

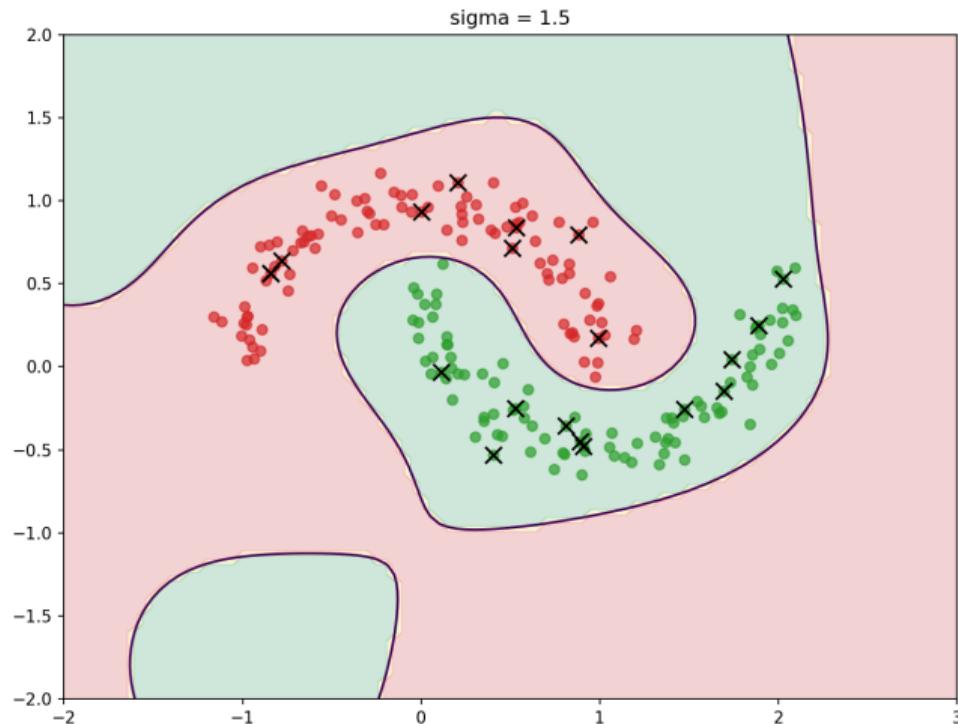
# DSC 190

## Machine Learning: Representations

Lecture 4 | Part 1

### Radial Basis Functions

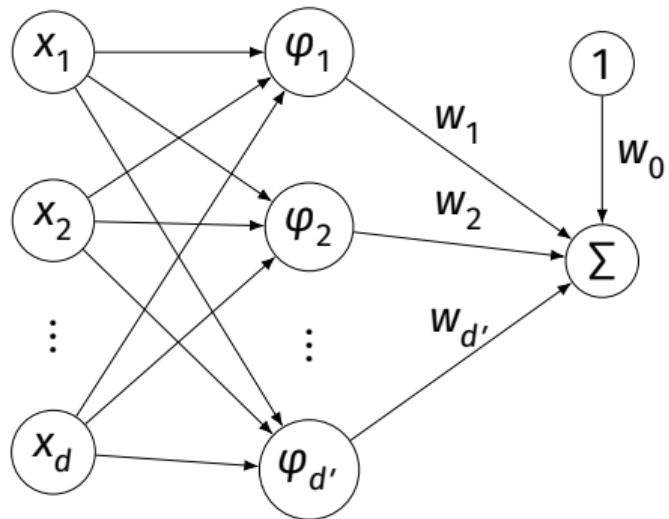
# Today



# Recap

- ▶ Linear prediction functions are limited.
- ▶ Idea: transform the data to a new space where prediction is “easier”.
- ▶ To do so, we used **basis functions**.

$$H(\vec{x}) = w_0 + w_1 \varphi_1(\vec{x}) + w_2 \varphi_2(\vec{x})$$



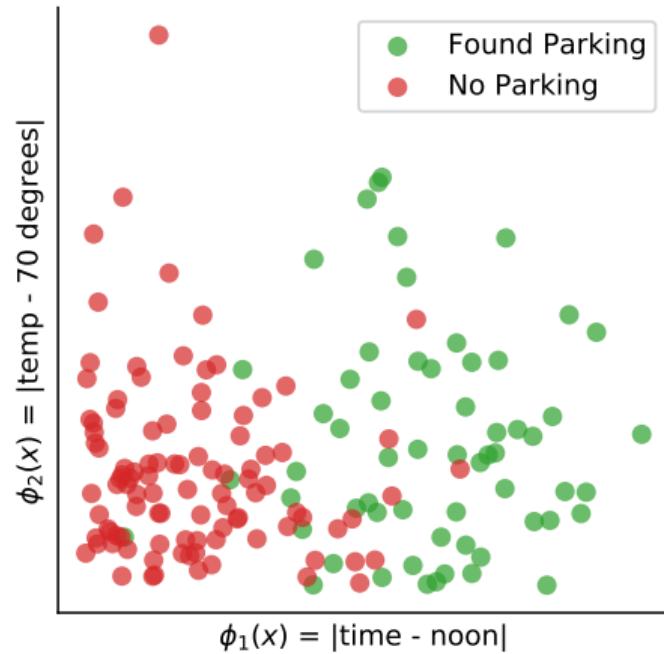
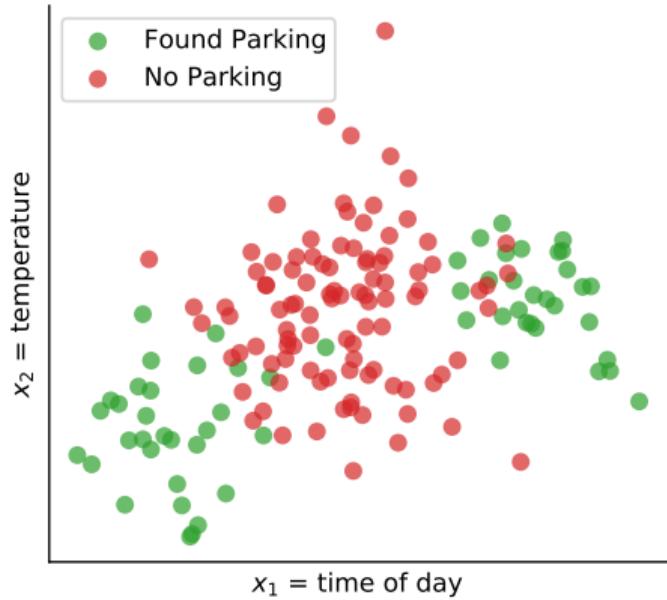
# Overview: Feature Mapping

1. Start with data in original space,  $\mathbb{R}^d$ .
2. Choose some basis functions,  $\varphi_1, \varphi_2, \dots, \varphi_{d'}$
3. Map each data point to **feature space**  $\mathbb{R}^{d'}$ :  
$$\vec{x} \mapsto (\varphi_1(\vec{x}), \varphi_2(\vec{x}), \dots, \varphi_{d'}(\vec{x}))^t$$

