow \mathbf{w} &= \sum_{i=1}^{n} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}
\end{aligned}
$$

SVM optimization problem

- This means that **w** will actually only be a linear combination of a *subset* of the input vectors $\mathbf{x}^{(i)}$.
  - The data $\mathbf{x}^{(i)}$ for which $\alpha^{(i)} > 0$ are called **support vectors**.

- The other data (for which $\alpha^{(i)} = 0$) are essentially *irrelevant — they do not influence the location or orientation of the hyperplane.*



LinearSVC

- In `sklearn`, the `LinearSVC` class is for training linear SVMs.
    - Soon, we will also examine non-linear SVMs.
- API:
    - `class sklearn.svm.LinearSVC(penalty='l2', loss='squared_hinge', dual=True, tol=0.0001, C=1.0, …)`
- What do `C` and `dual` refer to?

LinearSVC

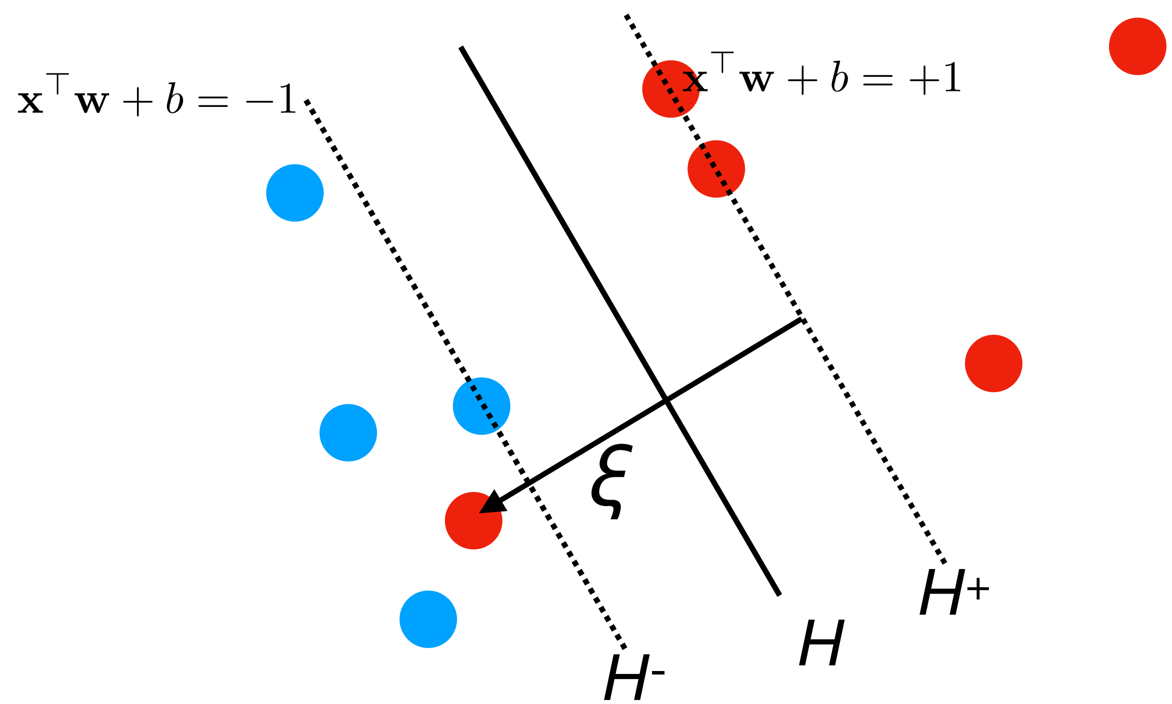
- `C`:
  - Error penalty for misclassification in soft-margin SVMs
- `dual`:
  - Primal versus dual optimization approach

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: $$\frac{1}{2}\mathbf{w}^\top\mathbf{w} + \overset{\text{Error penalty}}{C}\sum_{i=1}^{n}\xi^{(i)}$$
  - Subject to: $$y^{(i)}\left(\mathbf{x}^{(i)^\top}\mathbf{w} + b\right) \geq 1 \underset{\text{Slack}}{- \xi^{(i)}}$$

SVM: optimization of the dual problem

Dual form

- Recall that, in an SVM, the optimal **w** will always be a **linear combination** of the data points $\mathbf{x}^{(i)}$, weighted by the $\alpha^{(i)}$.

- Only the support vectors — those examples $\mathbf{x}^{(i)}$ such that $\alpha^{(i)} > 0$ — will contribute to **w**:

$$
\begin{aligned}
L(\mathbf{w}, b, \alpha) &= \frac{1}{2}\mathbf{w}^\top\mathbf{w} - \sum_{i=1}^{n} \alpha^{(i)} \left(y^{(i)} \left(\mathbf{x}^{(i)\top}\mathbf{w} + b - 1 \right) \right) \\
\frac{\partial L}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^{n} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)} \\
\Longrightarrow \mathbf{w} &= \sum_{i=1}^{n} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}
\end{aligned}
$$

Dual form

- This also suggests a different way of optimizing an SVM:
  - Instead of optimizing over $\mathbf{w} \in \mathbb{R}^m$, where *m* is size of the feature vector (e.g., number of image pixels), we can optimize over $\alpha \in \mathbb{R}^n$, where *n* is the number of training examples.

$$
\begin{aligned}
L(\mathbf{w}, b, \alpha) &= \frac{1}{2}\mathbf{w}^\top \mathbf{w} - \sum_{i=1}^n \alpha^{(i)} \left(y^{(i)} \left(\mathbf{x}^{(i)\top} \mathbf{w} + b - 1 \right) \right) \\
\frac{\partial L}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^n \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)} \\
\implies \mathbf{w} &= \sum_{i=1}^n \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}
\end{aligned}
$$

Dual form

- This also suggests a different way of optimizing an SVM:
  - Instead of optimizing over $\mathbf{w} \in \mathbb{R}^m$ (Primal variables), where *m* is size of the feature vector (e.g., number of image pixels), we can optimize over $\alpha \in \mathbb{R}^n$ (Dual variables), where *n* is the number of training examples.

$$
\begin{aligned}
L(\mathbf{w}, b, \alpha) &= \frac{1}{2}\mathbf{w}^\top \mathbf{w} - \sum_{i=1}^{n} \alpha^{(i)} \left(y^{(i)} \left(\mathbf{x}^{(i)\top} \mathbf{w} + b - 1 \right) \right) \\
\frac{\partial L}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^{n} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)} \\
\implies \mathbf{w} &= \sum_{i=1}^{n} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}
\end{aligned}
$$

Dual form

- Suppose we are training a smile detector, where the number of features *m* = 10,000 and *n*=1000 (examples).
- Which would you rather optimize: $\mathbf{w} \in \mathbb{R}^m$ or $\alpha \in \mathbb{R}^n$?

$$
\begin{aligned}
L(\mathbf{w}, b, \alpha) &= \frac{1}{2}\mathbf{w}^\top \mathbf{w} - \sum_{i=1}^{n} \alpha^{(i)} \left(y^{(i)} \left({\mathbf{x}^{(i)}}^\top \mathbf{w} + b - 1 \right) \right) \\
\frac{\partial L}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^{n} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)} \\
\implies \mathbf{w} &= \sum_{i=1}^{n} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}
\end{aligned}
$$

Show support_vectors.py demo

Dual form

- Optimizing over *α* instead of **w** is called the **dual form** of the constraint optimization.

- Optimizing **w** directly is called the primal form.

- Both approaches give the *same solution*.

- When the number of examples n < number of features m, it can be faster to train using the dual form.

