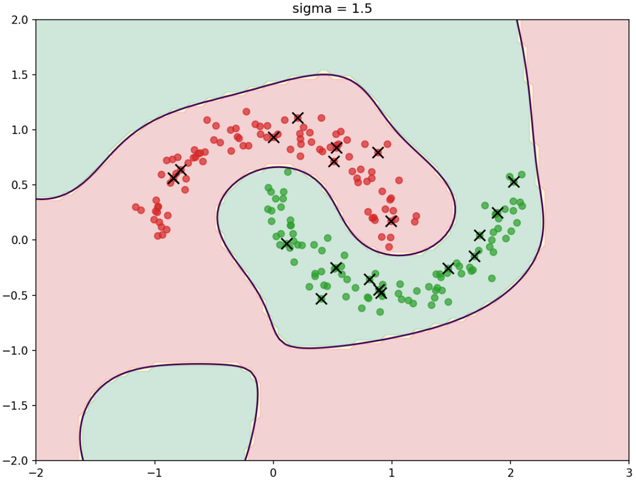
DSC 190

Machine Learning: Representations

Lecture 4 | Part 1

Radial Basis Functions

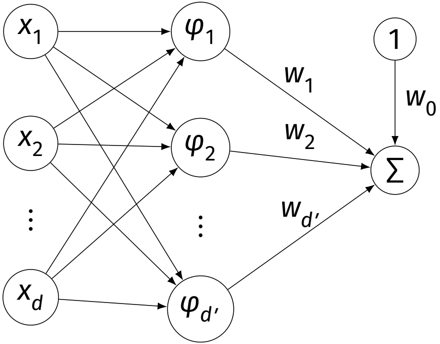
Today



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, $\varphi_1, \varphi_2, \ldots, \varphi_{d'}$

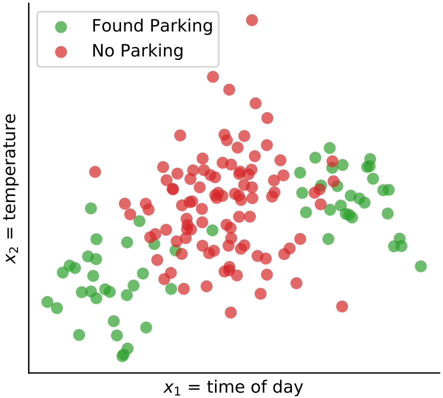
3. Map each data point to **feature space** $\mathbb{R}^{d'}$:

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

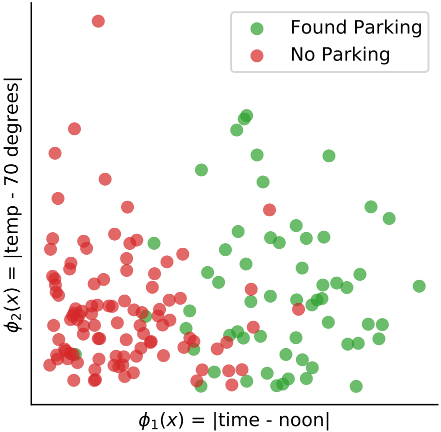
4. Fit linear prediction function in new space:

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

Last Time

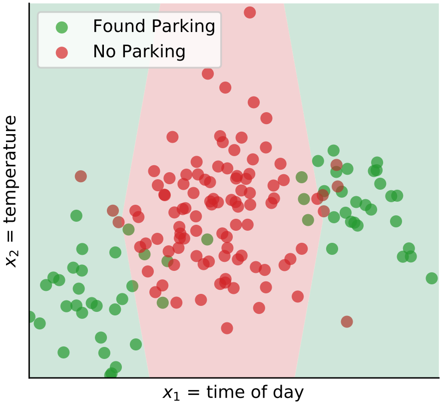


Found Parking
No Parking
x_2 = temperature
x_1 = time of day

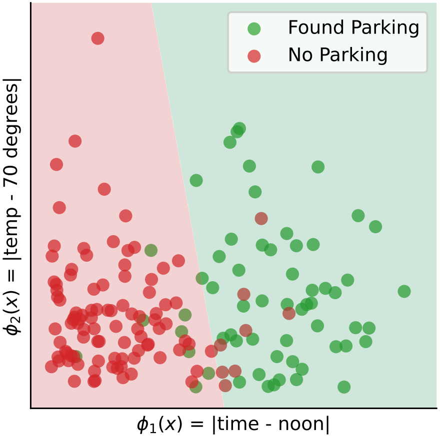


Found Parking
No Parking
$\phi_2(x)$ = |temp - 70 degrees|
$\phi_1(x)$ = |time - noon|