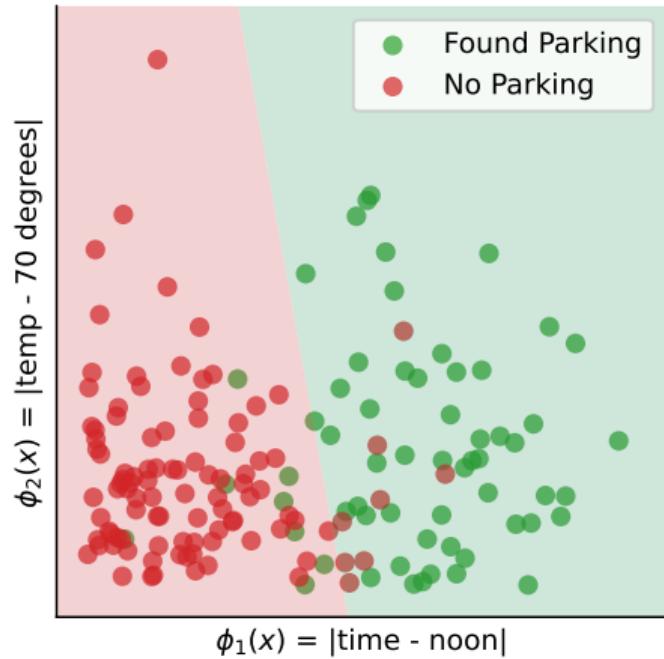
4. Fit linear prediction function in new space:

$$H(\vec{x}) = w_0 + w_1 \varphi_1(\vec{x}) + w_2 \varphi_2(\vec{x})$$

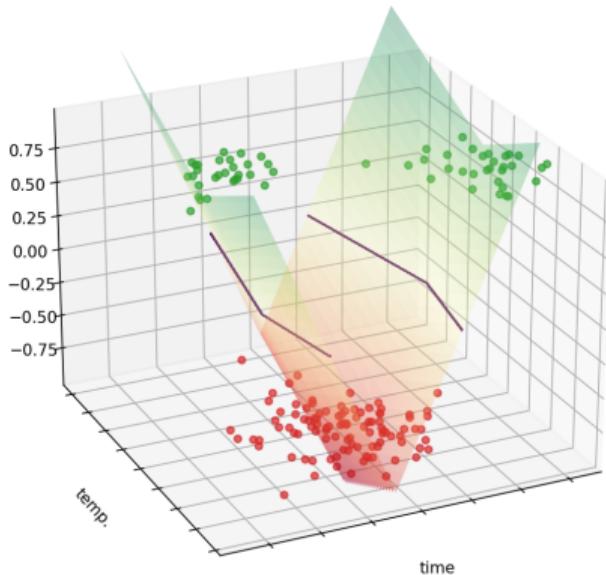
# Last Time



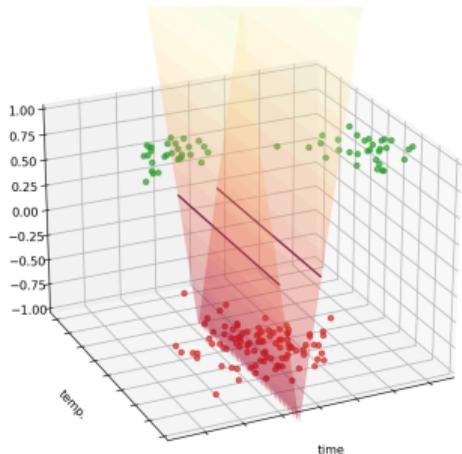
# Last Time



# Visualizing the “Prediction Surface”

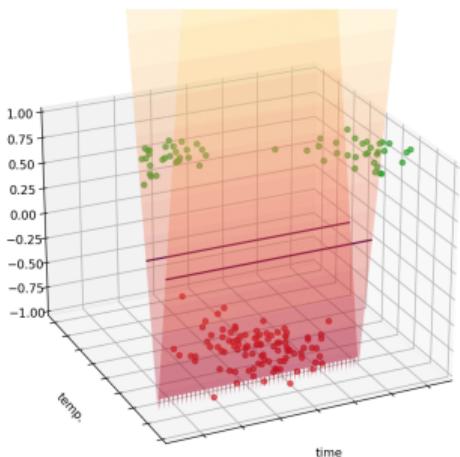


# Visualizing the Basis Function $\varphi_1$



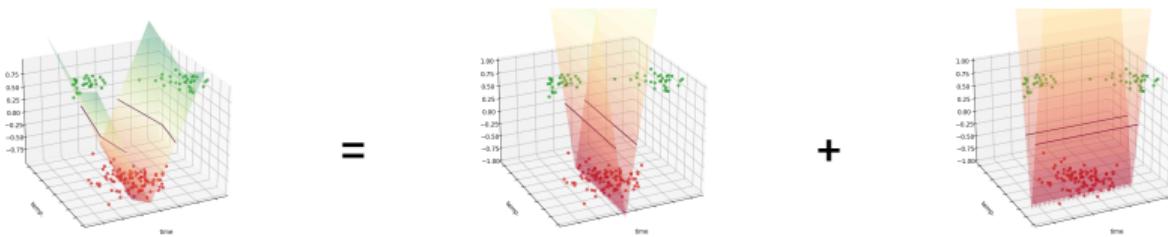
►  $w_0 + w_1 |x_1 - \text{noon}|$

# Visualizing the Basis Function $\varphi_2$



►  $w_0 + w_2 |x_2 - 72^\circ|$

# Visualizing the “Prediction Surface”

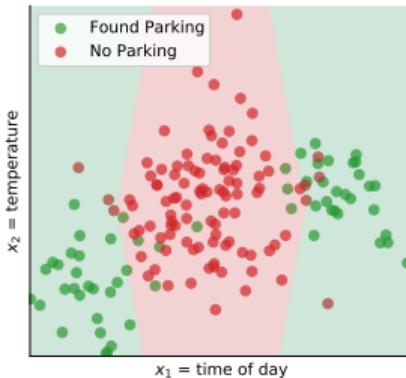


# The Decision Boundary

- ▶ The prediction surface is a sum of other surfaces.
- ▶ Each basis function is a “building block”.
- ▶ The **decision boundary** is where surface = zero.

## Exercise

The decision boundary has a single “pocket” where it is negative. Can it have more than one, assuming we use basis functions of the same form? What if we use more than two basis functions?



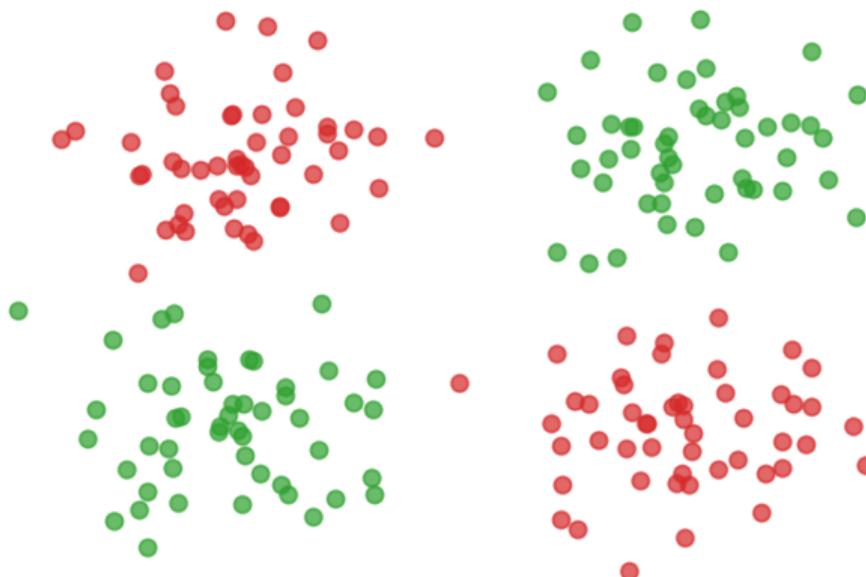
# Answer: No!