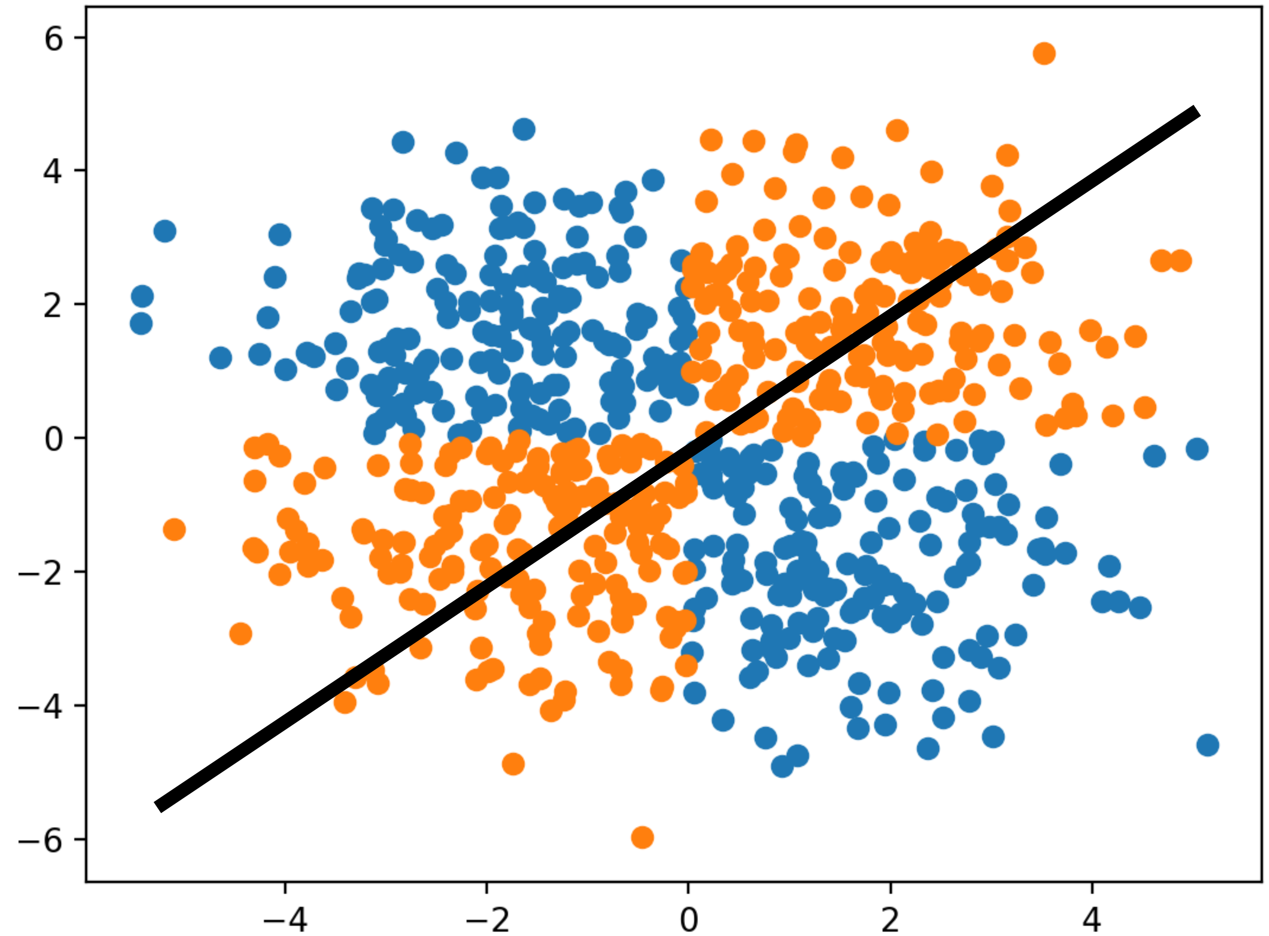
Dual form

- Training the SVM in dual form requires that we transform (**kernelize**) the optimization problem.
- Kernelization is also useful to implement non-linear **feature transformations** using SVMs.

Feature transformations

Linearly inseparable data

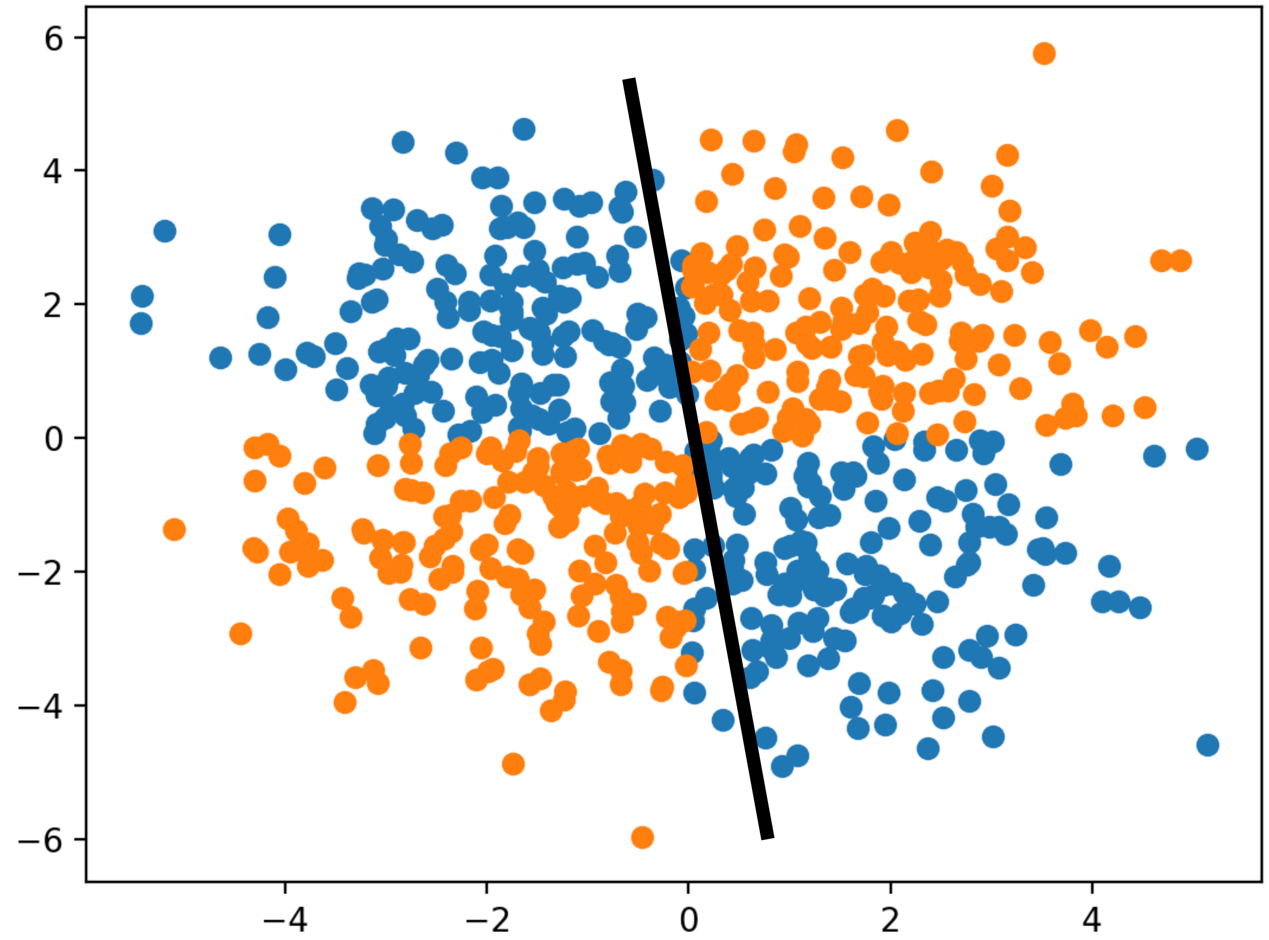
- SVMs use a hyperplane to separate data in two classes.
- But what if the data are **linearly inseparable**, e.g.:
- No matter what **w**, *b* we choose, the SVM will never do a good job of classifying the data.