Last Time

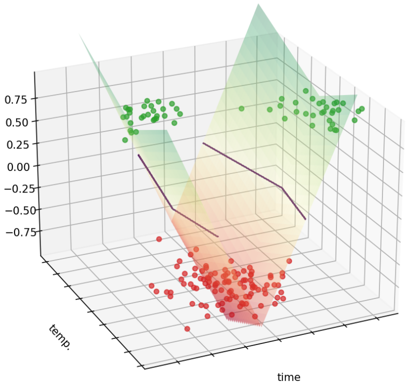


Found Parking
No Parking
x_2 = temperature
x_1 = time of day

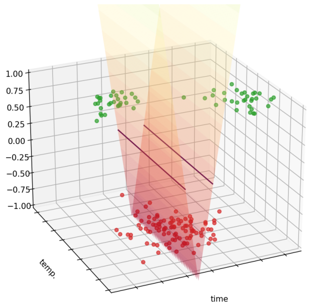


Found Parking
No Parking
$\phi_2(x)$ = |temp - 70 degrees|
$\phi_1(x)$ = |time - noon|

Visualizing the "Prediction Surface"

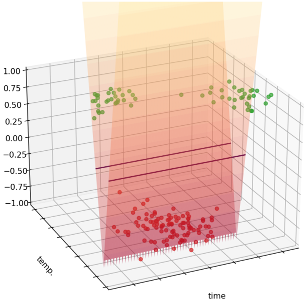


Visualizing the Basis Function φ_1



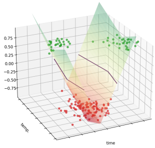
- $w_0 + w_1 |x_1 - \text{noon}|$

Visualizing the Basis Function φ_2

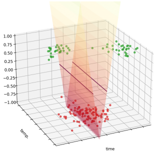


- $w_0 + w_2 |x_2 - 72^\circ|$

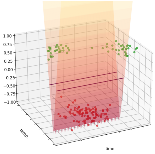
Visualizing the “Prediction Surface”



=



+

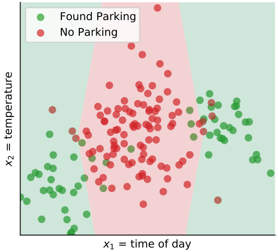


The Decision Boundary

- 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?



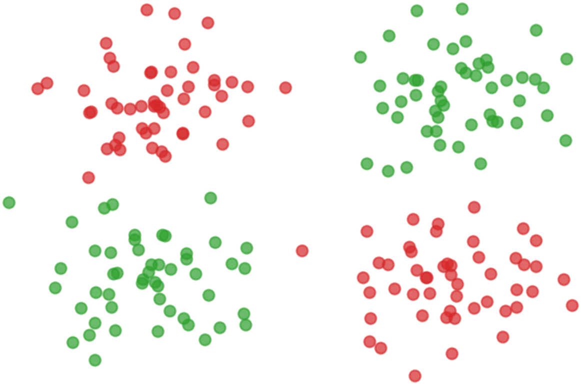
Answer: No!

- 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.

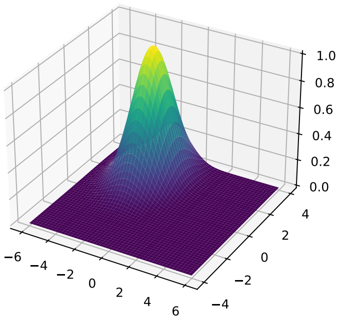
Choosing Basis Functions

- Our previous basis functions have limitations.
- They are convex: prediction surface can only have one negative/positive region.
- They diverge $\to \infty$ 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.
- σ 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: φ 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 $\vec{\mu}^{(1)}, \ldots, \vec{\mu}^{(2)}, ..., \vec{\mu}^{(d')}$
- Pick widths: $\sigma_1, \sigma_2, \ldots, \sigma_{d'}$ (usually all the same)
- Define ith basis function:

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

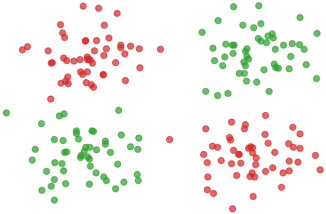
New Representation

- For any feature vector $\vec{x} \in \mathbb{R}^d$, map to vector $\vec{\varphi}(\vec{x}) \in \mathbb{R}^{d'}$.
  - φ_1: "similarity" of $\vec{x}$ to $\vec{\mu}^{(1)}$
  - φ_2: "similarity" of $\vec{x}$ to $\vec{\mu}^{(2)}$
  - ...
  - $\varphi_{d'}$: "similarity" of $\vec{x}$ to $\vec{\mu}^{(d')}$

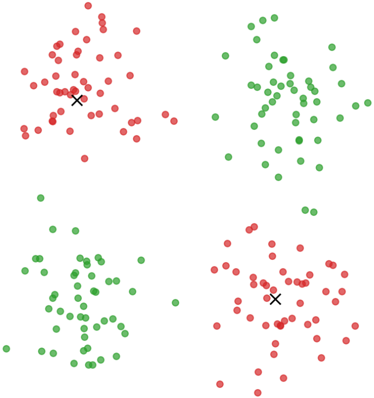
- Train linear classifier in this new representation.
  - E.g., by minimizing expected square loss.

Exercise

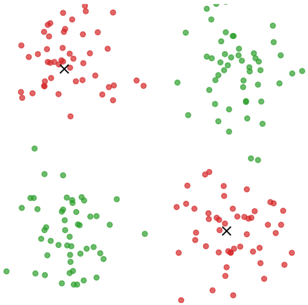
How many Gaussian basis functions would you use, and where would you place them to create a new representation for this data?

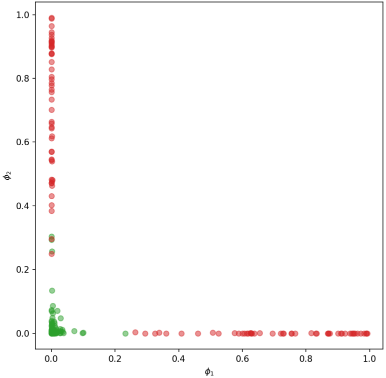


Placement



Feature Space





Prediction Function

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

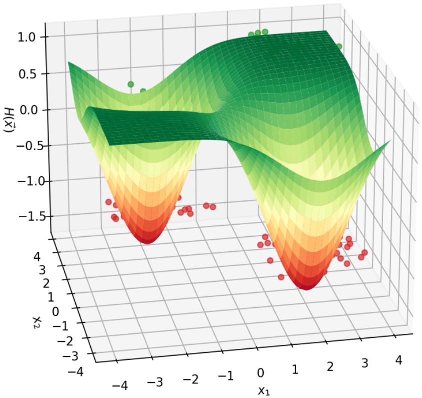
$$\begin{aligned} H(\vec{x}) &= w_0 + w_1 \varphi_1(\vec{x}) + w_2 \varphi_2(\vec{x}) + \ldots \\ &= w_0 + w_1 e^{-\|\vec{x}-\vec{\mu}_1\|^2/\sigma^2} + w_2 e^{-\|\vec{x}-\vec{\mu}_2\|^2/\sigma^2} + \ldots \end{aligned}$$

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

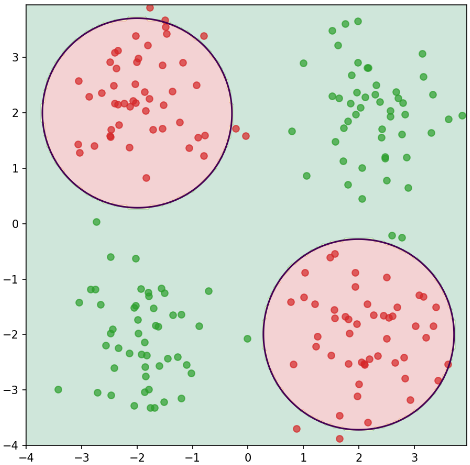


$$H(\vec{x}) = w_0 + w_1 e^{-\|\vec{x}-\vec{\mu}_1\|^2/\sigma^2} + w_2 e^{-\|\vec{x}-\vec{\mu}_2\|^2/\sigma^2}$$