- ▶ Recall: the sum of convex functions is convex.
- ▶ Each of our basis functions is convex.
- ▶ So the prediction surface will be convex, too.
- ▶ Limited in what patterns they can classify.

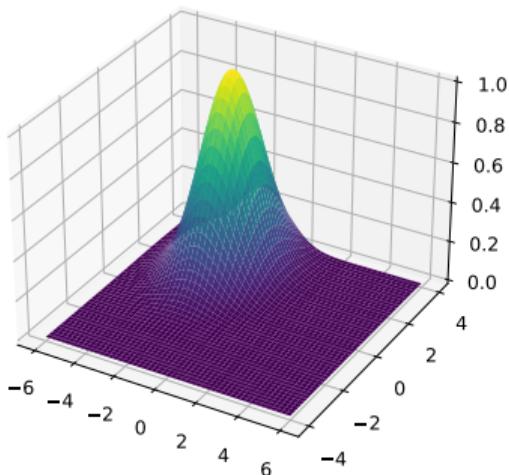
# Choosing Basis Functions

- ▶ Our previous basis functions have limitations.
- ▶ They are convex: prediction surface can only have one negative/positive region.
- ▶ They diverge  $\rightarrow \infty$  away from their centers.
  - ▶ They get more “confident”?

# Example



# Gaussian Basis Functions



- ▶ A common choice: **Gaussian** basis functions:

$$\varphi(\vec{x}; \vec{\mu}, \sigma) = e^{-\|\vec{x} - \vec{\mu}\|^2 / \sigma^2}$$

- ▶  $\vec{\mu}$  is the center.
- ▶  $\sigma$  controls the “width”

# Gaussian Basis Function

- ▶ If  $\vec{x}$  is close to  $\vec{\mu}$ ,  $\varphi(\vec{x}; \vec{\mu}, \sigma)$  is large.
- ▶ If  $\vec{x}$  is far from  $\vec{\mu}$ ,  $\varphi(\vec{x}; \vec{\mu}, \sigma)$  is small.
- ▶ Intuition:  $\varphi$  measures how “similar”  $\vec{x}$  is to  $\vec{\mu}$ .
  - ▶ Assumes that “similar” objects have close feature vectors.

# New Representation

- ▶ Pick number of new features,  $d'$ .
- ▶ Pick centers for Gaussians  $\vec{\mu}^{(1)}, \dots, \vec{\mu}^{(2)}, \dots, \vec{\mu}^{(d')}$
- ▶ Pick widths:  $\sigma_1, \sigma_2, \dots, \sigma_{d'}$  (usually all the same)
- ▶ Define  $i$ th basis function:

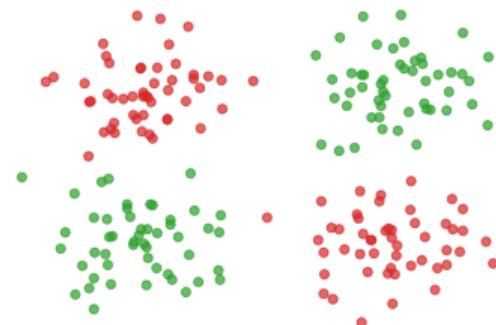
$$\varphi_i(\vec{x}) = e^{-\|\vec{x} - \vec{\mu}^{(i)}\|^2 / \sigma_i^2}$$

# New Representation

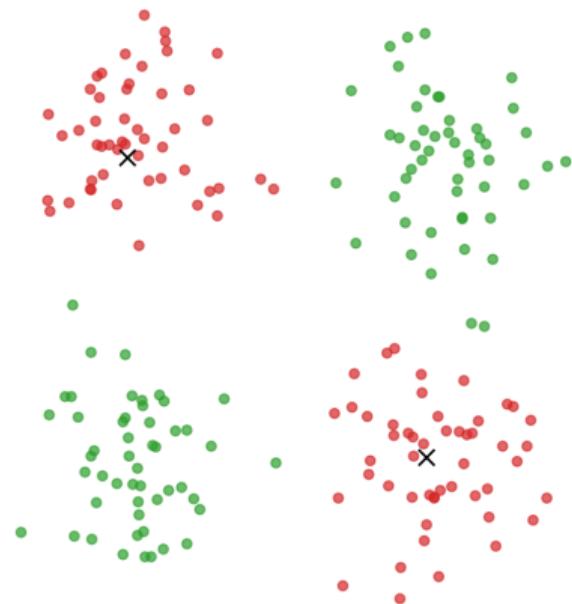
- ▶ For any feature vector  $\vec{x} \in \mathbb{R}^d$ , map to vector  $\vec{\varphi}(\vec{x}) \in \mathbb{R}^{d'}$ .
  - ▶  $\varphi_1$ : “similarity” of  $\vec{x}$  to  $\vec{\mu}^{(1)}$
  - ▶  $\varphi_2$ : “similarity” of  $\vec{x}$  to  $\vec{\mu}^{(2)}$
  - ▶ ...
  - ▶  $\varphi_{d'}$ : “similarity” of  $\vec{x}$  to  $\vec{\mu}^{(d')}$
- ▶ Train linear classifier in this new representation.
  - ▶ E.g., by minimizing expected square loss.

## Exercise

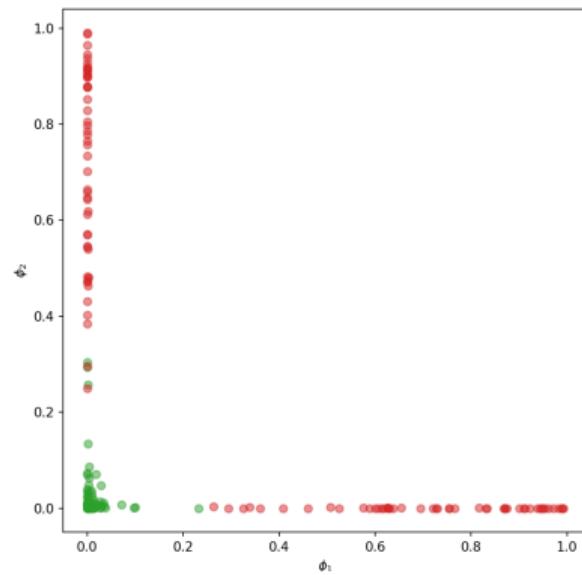
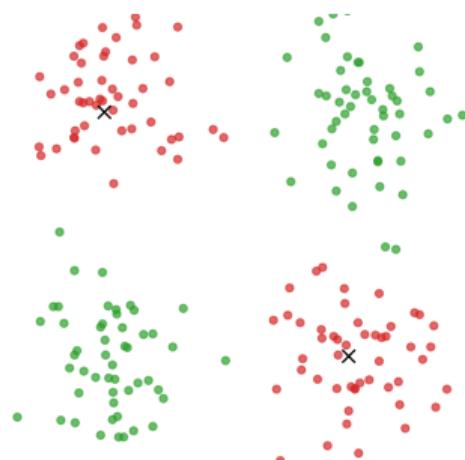
How many Gaussian basis functions would you use, and where would you place them to create a new representation for this data?