“XOR” problem

Linearly inseparable data

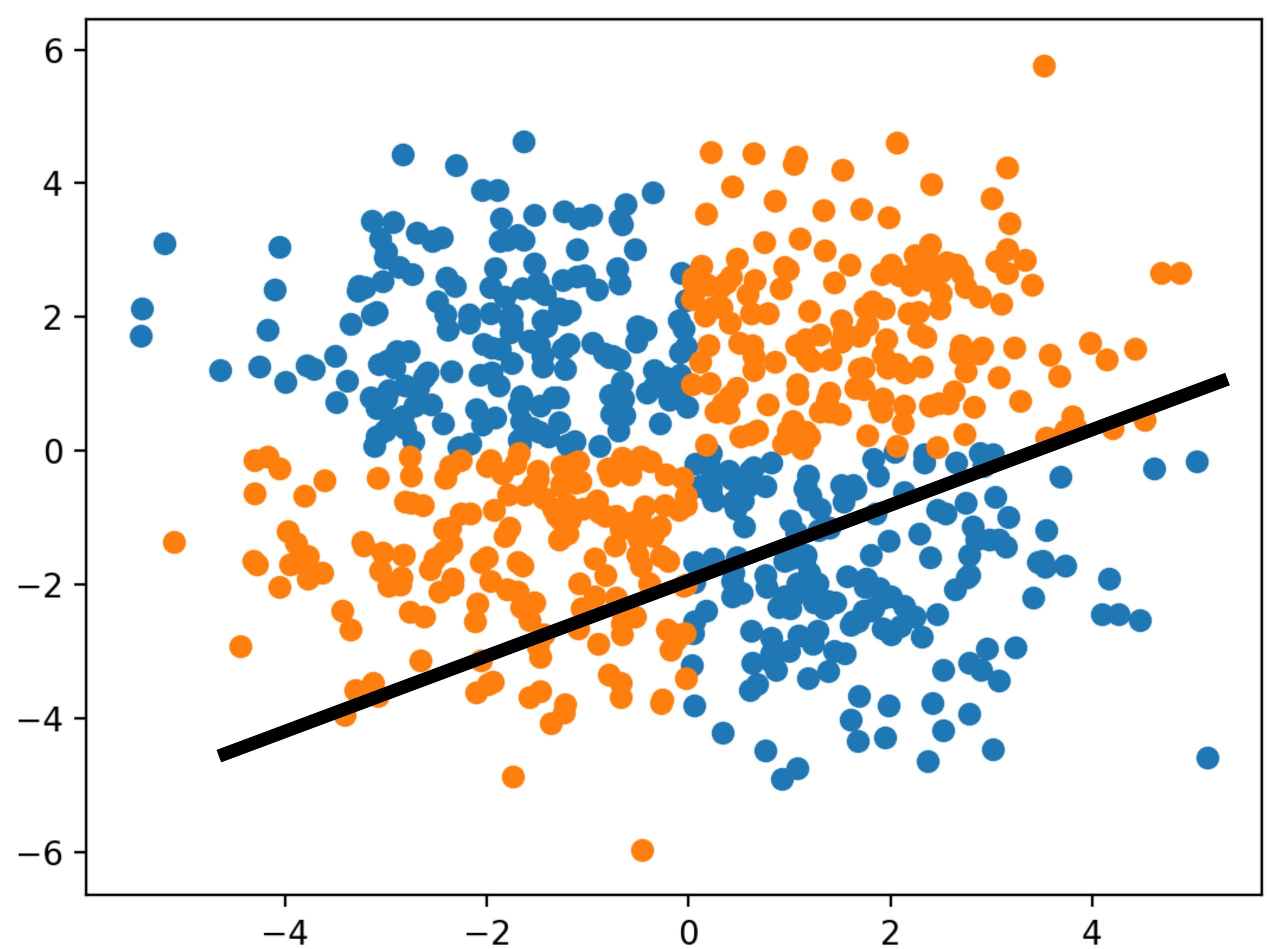
- SVMs use a hyperplane to separate data in two classes.
- But what if the data are **linearly inseparable**, e.g.:
- No matter what **w**, *b* we choose, the SVM will never do a good job of classifying the data.



“XOR” problem

Linearly inseparable data

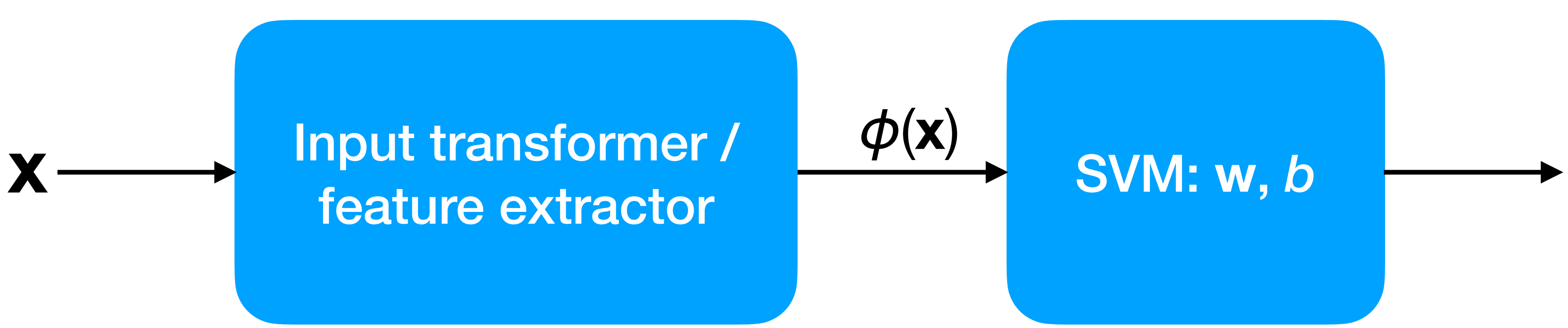
- SVMs use a hyperplane to separate data in two classes.
- But what if the data are **linearly inseparable**, e.g.:
- No matter what **w**, *b* we choose, the SVM will never do a good job of classifying the data.



“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?