An Interpretation

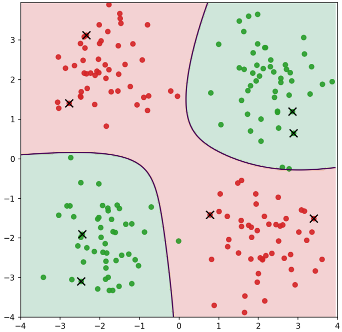
- Basis function φ_i makes a "bump" in surface of H
- w_i adjusts the "prominance" of this bump

Decision Boundary

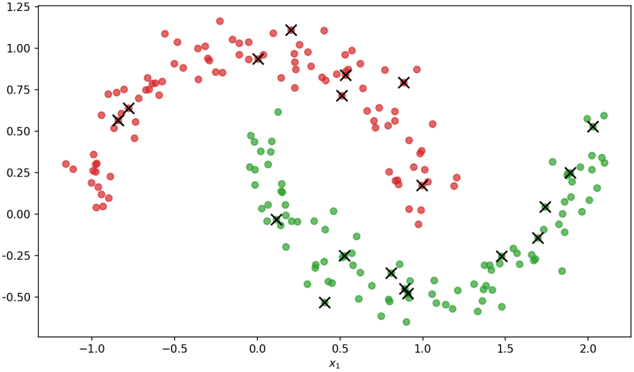


More Features

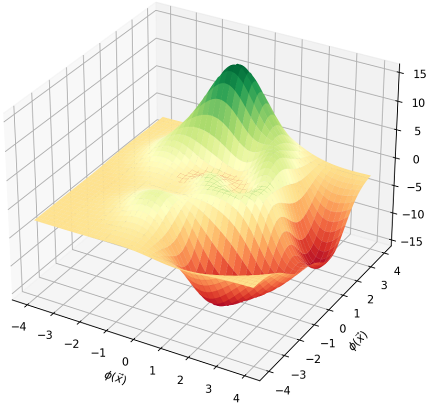
- By increasing number of basis functions, we can make more complex decision surfaces.



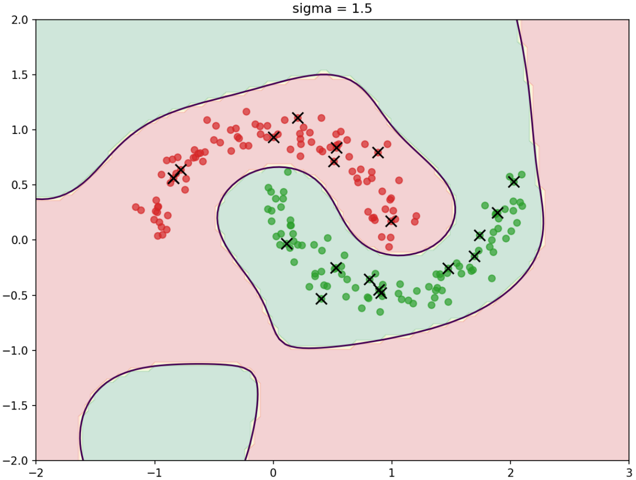
Another Example



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, $\varphi_1, \ldots, \varphi_{d'}$

2. Transform data to new representation:

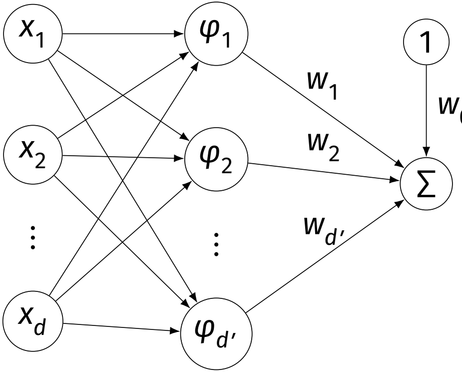
$$\vec{x} \mapsto (\varphi_1(\vec{x}), \varphi_2(\vec{x}), \ldots, \varphi_{d'}(\vec{x}))^T$$

3. Train a linear classifier in this new space:

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

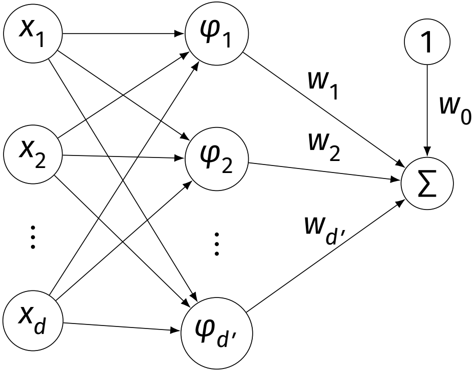
The Model

- The φ 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., σ)
- How do we choose the parameters?

Minimizing Expected Loss

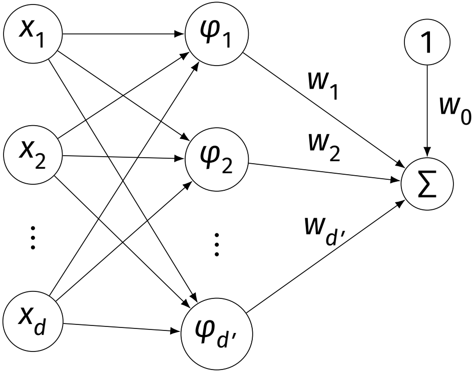
- As with most any model, we can try to find parameters by minimizing expected loss.

- However, now the risk is a complex, non-linear function of many things:

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

- 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

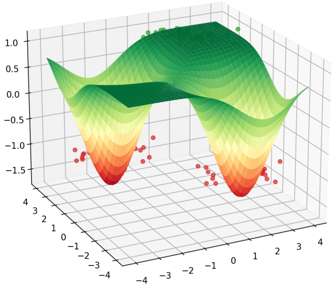
Choosing RBF Locations

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

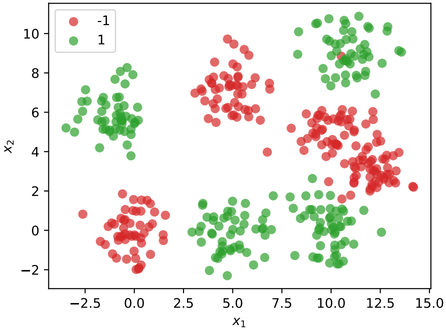
- Our prediction function H is a surface that is made up of Gaussian “bumps”.



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

Choosing Centers

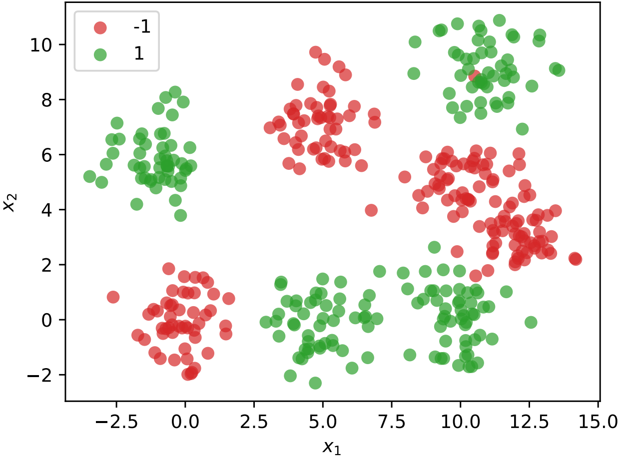
- Place the centers where the value of the prediction function should be controlled.
- Intuitively: place centers where the data is.



Approaches

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

Approach #1: Every Data Point as a Center



Dimensionality

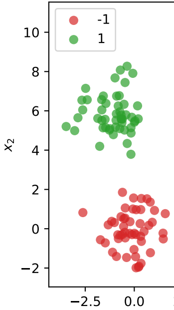
- We'll have n basis functions – one for each point.
- That means we'll have n features.
- Each feature vector $\vec{\phi}(\vec{x}) \in \mathbb{R}^n$.

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

Problems

- This causes problems.
- First: more likely to **overfit**.
- Second: computationally expensive[a].

[a] However, this is very doable with SVMs

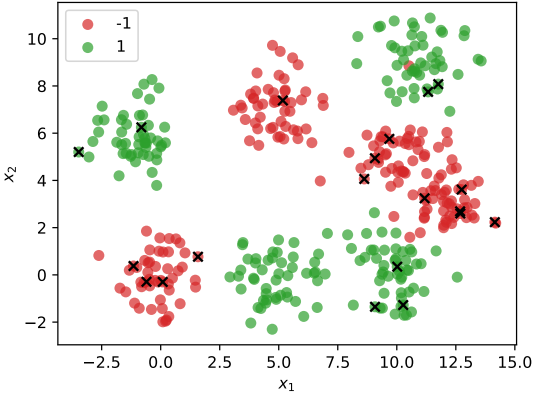


Computational Cost

- 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.