# Placement



# Feature Space



# Prediction Function

- ▶  $H(\vec{x})$  is a sum of Gaussians:

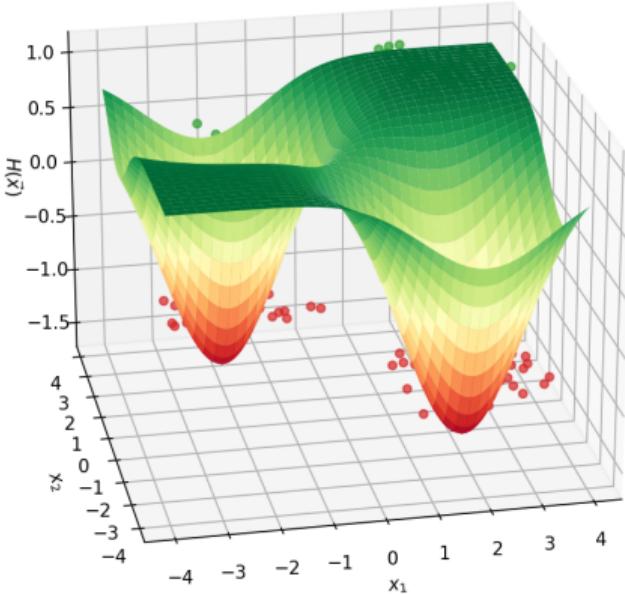
$$\begin{aligned} H(\vec{x}) &= w_0 + w_1 \varphi_1(\vec{x}) + w_2 \varphi_2(\vec{x}) + \dots \\ &= w_0 + w_1 e^{-\|\vec{x}-\vec{\mu}_1\|^2/\sigma^2} + w_2 e^{-\|\vec{x}-\vec{\mu}_2\|^2/\sigma^2} + \dots \end{aligned}$$

## Exercise

What does the surface of the prediction function look like?

Hint: what does the sum of 1-d Gaussians look like?

# Prediction Function Surface

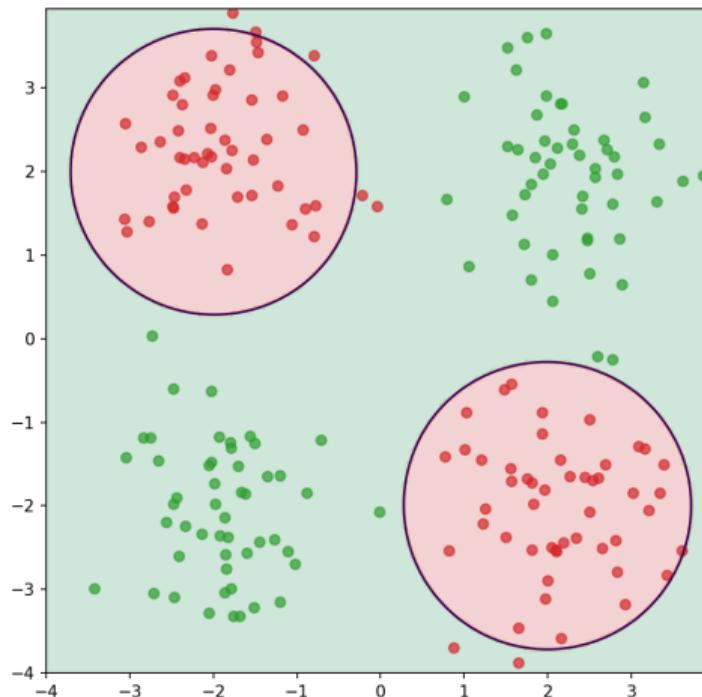


$$H(\vec{x}) = w_0 + w_1 e^{-\|\vec{x} - \vec{\mu}_1\|^2 / \sigma^2} + w_2 e^{-\|\vec{x} - \vec{\mu}_2\|^2 / \sigma^2}$$

# An Interpretation

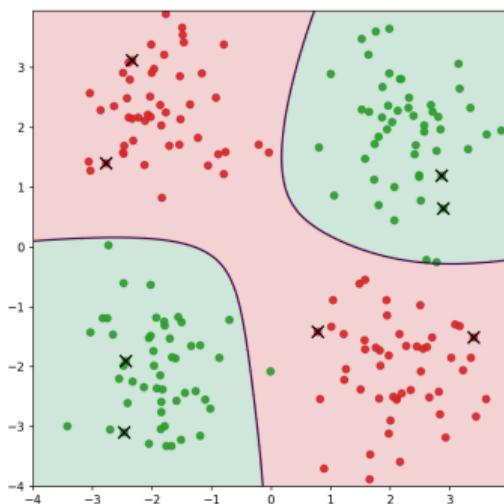
- ▶ Basis function  $\varphi_i$  makes a “bump” in surface of  $H$
- ▶  $w_i$  adjusts the “prominance” of this bump