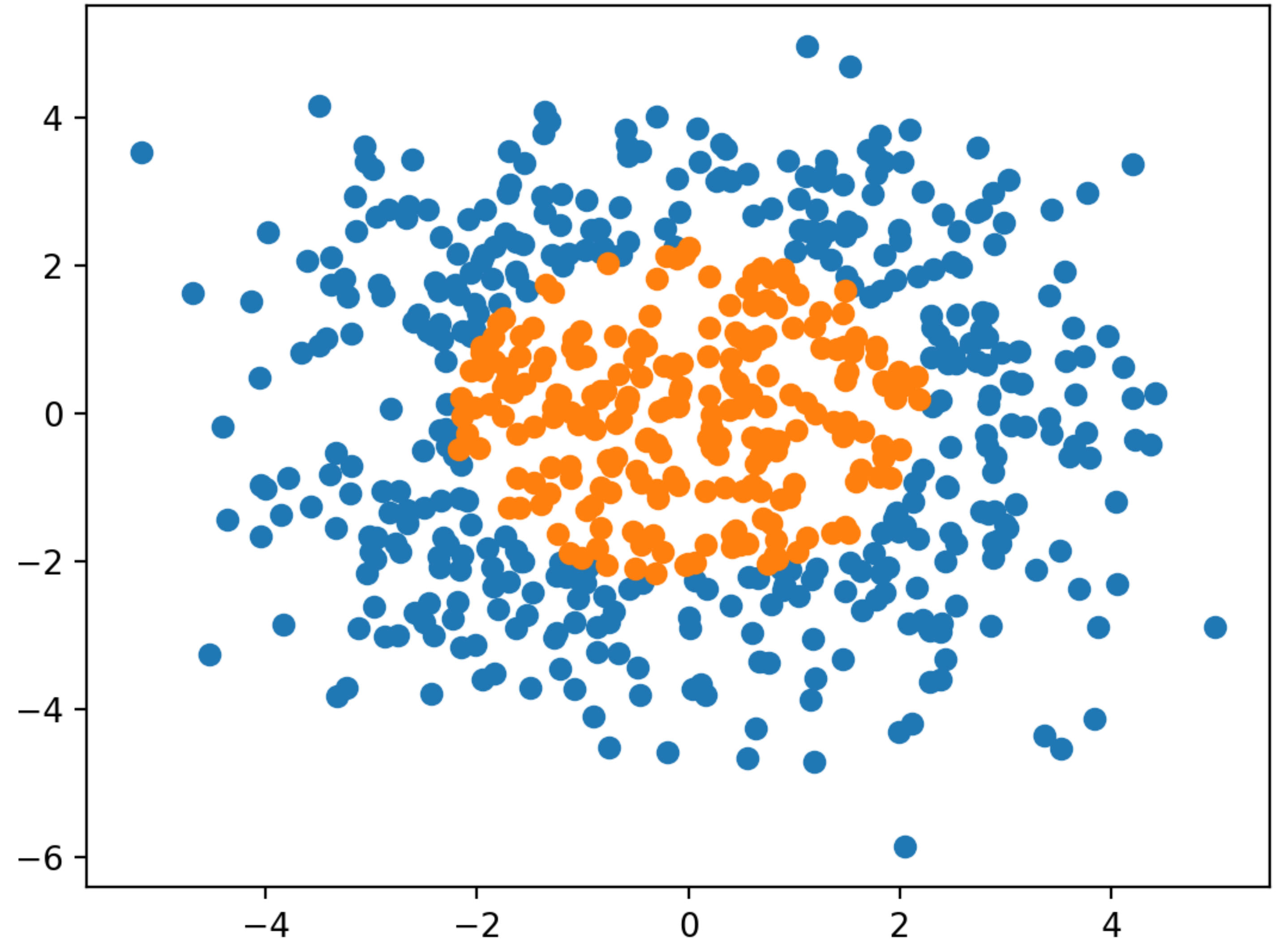


$$\hat{y} = \mathrm{sgn}(\phi(\mathbf{x})^\top \mathbf{w} + b)$$

Can decide class arbitrarily if sgn = 0.

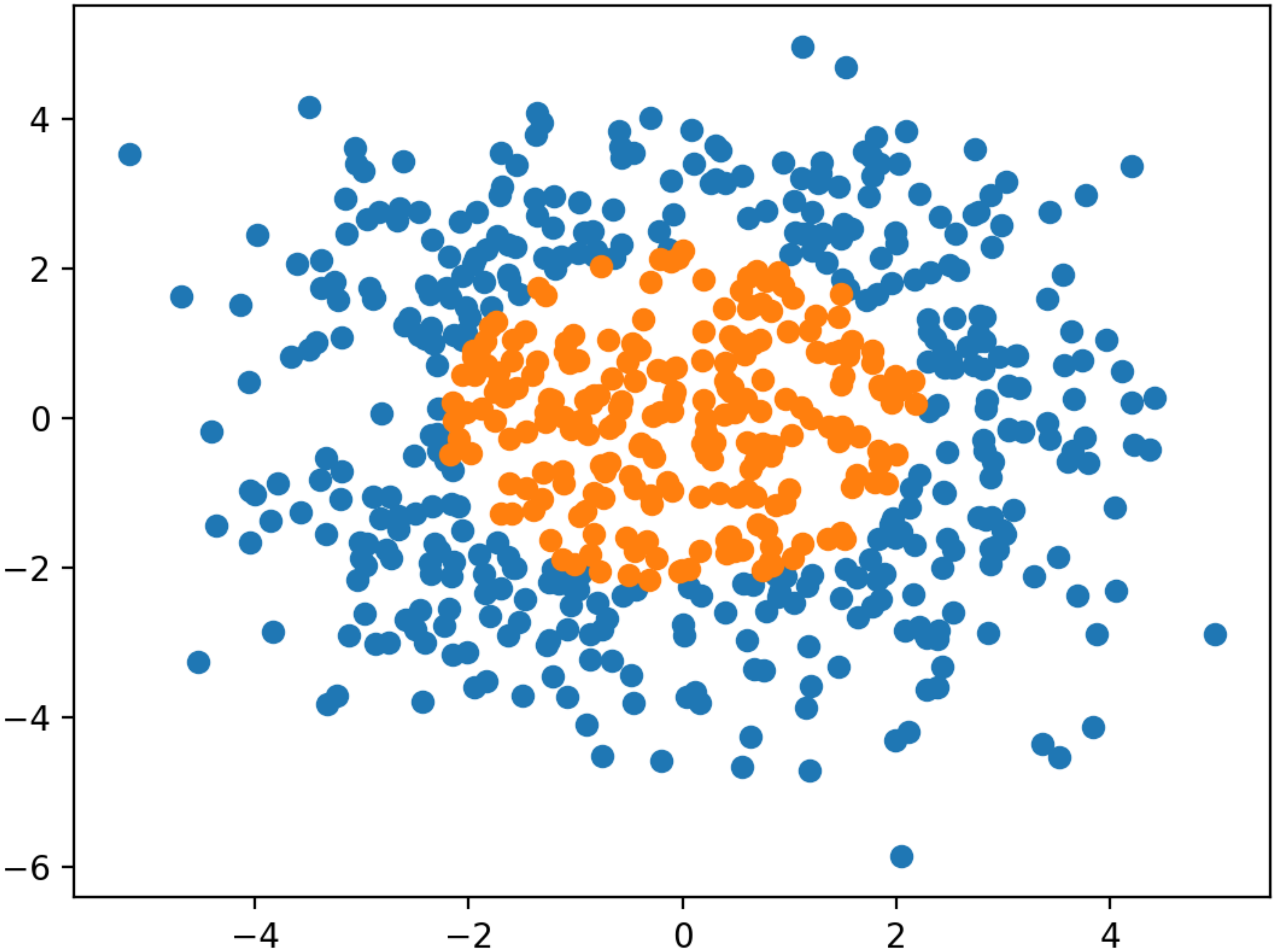
Example

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



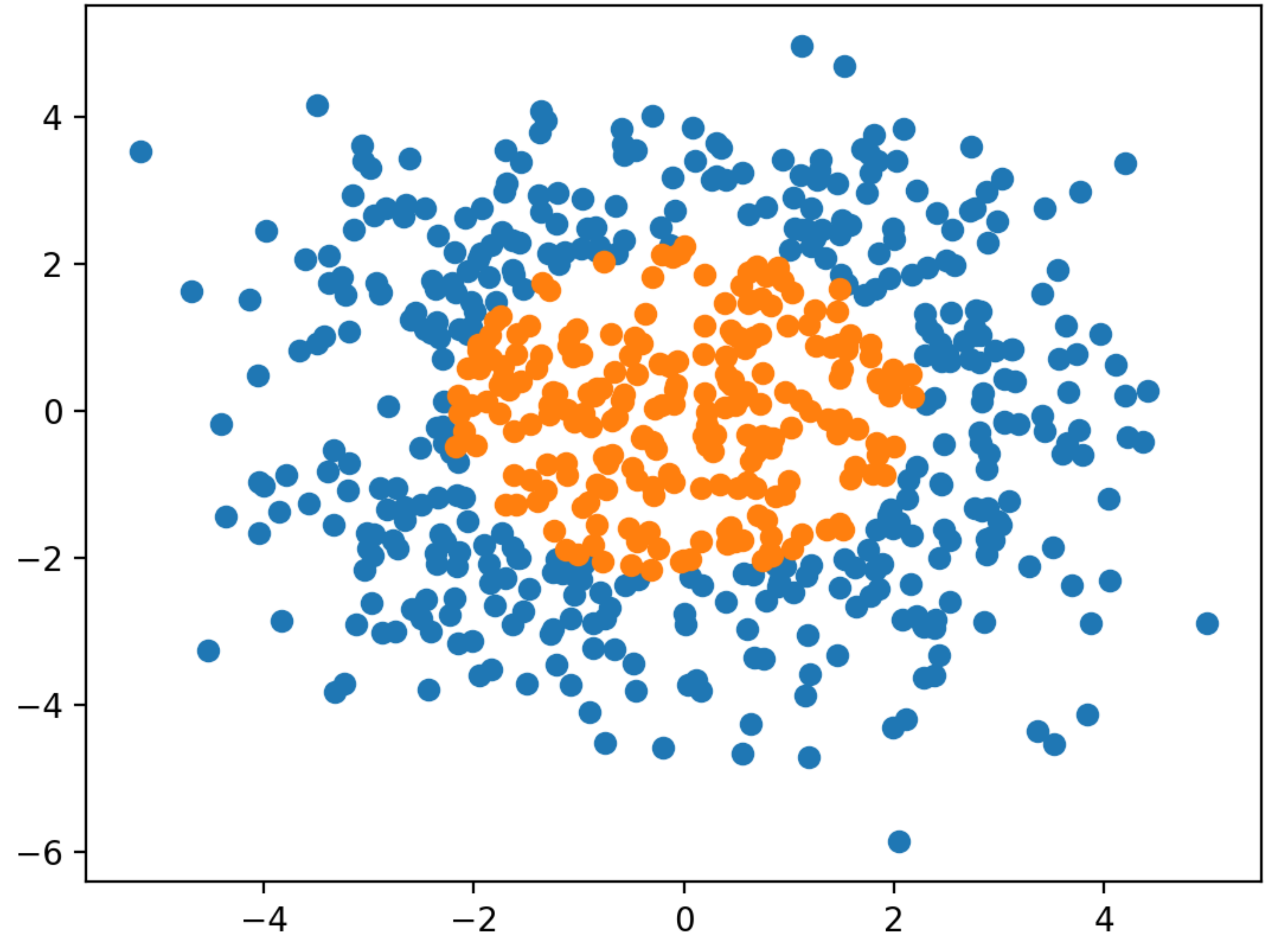
Example

- 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?
  - $x^2 + y^2$

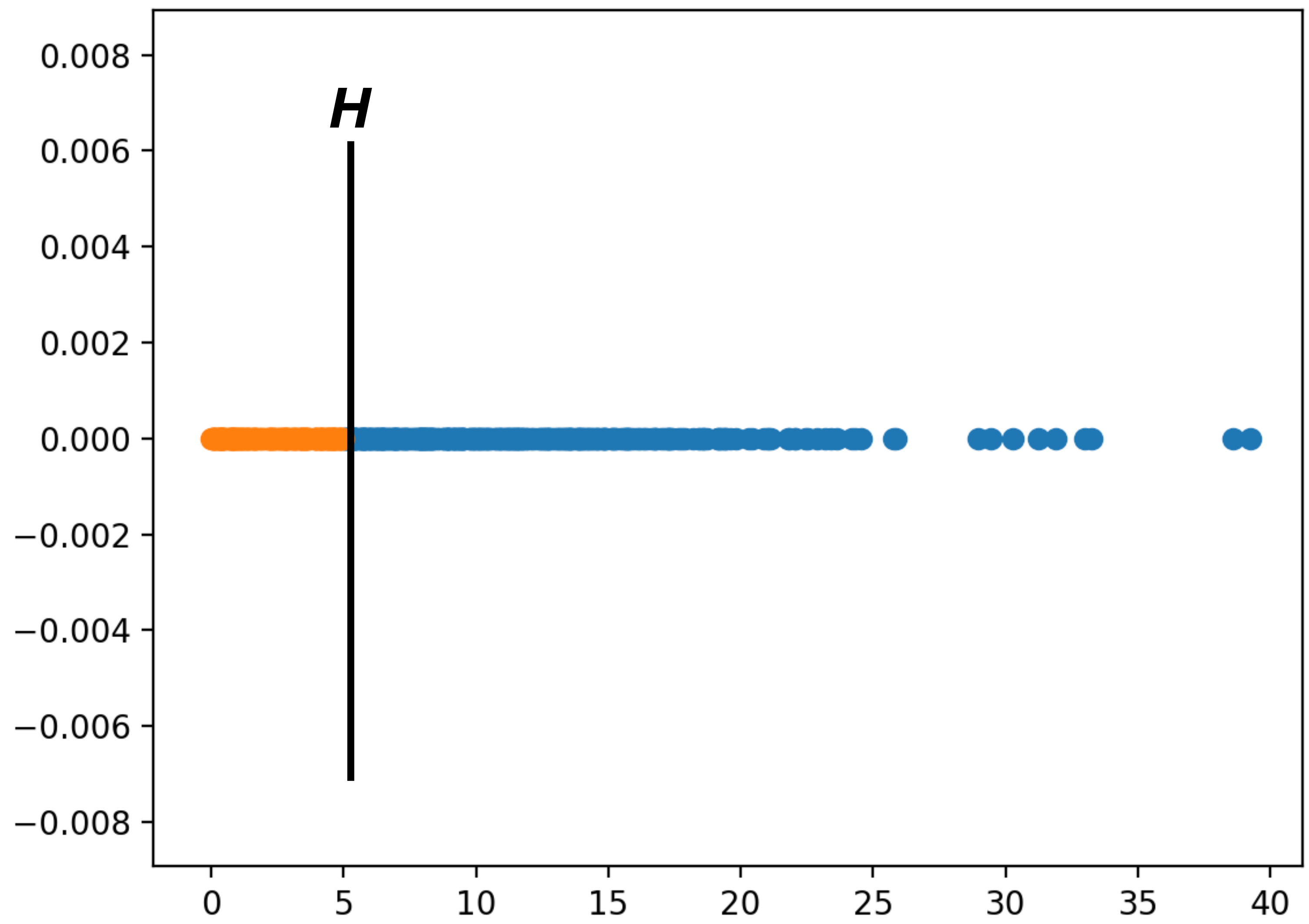


Example

- 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.