# Decision Boundary

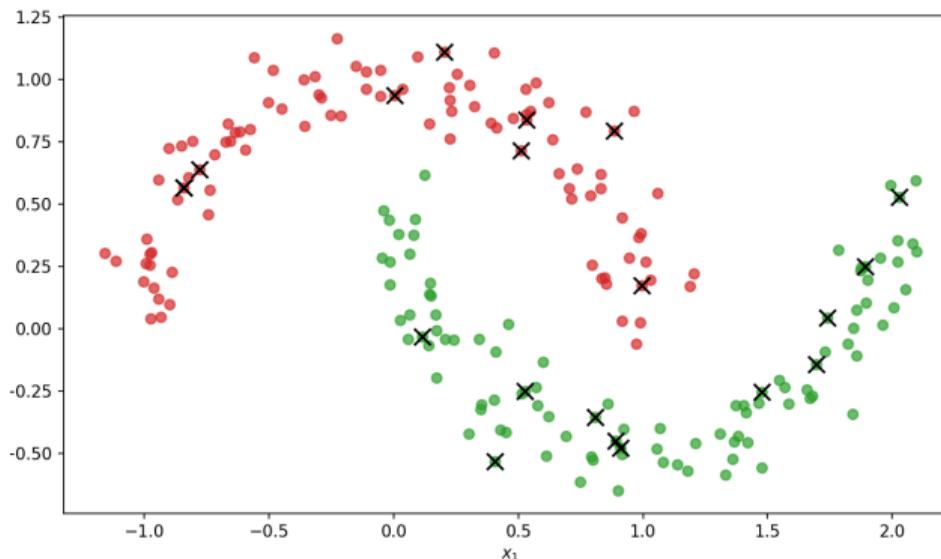


# More Features

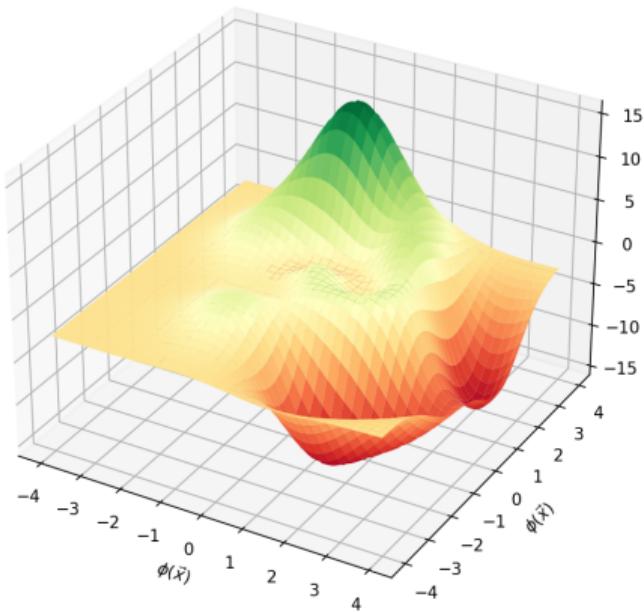
- ▶ By increasing number of basis functions, we can make more complex decision surfaces.



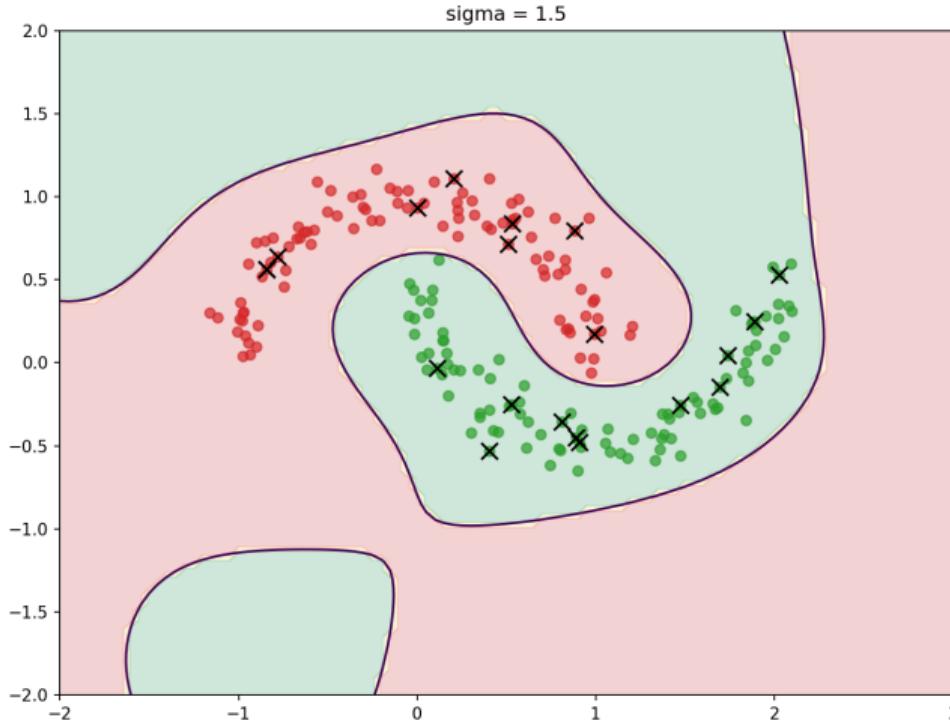
# Another Example



# Prediction Surface



# Decision Boundary



# Radial Basis Functions

- ▶ Gaussians are examples of **radial basis functions**.
- ▶ Each basis function has a **center**,  $\vec{c}$ .
- ▶ Value depends only on distance from center:

$$\varphi(\vec{x}; \vec{c}) = f(\|\vec{x} - \vec{c}\|)$$

# Another Radial Basis Function

- ▶ **Multiquadric:**  $\varphi(\vec{x}; \vec{c}) = \sqrt{\sigma^2 + \|\vec{x} - \vec{c}\|}/\sigma$

# DSC 190

## Machine Learning: Representations

Lecture 4 | Part 2

### Radial Basis Function Networks

# Recap

1. Choose basis functions,  $\varphi_1, \dots, \varphi_{d'}$
2. Transform data to new representation:

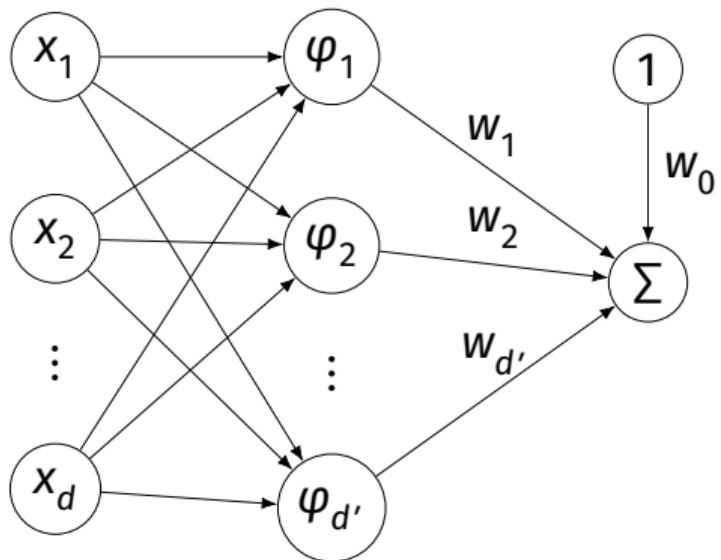
$$\vec{x} \mapsto (\varphi_1(\vec{x}), \varphi_2(\vec{x}), \dots, \varphi_{d'}(\vec{x}))^T$$

3. Train a linear classifier in this new space:

$$H(\vec{x}) = w_0 + w_1 \varphi_1(\vec{x}) + w_2 \varphi_2(\vec{x}) + \dots + w_{d'} \varphi_{d'}(\vec{x})$$

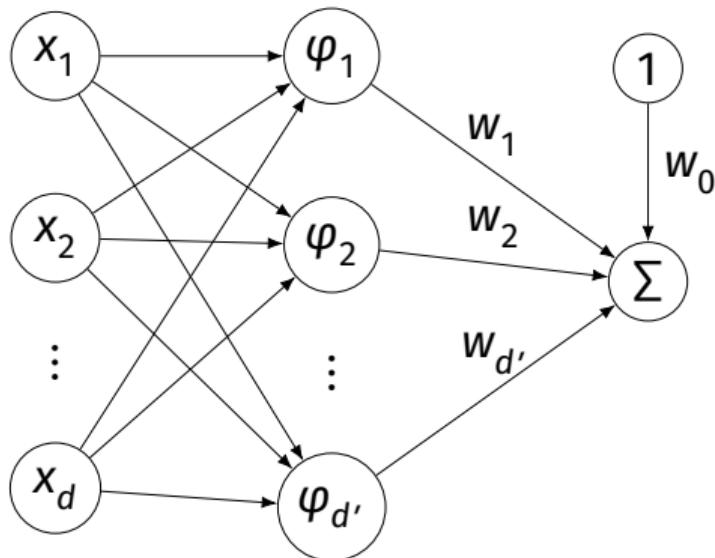
# The Model

- The  $\varphi$  are **basis functions**.



$$H(\vec{x}) = w_0 + w_1 \varphi_1(\vec{x}) + w_2 \varphi_2(\vec{x})$$

# Radial Basis Function Networks



- ▶ If the basis functions are **radial basis functions**, we call this a **radial basis function (RBF) network**.
- ▶ It is a simple type of neural network.

# Training

- ▶ An RBF network has these parameters:
  - ▶  $w_i$ : the weights associated to each “new” feature
  - ▶ the parameters of each individual basis function:
    - ▶  $\vec{\mu}_i$  (the center)
    - ▶ possibly others (e.g.,  $\sigma$ )
- ▶ How do we choose the parameters?

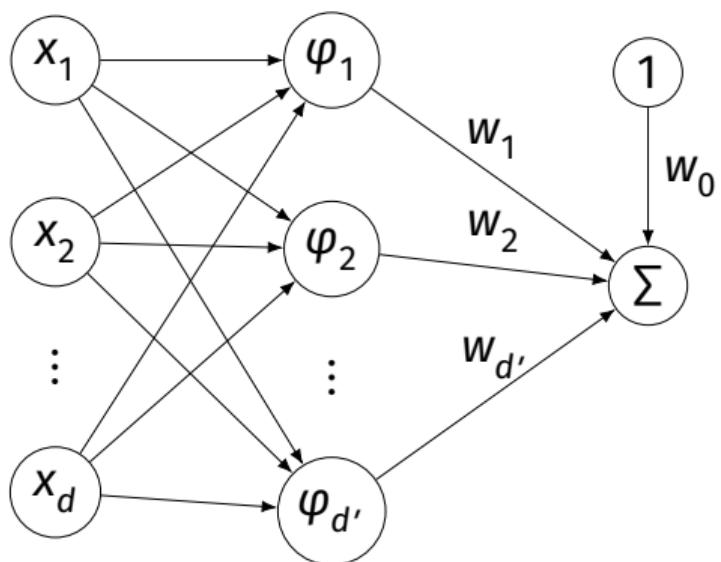
# Minimizing Expected Loss

- ▶ As with most any model, we can try to find parameters by minimizing expected loss.
- ▶ However, now the risk is a complex, non-linear function of many things:

$$R(\vec{w}, \vec{\mu}_1, \dots, \vec{\mu}_{d'}, \sigma, \dots).$$

- ▶ As opposed to a simple linear model:  $R(\vec{w})$ .

# Training



- ▶ Optimization is now much harder.
- ▶ Instead, we **decouple**:
  1. Find basis function parameters in some way, consider them fixed.
  2. Now train  $\vec{w}$  by minimizing risk

# Theory

- ▶ Given suitably-many basis functions, a Gaussian RBF is capable of approximating any continuous function arbitrarily well.

# DSC 190

## *Machine Learning: Representations*

Lecture 4 | Part 3

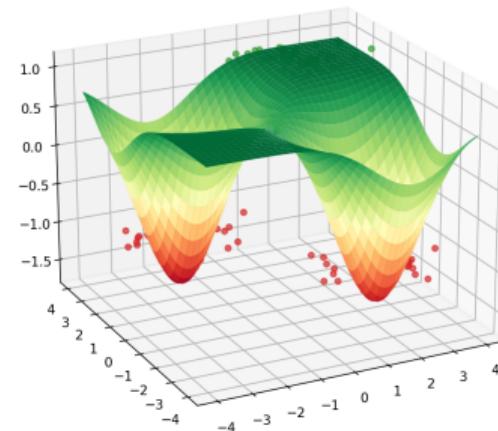
### Choosing RBF Locations

# Recap

- ▶ We map data to a new representation by first choosing **basis functions**.
- ▶ Radial Basis Functions (RBFs), such as Gaussians, are a popular choice.
- ▶ Requires choosing **center** for each basis function.

# Prediction Function

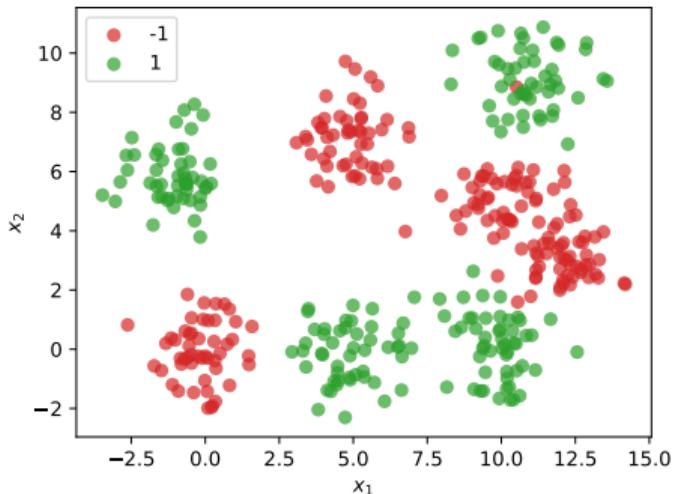
- ▶ Our prediction function  $H$  is a surface that is made up of Gaussian “bumps”.



$$H(\vec{x}) = w_0 + w_1 e^{-\|\vec{x} - \vec{\mu}_1\|^2 / \sigma^2} + w_2 e^{-\|\vec{x} - \vec{\mu}_2\|^2 / \sigma^2}$$

# Choosing Centers

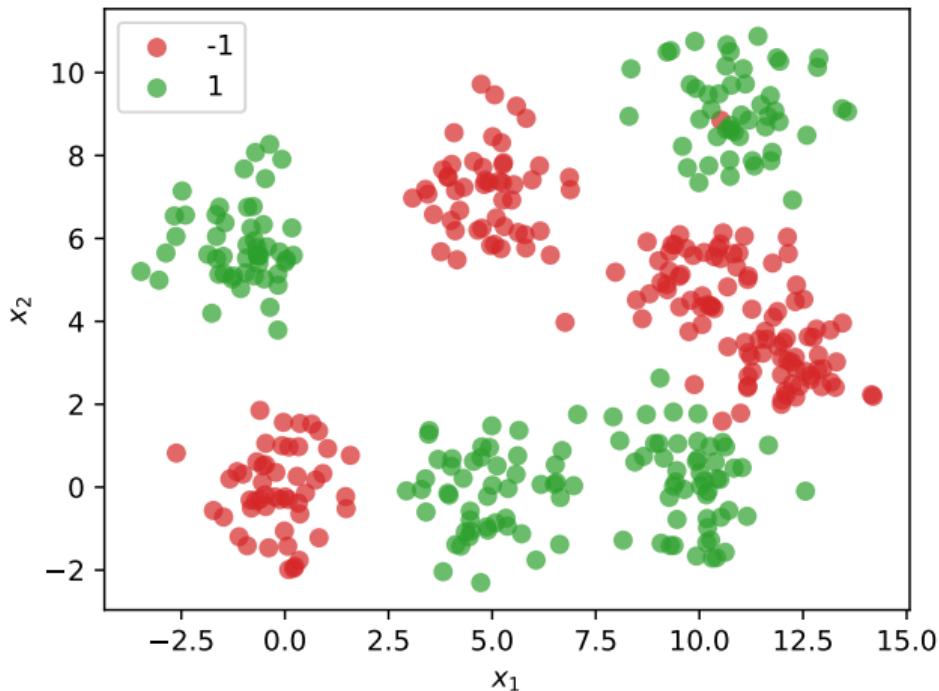
- ▶ Place the centers where the value of the prediction function should be controlled.
- ▶ Intuitively: place centers where the data is.



# Approaches

1. Every data point as a center
2. Randomly choose centers
3. Clustering

# Approach #1: Every Data Point as a Center



# Dimensionality

- ▶ We'll have  $n$  basis functions – one for each point.
- ▶ That means we'll have  $n$  features.
- ▶ Each feature vector  $\vec{\phi}(\vec{x}) \in \mathbb{R}^n$ .

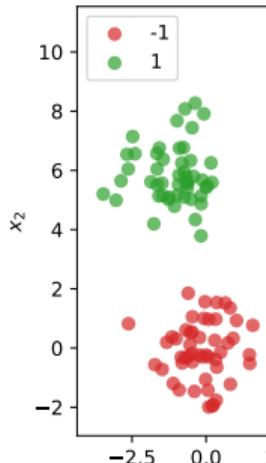
$$\vec{\phi}(\vec{x}) = (\phi_1(\vec{x}), \phi_2(\vec{x}), \dots, \phi_n(\vec{x}))^T$$

# Problems

- ▶ This causes problems.
- ▶ First: more likely to **overfit**.
- ▶ Second: computationally expensive<sup>a</sup>.

---

<sup>a</sup>However, this is very doable with SVMs

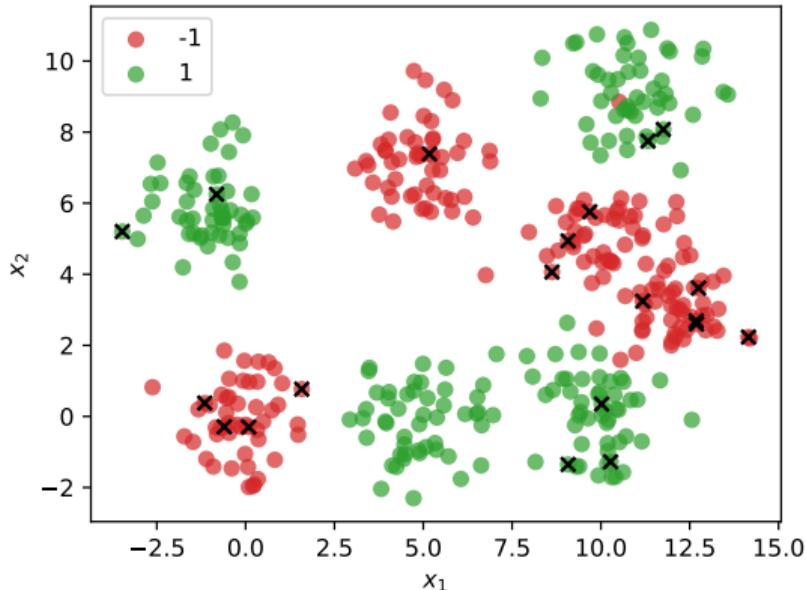


# Computational Cost

- ▶ Suppose feature matrix  $X$  is  $n \times d$ 
  - ▶  $n$  points in  $d$  dimensions
- ▶ Time complexity of solving  $X^T X \vec{w} = X^T \vec{y}$  is  $\Theta(nd^2)$
- ▶ Usually  $d \ll n$ . But if  $d = n$ , this is  $\Theta(n^3)$ .
- ▶ Not great! If  $n \approx 10,000$ , then takes > 10 minutes.

# Approach #2: A Random Sample

- Idea: randomly choose  $k$  data points as centers.



# **Problem**

- ▶ May undersample/oversample a region.
- ▶ More advanced sampling approaches exist.

## Approach #3: Clustering

- ▶ Group data points into **clusters**.
- ▶ Cluster centers are good places for RBFs.
- ▶ We'll use  $k$ -means clustering to pick  $k$  centers.

# DSC 190

## *Machine Learning: Representations*

Lecture 4 | Part 4

### Linear Algebra: Linear Transformations

# **And now for something completely different...**

- ▶ This and the next few lectures will end with linear algebra refreshers.

# Vectors

- ▶ A vector  $\vec{x}$  is an arrow from the origin to a point.
- ▶ We can make new arrows by:
  - ▶ scaling:  $\alpha\vec{x}$
  - ▶ addition:  $\vec{x} + \vec{y}$
  - ▶ both:  $\alpha\vec{x} + \beta\vec{y}$
- ▶  $\|\vec{x}\|$  is the **norm** (or length) of  $\vec{x}$

# Linear Combinations

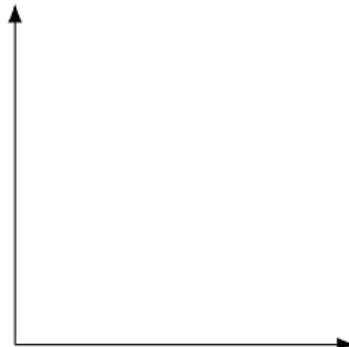
- ▶ We can add together a bunch of arrows:

$$\vec{y} = \alpha_1 \vec{x}^{(1)} + \alpha_2 \vec{x}^{(2)} + \dots + \alpha_n \vec{x}^{(n)}$$

- ▶ This is a **linear combination** of  $\vec{x}^{(1)}, \dots, \vec{x}^{(n)}$

# Decompositions

- ▶ Consider the two vectors,  $\vec{u}^{(1)}$  and  $\vec{u}^{(2)}$ .
- ▶ Claim: any vector  $\vec{x}$  in  $\mathbb{R}^2$  can be **decomposed** (written) as
$$\vec{x} = \alpha\vec{u}^{(1)} + \beta\vec{u}^{(2)}$$
- ▶  $\vec{u}^{(1)}$  and  $\vec{u}^{(2)}$  form a **basis** of  $\mathbb{R}^2$

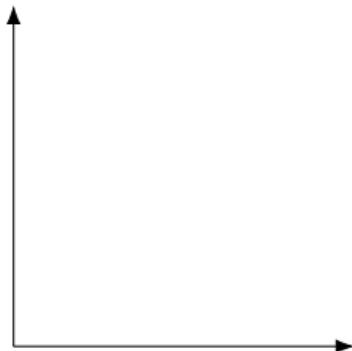


# Bases

- ▶ There was nothing special about  $\vec{u}^{(1)}$  and  $\vec{u}^{(2)}$ .
- ▶ There are **infinitely many** bases of  $\mathbb{R}^2$ .
- ▶ But there is one that is particularly natural...

# Standard Basis Vectors

- ▶  $\hat{e}^{(1)}$  and  $\hat{e}^{(2)}$  are the **standard basis vectors** in  $\mathbb{R}^2$ .
- ▶ We write  $\vec{x} = \alpha\hat{e}^{(1)} + \beta\hat{e}^{(2)}$



# Coordinate Vectors

- We often write a vector  $\vec{x}$  as a **coordinate vector**:

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix}$$

- Meaning:  $\vec{x} = x_1 \hat{e}^{(1)} + x_2 \hat{e}^{(2)} + \dots + x_d \hat{e}^{(d)}$

# Functions of a Vector

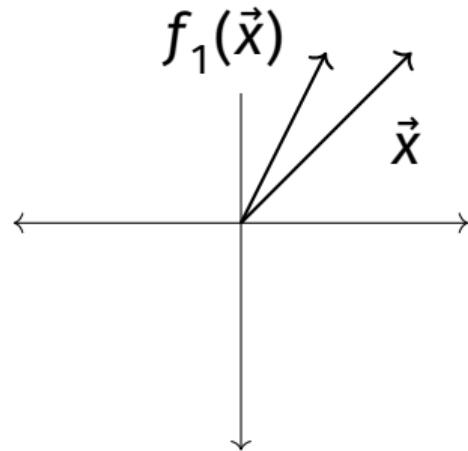
- ▶ In ML, we often work with functions of a vector:  
 $f : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ .
- ▶ Example: a prediction function,  $H(\vec{x})$ .
- ▶ Functions of a vector can return:
  - ▶ a number:  $f : \mathbb{R}^d \rightarrow \mathbb{R}^1$
  - ▶ a vector  $f : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$
  - ▶ something else?

# Transformations

- ▶ A **transformation**  $f$  is a function that takes in a vector, and returns a vector *of the same dimensionality*.
- ▶ That is,  $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ .

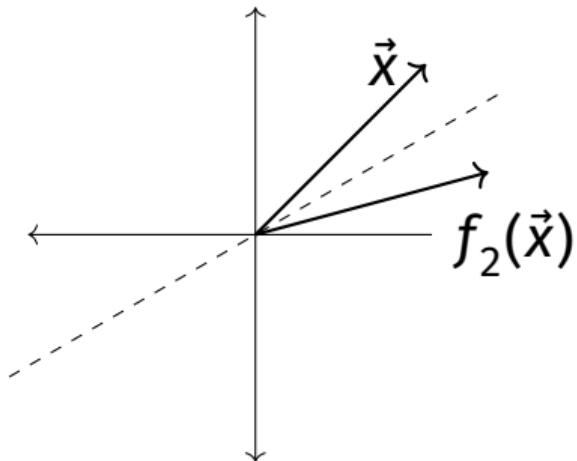
# Example

- ▶  $f_1(\vec{x})$  halves horizontal component.



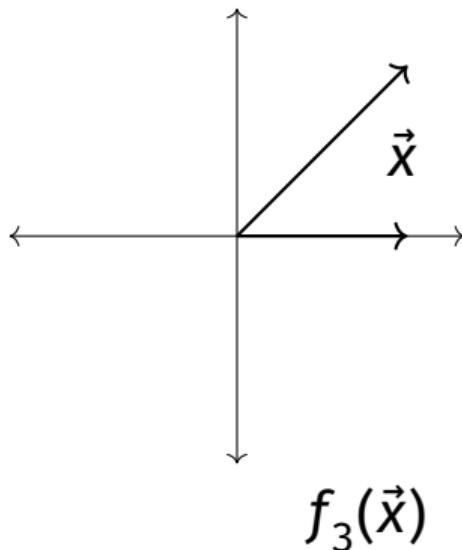
# Example

- ▶  $f_2(\vec{x})$  flips  $\vec{x}$  over the dashed line.



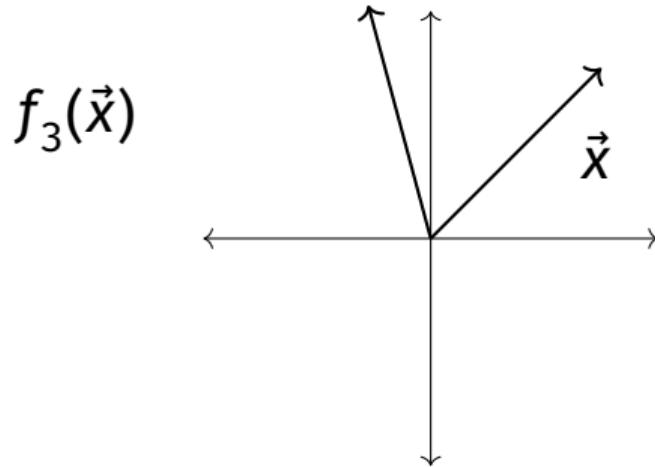
# Example

- ▶  $f_3(\vec{x})$  projects  $\vec{x}$  onto the horizontal axis.



# Example

- ▶  $f_4(\vec{x})$  rotates  $\vec{x}$  by  $45^\circ$  anticlockwise.



# Linear Transformations

- ▶ An arbitrary transformation can be quite complex.
- ▶ For mathematical ease, we may decide to consider only **linear transformations**.
- ▶ A transformation  $f$  is linear if:

$$f(\alpha \vec{x} + \beta \vec{y}) = \alpha f(\vec{x}) + \beta f(\vec{y})$$

## By the way...

- ▶ “Linear” functions,  $f(x) = mx + b$ , aren’t linear in this sense (unless  $b = 0$ ).
- ▶ Rather call these “affine” functions.

# Examples

- ▶ All of the previous four transformations are linear.
- ▶ Another example:  $f(\vec{x}) = (x_1 + x_2, x_1 - x_2)^T$
- ▶ Non-example:  $f$  scales the input by the square of its length.

## Main Idea

We use linear functions (and linear transformations) because they are simple and easy to work with mathematically.

# The Simplicity of Linear Transformations

- ▶ Suppose  $f$  is an **arbitrary** transformation.
- ▶ I tell you  $f(\hat{e}^{(1)}) = (2, 1)^T$  and  $f(\hat{e}^{(2)}) = (-3, 0)^T$ .
- ▶ I tell you  $\vec{x} = (x_1, x_2)^T$ .
- ▶ What is  $f(\vec{x})$ ?

# The Simplicity of Linear Transformations

- ▶ Suppose  $f$  is a **linear** transformation.
- ▶ I tell you  $f(\hat{e}^{(1)}) = (2, 1)^T$  and  $f(\hat{e}^{(2)}) = (-3, 0)^T$ .
- ▶ I tell you  $\vec{x} = (x_1, x_2)^T$ .
- ▶ What is  $f(\vec{x})$ ?

## Exercise

- ▶ Suppose  $f$  is a **linear** transformation.
- ▶ I tell you  $f(\hat{e}^{(1)}) = (2, 1)^T$  and  $f(\hat{e}^{(2)}) = (-3, 0)^T$ .
- ▶ I tell you  $\vec{x} = (3, -4)^T$ .
- ▶ What is  $f(\vec{x})$ ?

# Key Fact

- ▶ Linear functions are determined **entirely** by what they do on the basis vectors.
- ▶ I.e., to tell you what  $f$  does, I only need to tell you  $f(\hat{e}^{(1)})$  and  $f(\hat{e}^{(2)})$ .
- ▶ This makes the math easy!