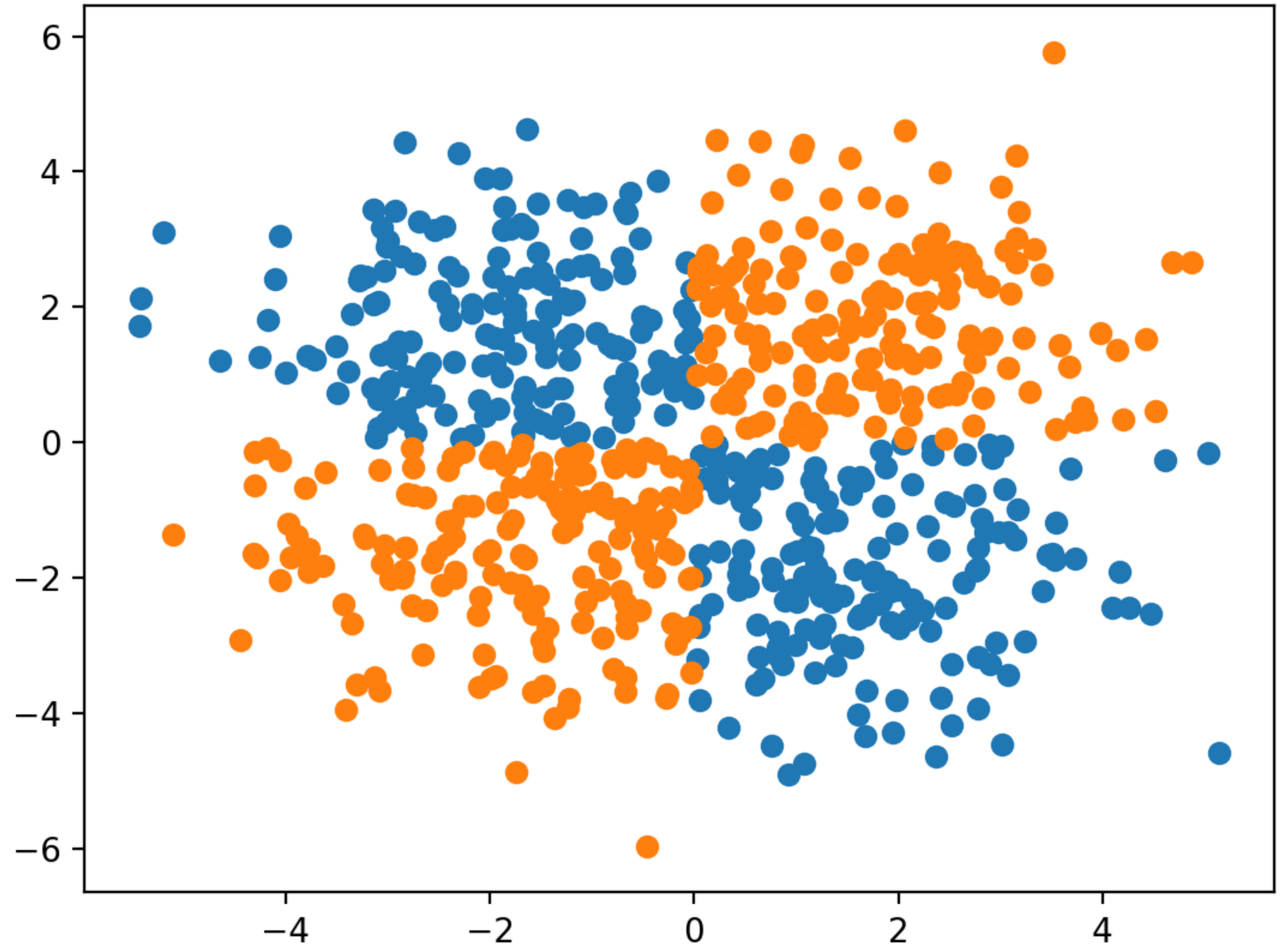
Exercise 1

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

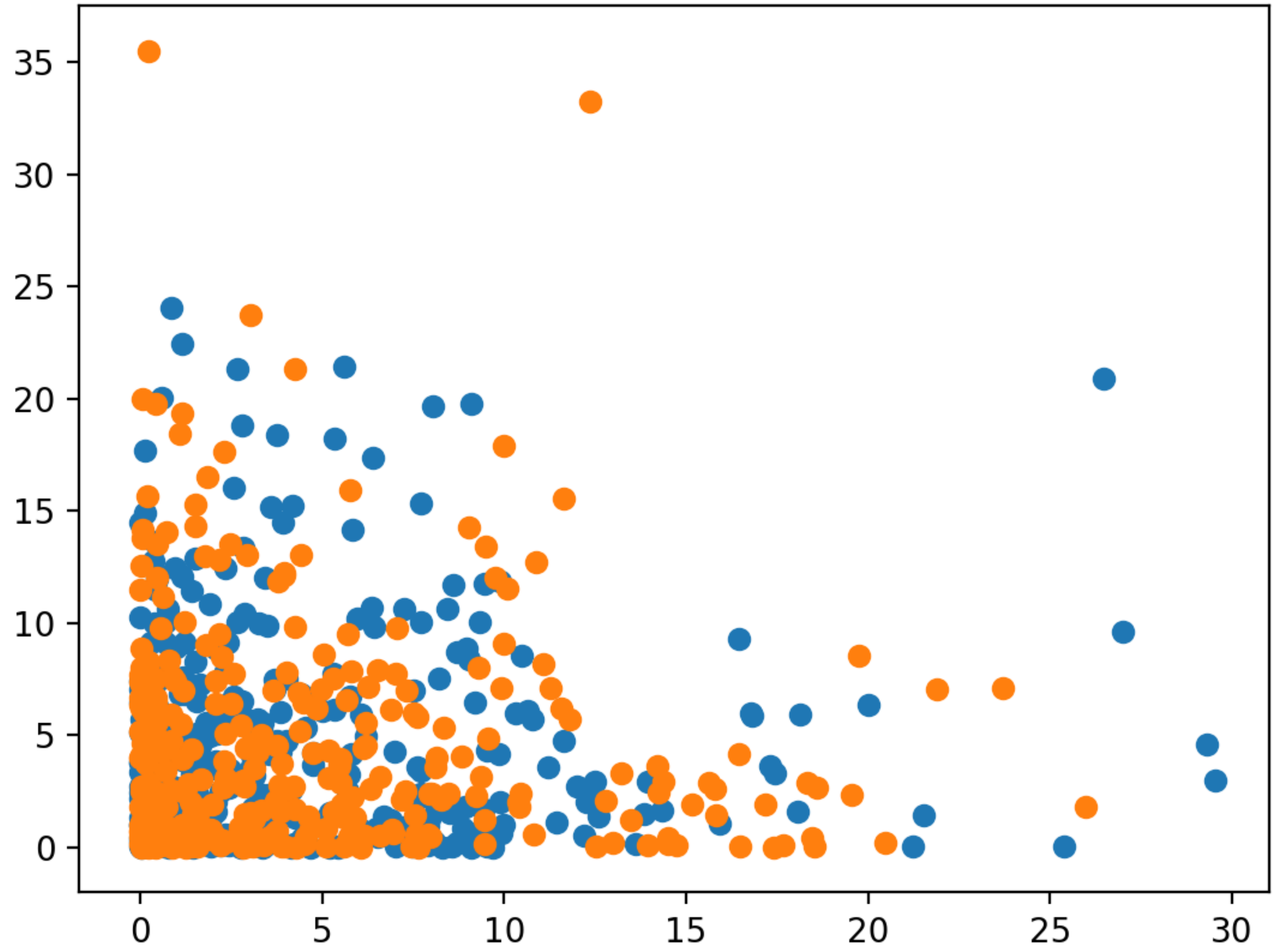


Exercise 1

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

1. $\phi(x,y) = \left[\begin{array}{c} x^2 \\ y^2 \end{array} \right]$

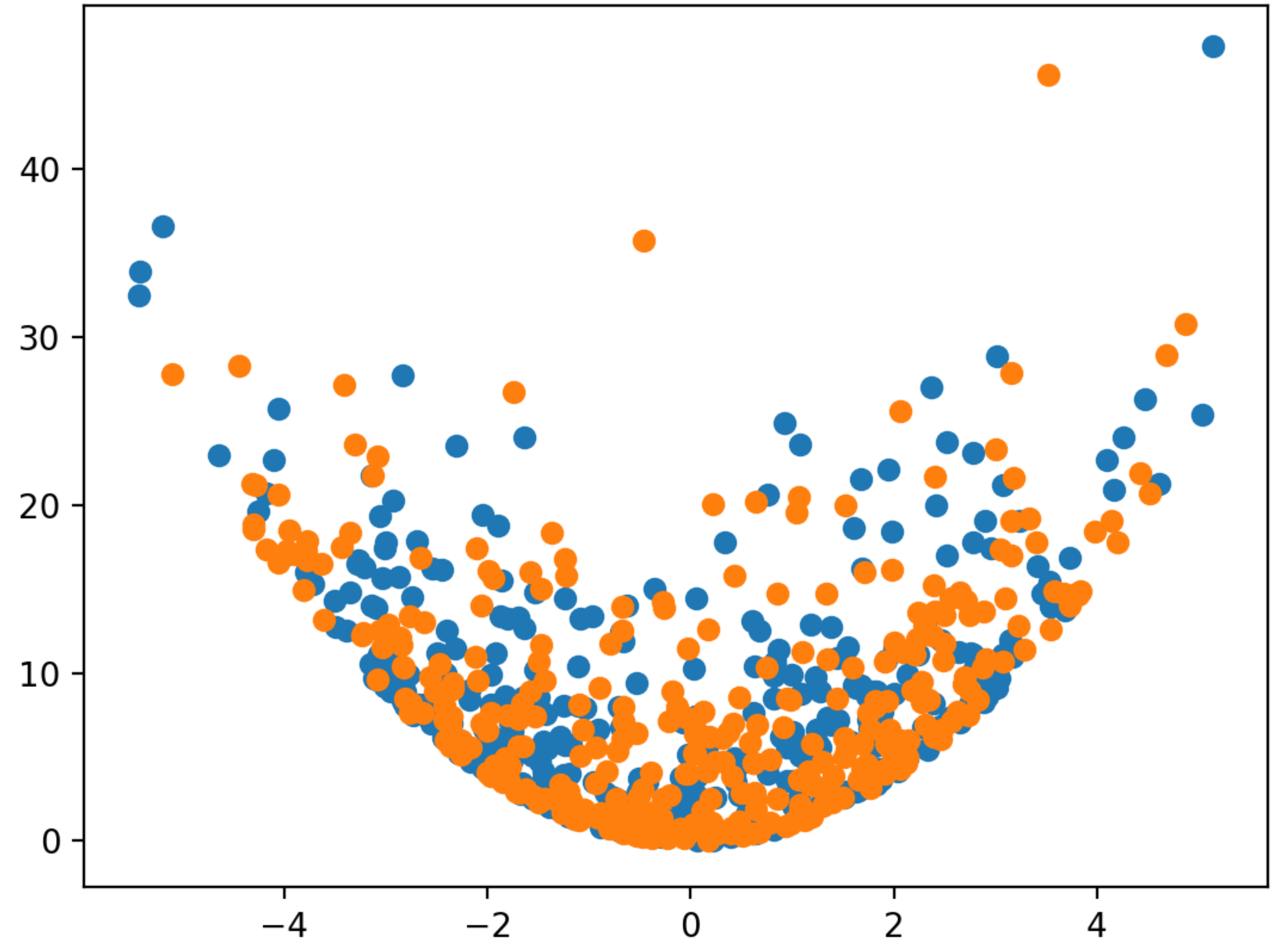
This collapses across both the left-right and up-down half-spaces, but does not render the two classes linearly separable.



Exercise 1

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

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



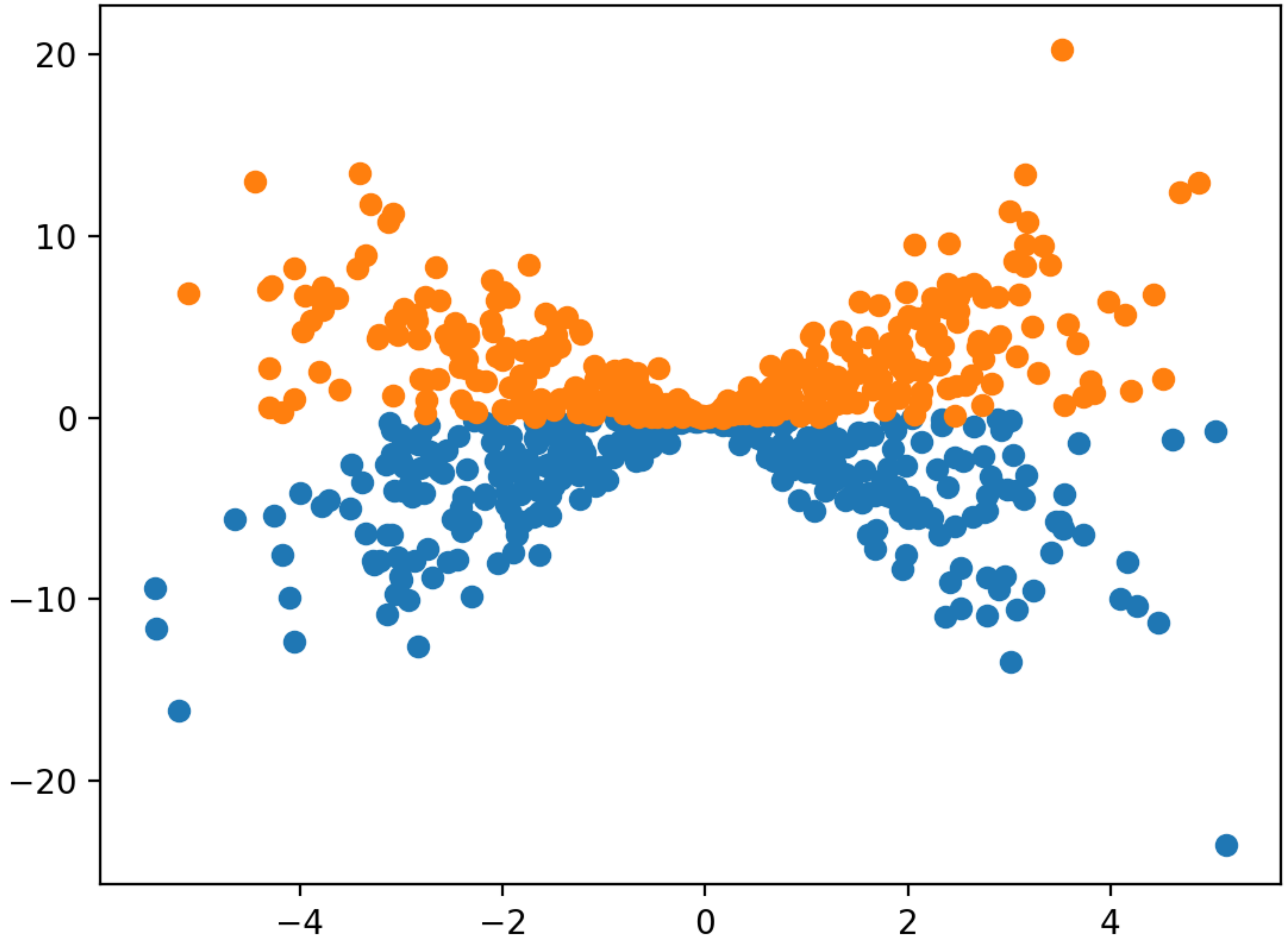
$x^2 + y^2$ computes the distance from the origin, which is not related to the class label in this problem.

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

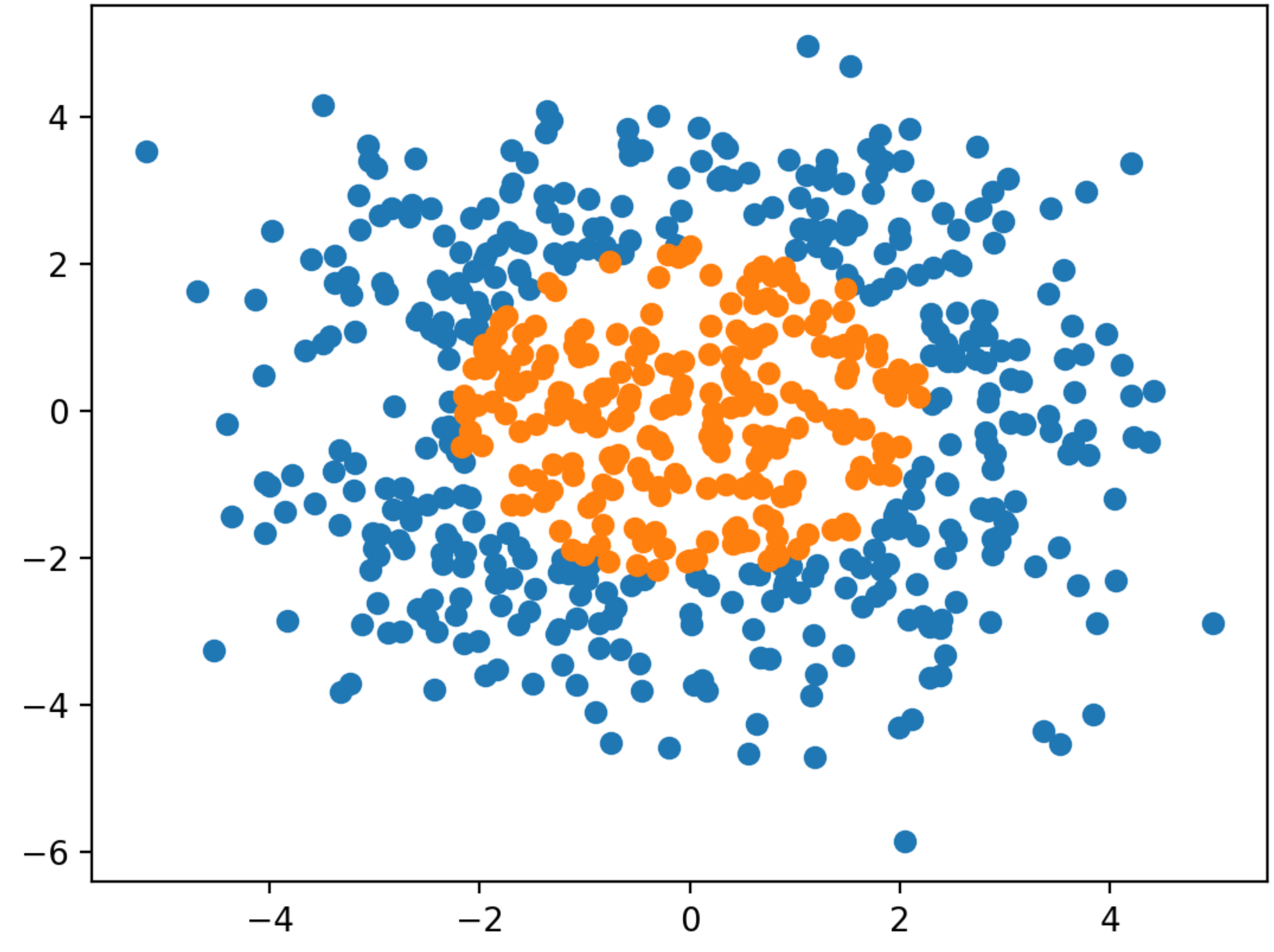


Feature transformations to higher dimensions

- 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.



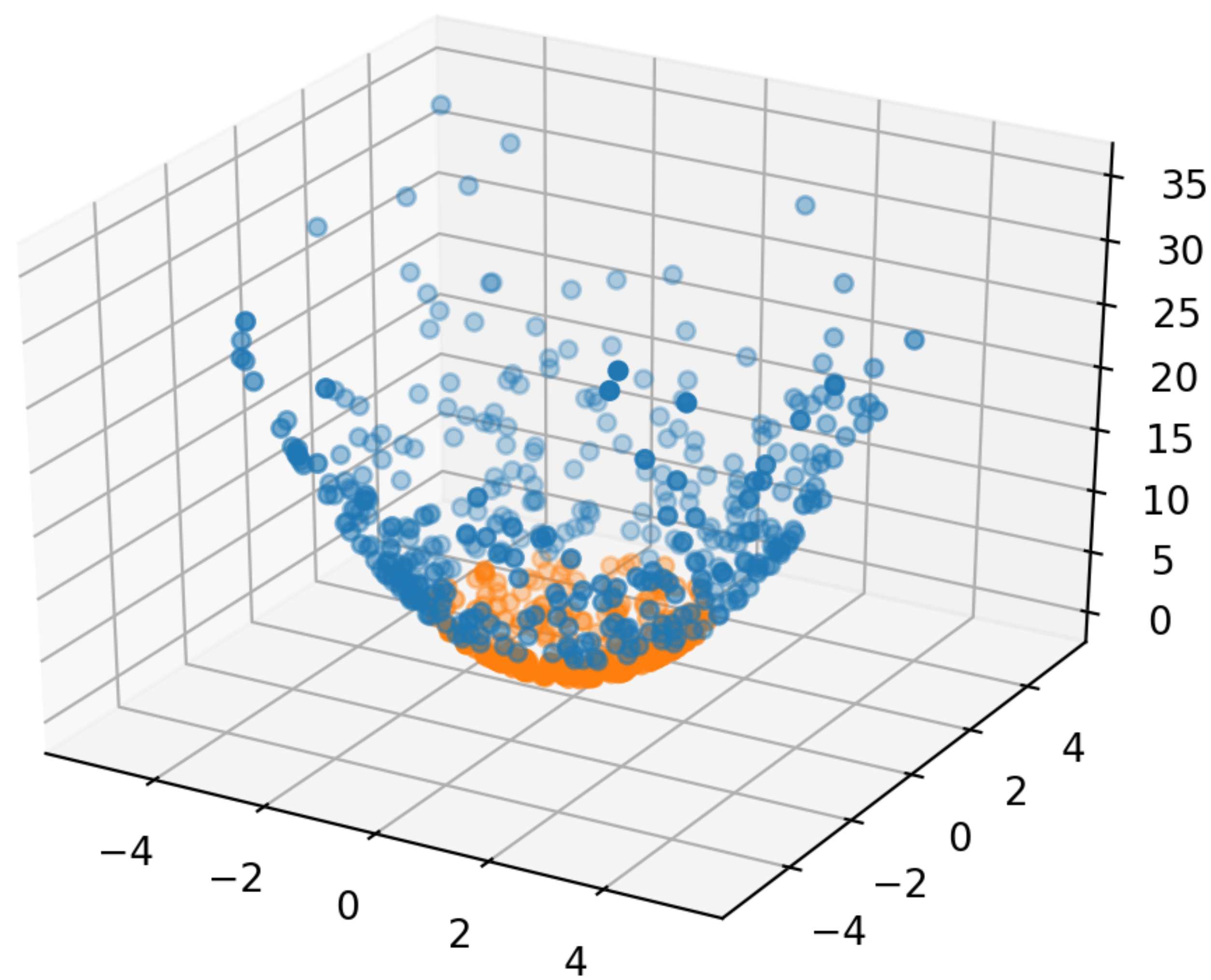
Feature transformations to higher dimensions

- 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.

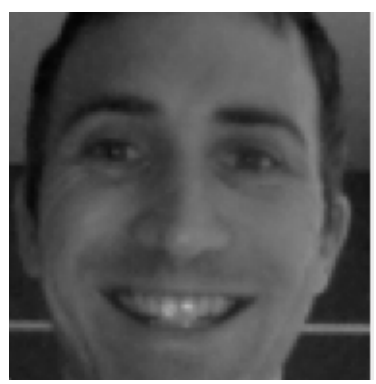
- We can measure the change of the stock price relative to the previous day's price with variable $\Delta t = x_t - x_{t-1}$.

- 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: