DS-GA 1003: Machine Learning

Lecture 5: Features and Kernels

Slides adapted from material from David Rosenberg.

Logistics & Announcements

PS 2 due tonight (11:59 pm) & PS 3 released (due Mar. 24 11:59pm). 1 month for PS 3.

Midterm review. Poll currently on Ed for Zoom midterm review; lab next week also review.

Midterm page on website has details about midterm + practice sheet + what to study.

If you can do the practice sheet only with 8.5x11 cheatsheet and ace it, you should be able to ace the midterm!

Group formation (due Feb. 28 11:59pm). Submit your groups on Gradescope, use Ed thread to find group members.

Let us know ASAP if you are having trouble finding a group.

Outline

Input Space $\mathscr{X}$

Going beyond $\mathscr{X} = \mathbb{R}^d$

So far, $\mathscr{X} = \mathbb{R}^d$ for the methods we've developed:

Ridge regression/Lasso regression

Logistic Regression

Support vector machines

Our hypothesis space for these was all affine functions on $\mathbb{R}^d$:

$$\mathscr{H} = \{x \mapsto w^\top x + b : w \in \mathbb{R}^d, b \in \mathbb{R}\}.$$

What if we want to do prediction on inputs not natively in $\mathbb{R}^d$?

Input Space $\mathscr{X}$

Going beyond $\mathscr{X} = \mathbb{R}^d$

What if we want to do prediction on inputs not natively in $\mathbb{R}^d$?

Examples: Text documents, Image files, Sound recordings, DNA sequences…

But we need to encode data onto a computer (everything is a sequence of numbers):

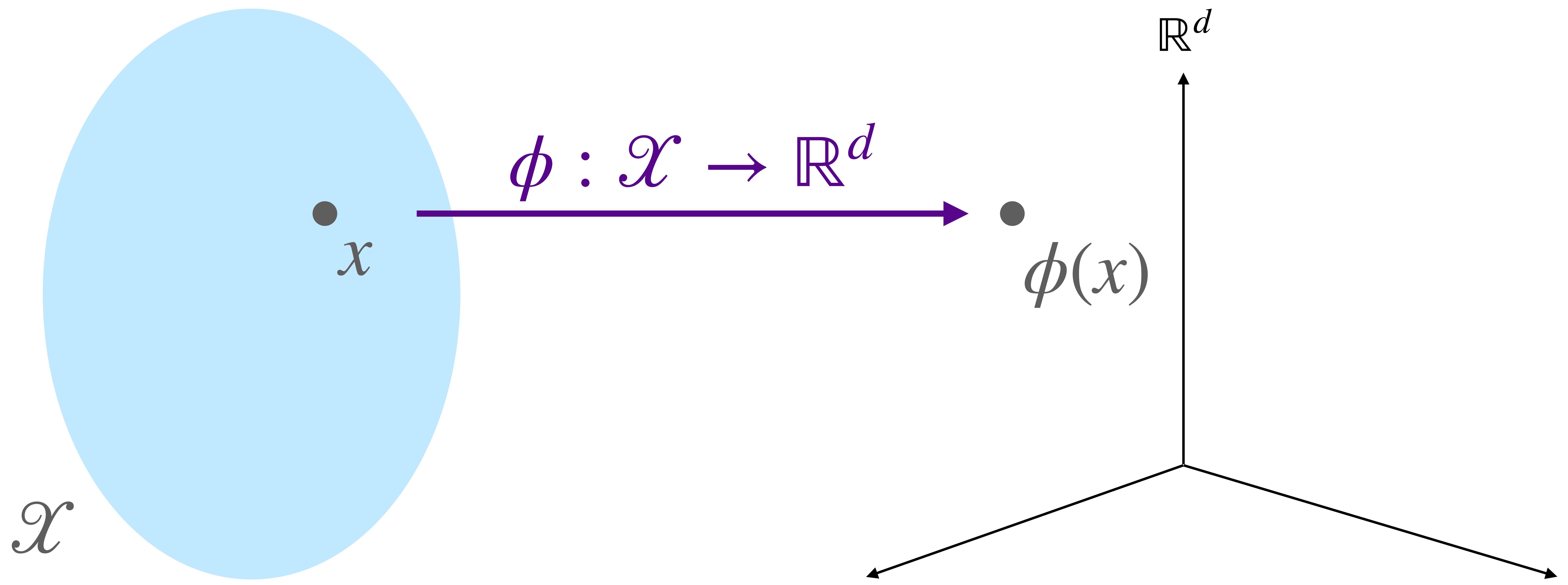
Entry i of each sequence should have same "meaning."

All sequences should have same length.

Feature Extraction

Mapping back to $\mathbb{R}^d$

Mapping an input from $\mathscr{X}$ to a vector in $\mathbb{R}^d$ is called feature extraction or featurization.



Feature Extraction

Terminology

Input space: $\mathcal{X}$ (no assumptions)

Introduce a feature map $\phi : \mathcal{X} \to \mathbb{R}^d$ that maps into feature space $\mathbb{R}^d$.

Hypothesis space of affine functions on the feature space:

$$\mathcal{H} = \{x \mapsto w^\top \phi(x) + b : w \in \mathbb{R}^d, b \in \mathbb{R}\}$$

Geometric Example

Changing feature space

Binary classification setting in $\mathbb{R}^2$.

$$\mathcal{X} = \mathbb{R}^2, \mathcal{Y} = \{-1,1\}$$

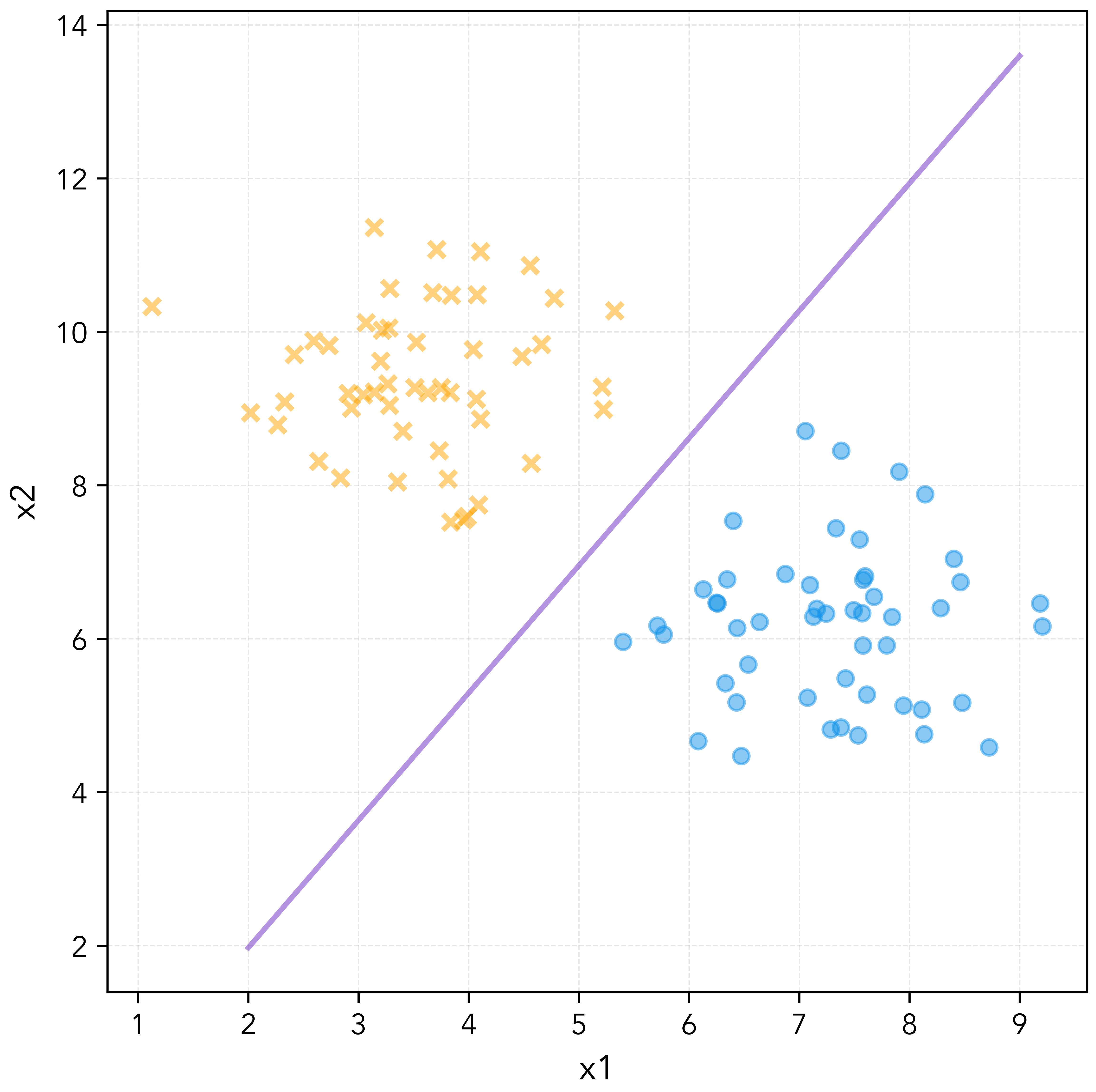
Goal: Find a linear function

$$f_{w,b}(x) = w^\top x + b$$

such that we can separate the points:

$$f_{w,b}(x) > 0 \text{ for } x \text{ such that } y = 1.$$

$$f_{w,b}(x) < 0 \text{ for } x \text{ such that } y = -1.$$

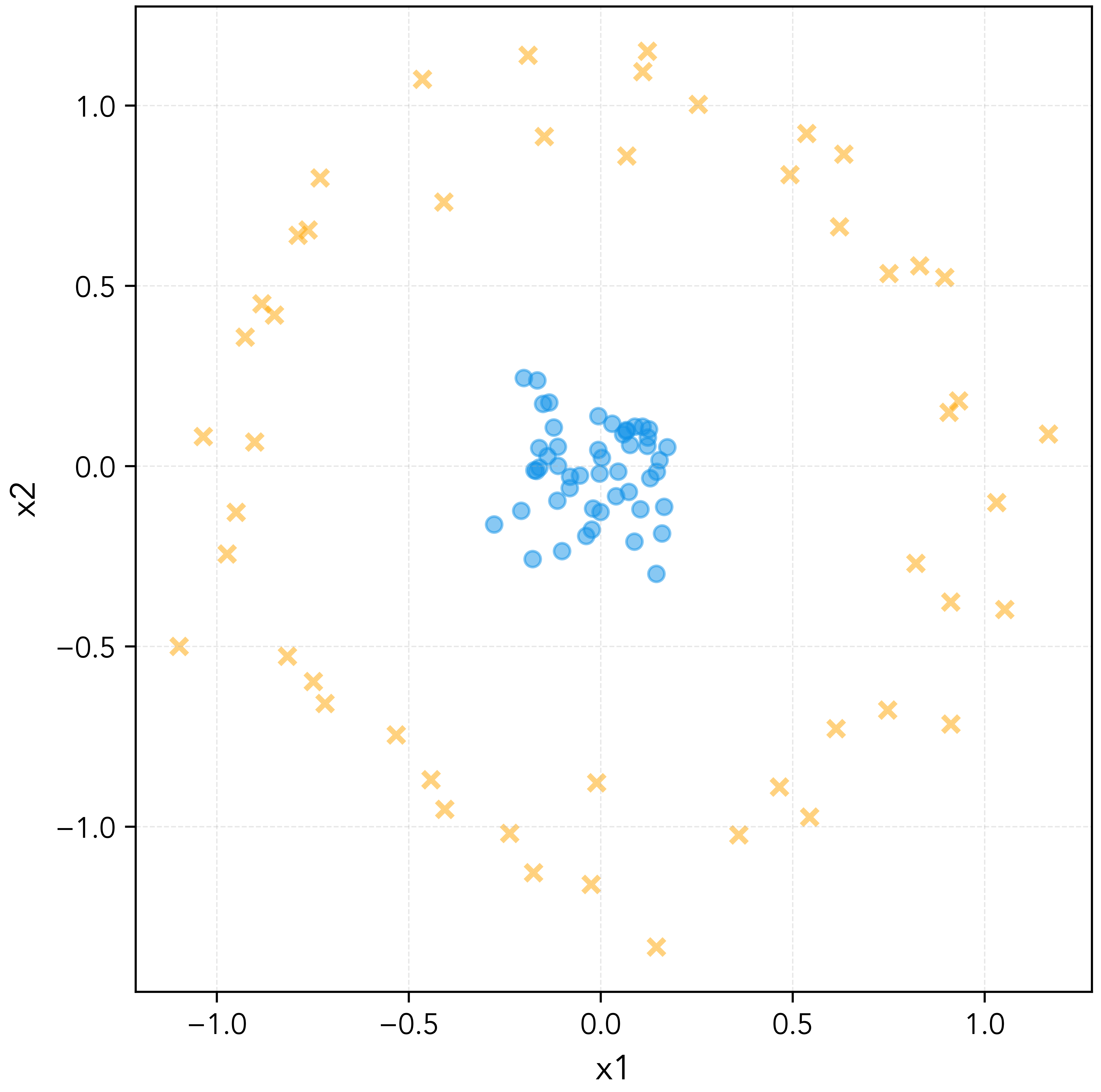


Geometric Example

Changing feature space

$$f_{w,b}(x) = w^\top x + b$$

What if the data are *not* linearly separable?



Geometric Example

Changing feature space

$$f_w(x) = w^\top x$$

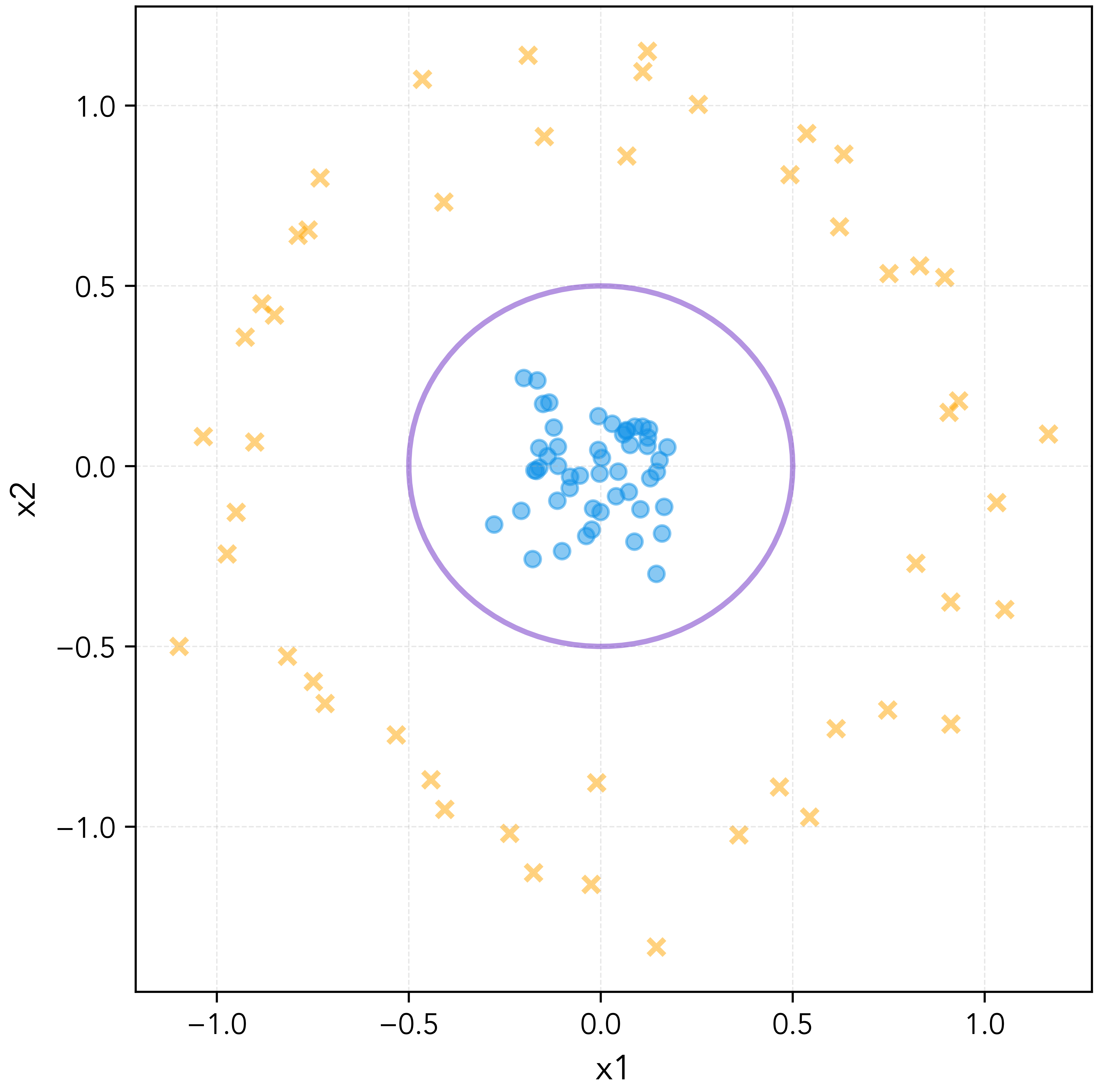
What if the data are *not* linearly separable?

Solution: $\phi : \mathbb{R}^2 \to \mathbb{R}^5$ (a feature map)

$$(x_1, x_2) \mapsto (x_1, x_2, x_1^2, x_2^2, x_1 x_2, 1)$$

$$w^* = (0,0,1,1,0,-r)$$

$$\phi(x)^\top w^* = x_1^2 + x_2^2 - r$$



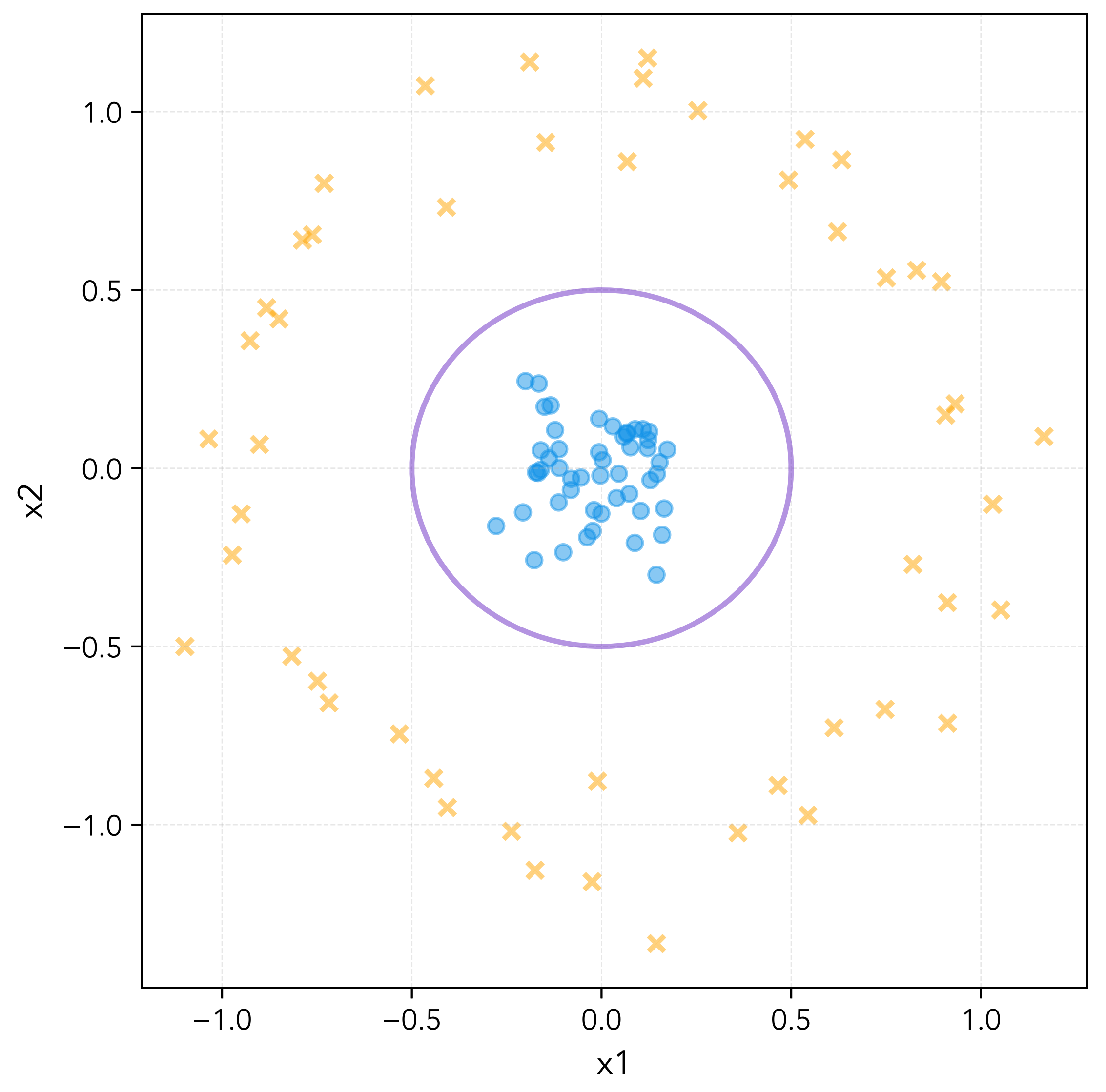
Geometric Example

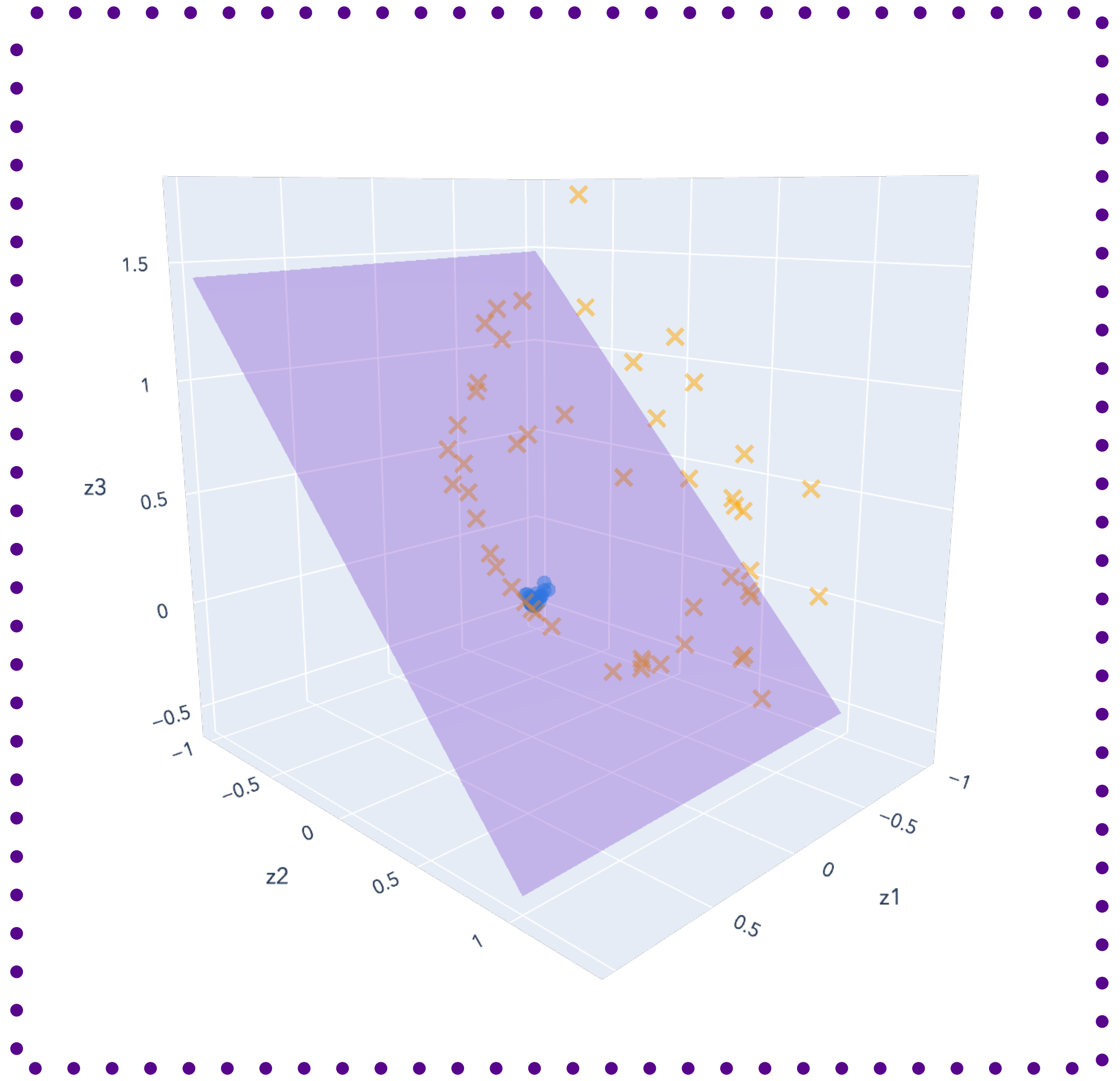
Changing feature space

Data not linearly separable in lower-dim space might be separable in higher dimensions!

$$\phi : \mathbb{R}^2 \to \mathbb{R}^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$





Geometric Example

Changing feature space

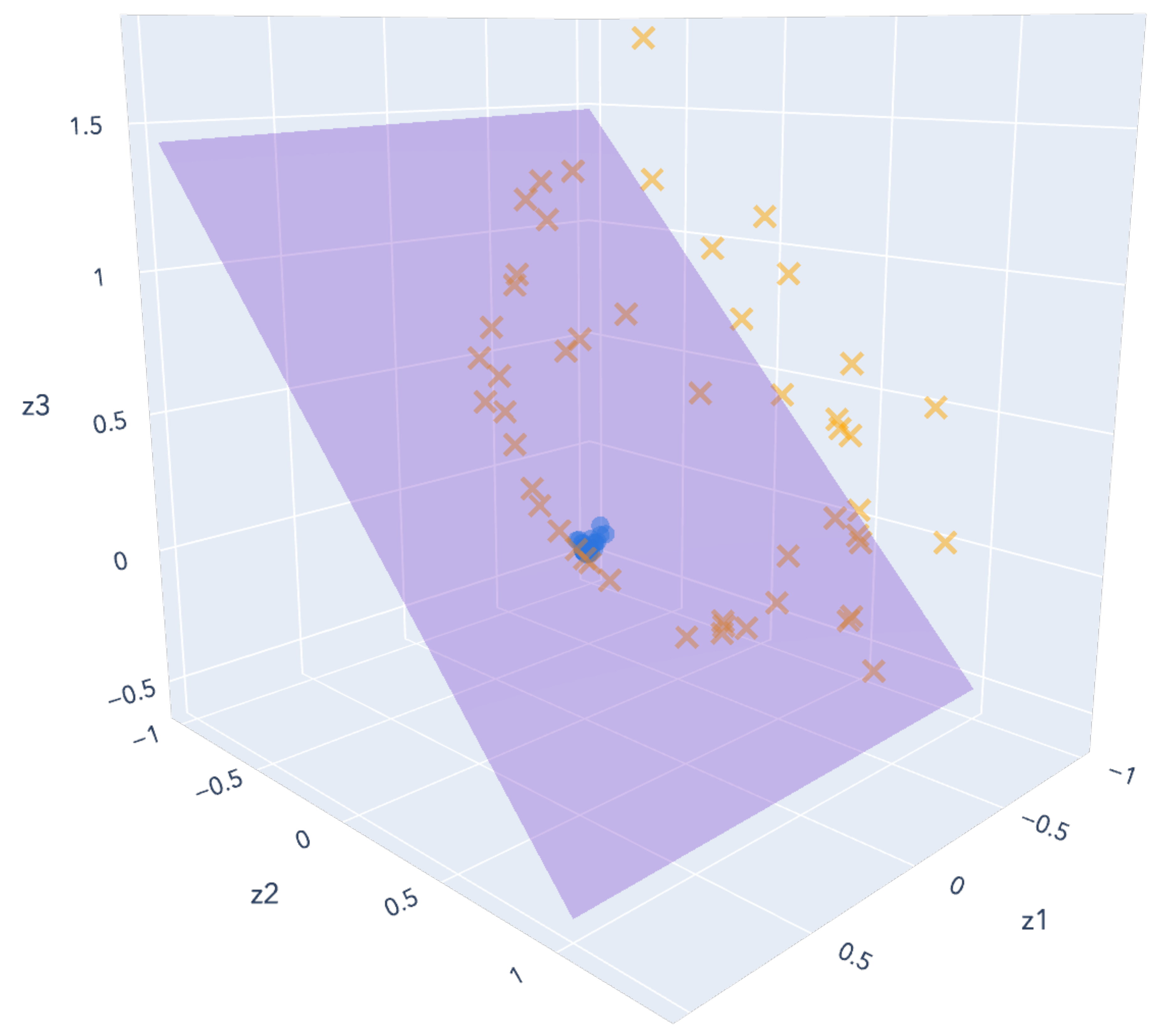
Goal: Find feature transformation ϕ that allows linear separability in higher dimensions.

Then, fit a linear classifier using linear methods we already know:

SVM

Logistic Regression

Later: neural networks are *automatic* feature extractors!



Linear Hypothesis Spaces

Expressivity

$$\mathscr{H} = \{x \mapsto w^\top \phi(x) : w \in \mathbb{R}^d\}$$

For linear models, to grow the hypothesis space, we must add features.

Sometimes we say that a larger hypothesis space is **more expressive**.

Can fit more relationships between inputs and action.

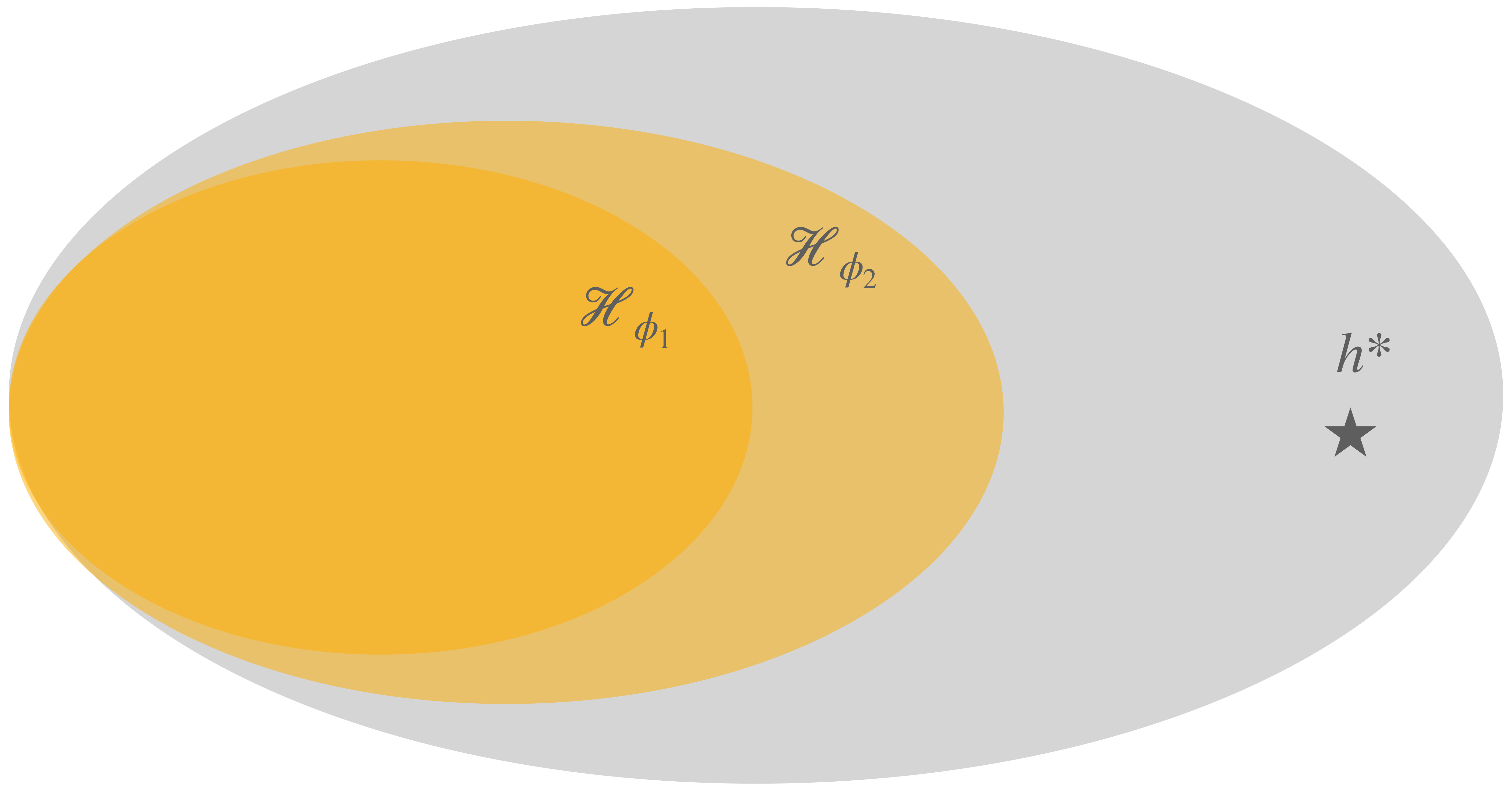
So, by choosing a good $\phi(\,\cdot\,)$, we can “expand” our hypothesis space.

Linear Hypothesis Spaces

Expressivity

$$\mathscr{H} = \{x \mapsto w^\top \phi(x) : w \in \mathbb{R}^d\}$$

Choosing a "more expressive" feature map gives us more flexibility in functions of x.



Outline

Feature Extraction

Example: Predicting Health

General philosophy: extract every feature that might be relevant.

Feature for medical diagnosis:

Height, weight, body temperature, blood pressure, etc. …

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Feature Extraction

Issues for linear predictors

Possible issues (for linear predictors):

The relationship between feature and label may not be linear.

There may be complex dependencies between features.

Three types of nonlinearities can cause problems:

1. Non-monotonicity.
2. Saturation.
3. Interactions between features.

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Non-Monotonicity

Issue

Feature map: $\phi(x) = (1, \text{temperature}(x))$

Action: Predict a "health score" $y \in \mathbb{R}$ (positive is good).

Hypothesis space:

$$\mathscr{H} = \{x \mapsto w^\top \phi(x) : w \in \mathbb{R}^2\} = \{\text{affine functions of temperature}\}$$

Issue: Health is *not* an affine function of temperature! Affine functions can only express:

Very high is bad, very low is good.

Very low is bad, very high is good.

But both extremes are bad!

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Non-Monotonicity

Solution 1

Transform the input:

$$\phi(x) = \left(1,\ (\text{temperature}(x) - 37)^2\right)$$

where 37 is the “normal” temperature in Celsius.

Issue: requires manually-specified domain knowledge.

Do we really need that?

What does $w^\top \phi(x)$ look like?

From Percy Liang’s "Lecture 3" slides from Stanford’s CS221, Autumn 2014.

Non-Monotonicity

Solution 2

What does $w^\top \phi(x)$ look like? A quadratic function!

$$\phi(x) = \big(1, \text{ temperature}(x), \text{ temperature}(x)^2\big)$$

More expressive than Solution 1.

General principle: Features should be simple building blocks that can be pieced together.

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Saturation

Issue

Problem: Find products relevant to a user's query.

Feature map: $\phi(x) = (1,\ N(x))$, where $N(x)$ is the number of people who bought x.

Action: Predict a "relevance score" $y \in \mathbb{R}$ to user's query (positive is more relevant).

Hypothesis space:

$$\mathscr{H} = \{x \mapsto w^\top \phi(x) : w \in \mathbb{R}^2\} = \{\text{affine functions of } N(x)\}$$

We expect a monotonic relationship between $N(x)$ and relevance, but we also expect **diminishing returns.**

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Saturation

Solution

Smooth nonlinear transform:

$$\phi(x) = (1,\ \log(1 + N(x)))$$

Using $\log(\cdot)$ is good for values with large dynamic ranges.

Discretization (discontinuous transform):

$$\phi(x) = (\mathbf{1}\{0 \le N(x) < 10\}, \mathbf{1}\{10 \le N(x) < 100\}, \ldots)$$

Small buckets allow flexible relationship.

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Feature Interactions

Issue

Feature map: $\phi(x) = (\text{height}(x), \text{weight}(x))$

Action: Predict a "health score" $y \in \mathbb{R}$ (positive is good).

Hypothesis space:

$$\mathscr{H} = \{x \mapsto w^\top \phi(x) : w \in \mathbb{R}^2\} = \{\text{linear combinations of height and weight}\}$$

Issue: The weight *relative* to height is what's important!

Impossible to get with these features and a linear classifier.

Need some **interaction** between height and weight.

Feature Interactions

Solution 1

Google: "Ideal weight from height."

J.D. Robinson's "ideal weight" formula (for a male):

$$\text{weight(kg)} = 52 + 1.9 * (\text{height(in)} - 60)$$

Make score square deviation between height and ideal weight:

$$f(x) = (52 + 1.9(h(x) - 60) - w(x))^2$$

$$f(x) = 3.61h(x)^2 - 3.8h(x)w(x) - 235.6h(x) + w(x)^2 + 124w(x) + 3844$$

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Feature Interactions

Solution 2

Just include all second order features:

$$\phi(x) = \left(1,\ h(x),\ w(x),\ h(x)^2, w(x)^2, h(x)w(x)\right)$$

More flexible, no Googling.

General principle: Simple building blocks replace a single "smart" feature.

Feature Interactions

Monomial Interaction Terms

Interaction terms are useful building blocks to model non-linearities in features.

Start with $x = (1, x_1, \ldots, x_d) \in \mathbb{R}^{d+1} = \mathcal{X}$.

Consider adding all **monomials** of degree M: $x_1^{p_1} \ldots x_d^{p_d}$ with $p_1 + \ldots + p_d = M$.

Monomials with degree 2 in $d = 2$ are: $x_1^2, x_2^2, x_1 x_2$.

How many features will we end up with? $\binom{M + d - 1}{M}$ ("stars and bars").

This leads to **extremely large data matrices** ($d = 40$ and $M = 8$ gives 314457495 features).

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Big Feature Spaces

Motivation for Kernels

This leads to **extremely large data matrices** ($d = 40$ and $M = 8$ gives 314457495 features).

Very large feature spaces can have two potential issues:

1. Overfitting.
2. Memory and computational costs.

Solutions:

1. Handle overfitting with regularization.
2. **Kernel methods** can help with memory and computational costs in high-dimensional space.

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Outline

SVM Objective

With explicit feature map

Let $\phi : \mathcal{X} \to \mathbb{R}^d$ be a feature map.

The SVM objective (with explicit feature map):

$$\min_{w \in \mathbb{R}^d} \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n} \max\left(0, 1 - y^{(i)} w^\top \phi(x^{(i)})\right)$$

Computation is costly if d is large (e.g. with high-degree monomials).

SVM Dual Problem

With explicit feature map

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y^{(i)} y^{(j)} \phi(x^{(j)})^\top \phi(x^{(i)})$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y^{(i)} = 0 \quad \text{and} \quad \alpha_i \in \left[0, \frac{C}{n}\right] \quad \text{for } i = 1, \ldots, n$$

If α^* is an optimal value, then:

$$w^* = \sum_{i=1}^{n} \alpha_i^* y^{(i)} \phi(x^{(i)}) \quad \text{and} \quad \hat{f}(x) = w^\top \phi(x) = \sum_{i=1}^{n} \alpha_i^* \phi(x^{(i)})^\top \phi(x)$$

Key Idea: $\phi(x)$ only shows up in **inner products** with another $\phi(x')$ in training *and* prediction.

Computing Inner Products

Example: Degree-2 Monomials

Consider data in $\mathbb{R}^2$ and introduce **degree-2 monomials** using $\phi : \mathbb{R}^2 \to \mathbb{R}^3$.

$$(x_1,\ x_2) \mapsto (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

The inner product is

$$\begin{aligned}\phi(x)^\top\phi(z) &= x_1^2z_1^2 + (\sqrt{2}x_1x_2)(\sqrt{2}z_1z_2) + x_2z_2^2 \\ &= (x_1z_1)^2 + 2(x_1z_1)(x_2z_2) + (x_2z_2)^2 \\ &= (x_1z_1 + x_2z_2)^2 \\ &= (x^\top z)^2\end{aligned}$$

We can calculate $\phi(x)^\top\phi(z)$ in original input space without accessing the features $\phi(x)$!

Computing Inner Products

Example: Monomials up to degree-2

Consider **monomials up to degree-2** using $\phi : \mathbb{R}^2 \to \mathbb{R}^5$.

$$(x_1,\, x_2) \mapsto (1,\, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

Similarly, the inner product can be computed by:

$$\phi(x)^\top \phi(z) = (1 + x^\top z)^2.$$

For feature maps producing monomials up to degree-*p*: $\phi(x)^\top \phi(z) = (1 + x^\top z)^p$

Kernel trick: we do not need explicit features to compute inner products!

Using explicit features takes $O(d^p)$ time. Using implicit computation only $O(d)$ time!

The Kernel Function

Setup

Input space: $\mathcal{X}$

Feature space: H (a Hilbert space, e.g. $\mathbb{R}^d$, a vector space with an inner product)

Feature map: $\phi : \mathcal{X} \to H$

The **kernel function** corresponding to ϕ is

$$k(x, z) = \langle \phi(x), \phi(z) \rangle, \text{ where } \langle \cdot , \cdot \rangle \text{ is the inner product for } H.$$

Why introduce new notation $k(x, z)$?

We can often evaluate $k(x, x')$ without computing $\phi(x)$ and $\phi(z)$ explicitly!

Kernelized Algorithms

Example: SVM

A method can be kernelized if every feature vector $\phi(x)$ only appears inside an inner product with another feature vector $\phi(z)$. This applies to both the algorithm/optimization problem and the prediction function.

SVM Optimization:

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \phi(x^{(j)})^\top \phi(x^{(i)})$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y^{(i)} = 0 \quad \text{and} \quad \alpha_i \in \left[0, \frac{C}{n}\right] \quad \text{for } i = 1, \ldots, n$$

SVM prediction: $\hat{f}(x) = w^\top \phi(x) = \sum_{i=1}^{n} \alpha_i^* \phi(x^{(i)})^\top \phi(x)$

Kernel Matrix

Definition

The kernel matrix for a kernel $k(\cdot, \cdot)$ on $x_1, \ldots, x_n \in \mathcal{X}$ is:

$$K = (k(x^{(i)}, x^{(j)}))_{i,j} = \begin{bmatrix} k(x^{(1)}, x^{(1)}) & \ldots & k(x^{(1)}, x^{(n)}) \\ \vdots & \ddots & \vdots \\ k(x^{(n)}, x^{(1)}) & \ldots & k(x^{(n)}, x^{(n)}) \end{bmatrix} \in \mathbb{R}^{n \times n}$$

This matrix summarizes all the information we need about training inputs $x^{(1)}, \ldots, x^{(n)}$ to solve a kernelized optimization problem:

$$K_{i,j} = k(x^{(i)}, x^{(j)})$$

Kernelized Algorithms

Example: SVM

In kernelized SVM, we can replace $\phi(x^{(i)})^\top \phi(x^{(j)})$ with K_{ij}:

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} K_{ij}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y^{(i)} = 0 \quad \text{and} \quad \alpha_i \in \left[0, \frac{C}{n}\right] \quad \text{for } i = 1, \ldots, n$$

Kernel Methods

High-level Advantages

Given a kernelized algorithm:

Can swap out the inner product for a new kernel function.

New kernel might correspond to *very high-dimensional* feature space.

Once kernel is computed, computational cost depends on **number of data points** n instead of the **dimension of feature space** d.

Useful when $d \gg n$.

Computing kernel matrix may still depend on d: “kernel trick” avoids $O(d)$ dependence.

Outline

Kernels

As “similarity score”

Often useful to think of $k(x, z)$ as a **“similarity score”** for x and z.

We can design similarity functions without thinking about explicit feature map (e.g. “string kernels” or “graph kernels”).

How do we know that kernel functions correspond to inner products in some feature space?

Kernels

How to obtain kernels?

Two approaches:

1. Explicitly construct $\phi(x) : \mathscr{X} \to \mathbb{R}^d$ (e.g. monomials) and define $k(x, z) = \phi(x)^\top \phi(z)$.

2. Directly define $k(x, z)$ as a "similarity score" and verify it corresponds to $\langle \phi(x), \phi(z) \rangle$ for some ϕ.

There are many theorems to help us with second approach.

Positive Semidefinite Matrix

Definition

A real, symmetric matrix $M \in \mathbb{R}^{n \times n}$ is positive semidefinite (PSD) if for any $x \in \mathbb{R}^n$,

$$x^\top M x \geq 0.$$

The following conditions are equivalent (necessary and sufficient conditions) for a symmetric matrix M to be positive semidefinite:

M can be factorized as $M = R^\top R$ for some matrix R.

All eigenvalues of M are nonnegative.

Positive Semidefinite Matrix

Definition

A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite kernel on $\mathcal{X}$ if for *any* finite set $\{x^{(1)}, \ldots, x^{(n)}\} \in \mathcal{X}$, the kernel matrix on this set

$$K = (k(x^{(i)}, x^{(j)}))_{i,j} = \begin{bmatrix} k(x^{(1)}, x^{(1)}) & \ldots & k(x^{(1)}, x^{(n)}) \\ \vdots & \ddots & \vdots \\ k(x^{(n)}, x^{(1)}) & \ldots & k(x^{(n)}, x^{(n)}) \end{bmatrix} \in \mathbb{R}^{n \times n}$$

is a positive semidefinite matrix.

Note: The kernel matrix needs to be positive semidefinite for **any** finite set of points.

Equivalent definition: $\displaystyle\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(x^{(i)}, x^{(j)}) \geq 0$ for any $\alpha \in \mathbb{R}^n$.

Mercer's Thoerem

Upshot of PD Kernels

A symmetric function $k(x, z)$ can be expressed as an inner product

$$k(x, z) = \langle \phi(x), \phi(z) \rangle$$

for some ϕ if and only if $k(x, z)$ is **positive definite**.

Proving a kernel function is positive definite is typically not easy.

But we can construct new kernels from valid kernels.

Generating New Kernels

"Algebra" of Kernels

Suppose $k, k_1, k_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ are PD kernels. Then so are the following:

Non-negative scaling: $k_{new}(x, z) = \alpha k(x, z)$

Sum: $k_{new}(x, z) = k_1(x, z) + k_2(x, z)$

Product: $k_{new}(x, z) = k_1(x, z) k_2(x, z)$

Recursion: $k_{new}(x, z) = k(\phi(x), \phi(z))$ for any function $\phi(\cdot)$.

1D feature map: $k_{new}(x, z) = f(x) f(z)$ for any function $f : \mathcal{X} \to \mathbb{R}$.

Example Kernels

Linear Kernel

Input space: $\mathcal{X} = \mathbb{R}^d$

Feature space: $H = \mathbb{R}^d$, with standard inner product

Feature map:

$$\phi(x) = x$$

Kernel:

$$k(x, z) = x^\top z$$

Example Kernels

Quadratic Kernel

Input space: $\mathcal{X} = \mathbb{R}^d$

Feature space: $H = \mathbb{R}^D$, where $D = 2d + \binom{d}{2} \approx d^2/2$.

Feature map:

$$\phi(x) = \left(x_1, \ldots, x_d, x_1^2, \ldots, x_d^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_ix_j, \ldots, \sqrt{2}x_{d-1}x_d\right)$$

$$k(x, z) = \langle \phi(x), \phi(z) \rangle = \langle x, z \rangle + \langle x, z \rangle^2$$

Explicit computation takes $O(d^2)$ time; implicit kernel computation takes $O(d)$ time.

Example Kernels

Polynomial Kernel

Input space: $\mathcal{X} = \mathbb{R}^d$

Kernel:

$$k(x, z) = (1 + \langle x, z \rangle)^M$$

Corresponds to feature map with all monomials up to degree M.

For any M, computing kernel has same computational cost!

While computing the explicit inner product grows rapidly in M.

Example Kernels

Radial Basis Function (RBF) / Gaussian Kernel

Input space: $\mathscr{X} = \mathbb{R}^d$

Kernel:

$$k(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right),$$ where σ^2 is the "bandwidth" parameter.

Most common nonlinear kernel.

Does it act like a similarity score?

Corresponds to an "infinite dimensional" feature vector.

Kernelized Methods

High-level Recipe

1. Recognize a kernelized problem: $\phi(x)$ only occurs in inner products $\phi(x)^\top \phi(x)$.

2. Pick a kernel function ("similarity score").

3. Compute the kernel matrix ($n \times n$ where n is the dataset size).

4. Optimize the model and make predictions by accessing kernel matrix.

When can we apply kernelization?

Outline

SVM Dual

Solution in "span of the data"

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)})^\top x^{(j)}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y^{(i)} = 0 \quad \text{and} \quad \alpha_i \in \left[0, \frac{C}{n}\right] \quad \text{for } i = 1, \ldots, n$$

Given a dual solution α_i^*, the primal solution is $w^* = \sum_{i=1}^{n} \alpha_i^* y^{(i)} x^{(i)}$.

Notice: w^* is a linear combination of training inputs $x^{(1)}, \ldots, x^{(n)}$.

We say that w^* is "in the span of the data," or $w^* \in \mathrm{span}(x^{(1)}, \ldots, x^{(n)})$.

Ridge Regression

Solution in “span of the data”

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2 + \lambda \|w\|_2^2$$

Closed-form solution of ridge regression:

$$w^* = (X^\top X + \lambda I)^{-1} X^\top y$$

where $X \in \mathbb{R}^{n \times d}$ is the input data matrix, $y \in \mathbb{R}^n$ is the vector of labels.

Ridge Regression

Solution in “span of the data”

$$w* = (X^\top X + \lambda I)^{-1} X^\top y$$

We can rearrange to show, using identity $(A + B)^{-1} = A^{-1} - A^{-1}B(A + B)^{-1}\ldots$

$$w* = X^\top \left(\frac{1}{\lambda} y - \frac{1}{\lambda} X w* \right)$$

$$= X^\top \alpha* = \sum_{i=1}^{n} \alpha_i^* x^{(i)}$$

So $w*$ is in the span of the data, i.e. $w* \in \mathrm{span}(x^{(1)}, \ldots, x^{(n)})$

Ridge Regression

Solution in “span of the data”

$$\arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2 + \lambda \|w\|_2^2$$

We know $w^* \in \mathrm{span}(x^{(1)}, \ldots, x^{(n)})$, so

$$w^* = \sum_{i=1}^{n} \alpha_i x^{(i)} \text{ for coefficients } \alpha_1, \ldots, \alpha_n.$$

So, rather than minimizing over all of $\mathbb{R}^d$, we can minimize over $\mathrm{span}(x^{(1)}, \ldots, x^{(n)})$:

$$w^* = \arg\min_{w \in \mathrm{span}(x^{(1)}, \ldots, x^{(n)})} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2 + \lambda \|w\|_2^2$$

Ridge Regression

Solution in "span of the data"

$$w^* = \sum_{i=1}^{n} \alpha_i x^{(i)} \text{ for coefficients } \alpha_1, \ldots, \alpha_n.$$

We can write this as $w^* = X^\top \alpha$ for some $\alpha \in \mathbb{R}^n$. Minimize over α instead!

$$\text{Original ridge objective: } w^* = \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2 + \lambda \|w\|_2^2$$

$$\text{Reparameterized: } \alpha^* = \arg\min_{\alpha \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} ((X^\top \alpha)^\top x^{(i)} - y^{(i)})^2 + \lambda \|X^\top \alpha\|_2^2$$

Large Feature Spaces

Benefit of reparameterization

The SVM dual and the reparametrized ridge regression solve for $\alpha^* \in \mathbb{R}^n$. When is this useful?

In cases where $d \gg n$!

Example. Suppose we have training set of $n = 300{,}000$ examples (fairly large).

But we want to use a $d = 300{,}000{,}000$ dimension feature space (very large).

e.g. Using higher-order monomial interaction terms as features.

Reparameterization addresses the computational/optimization issue of solving for $300{,}000$ parameters vs. $300{,}000{,}000$ parameters!

Representer Theorem

General Case

$$\min_{w \in H} R(\|w\|) + L(\langle w, x^{(1)} \rangle, \ldots, \langle w, x^{(n)} \rangle)$$

By the **representer theorem**, we can look for w^* in the span of the data:

$$w^* = \underset{w \in \operatorname{span}(x^{(1)}, \ldots, x^{(n)})}{\arg\min} \; R(\|w\|) + L(\langle w, x^{(1)} \rangle, \ldots, \langle w, x^{(n)} \rangle).$$

So we can reparameterize as before:

$$\alpha^* = \underset{\alpha \in \mathbb{R}^n}{\arg\min} \; R\left(\left\| \sum_{i=1}^{n} \alpha_i x^{(i)} \right\| \right) + L\left(\left\langle \sum_{i=1}^{n} \alpha_i x^{(i)}, x^{(1)} \right\rangle, \ldots, \left\langle \sum_{i=1}^{n} \alpha_i x^{(i)}, x^{(n)} \right\rangle \right)$$

Representer Theorem

General Case

$$\min_{w \in H} R(\|w\|) + L(\langle w, x^{(1)} \rangle, \ldots, \langle w, x^{(n)} \rangle)$$

By the **representer theorem**, we can look for w^* in the span of the data:

$$w^* = \underset{w \in \operatorname{span}(x^{(1)}, \ldots, x^{(n)})}{\arg\min} R(\|w\|) + L(\langle w, x^{(1)} \rangle, \ldots, \langle w, x^{(n)} \rangle).$$

The representer theorem says:

The reparameterization we did for *ridge regression* and *SVM* applies more generally.

All norm-regularized linear models can be kernelized!

Representer Theorem

Theorem

Suppose $J(w) = R(\|w\|) + L(\langle w, x^{(1)}\rangle, \ldots, \langle w, x^{(n)}\rangle)$, where:

$w, x^{(1)}, \ldots, x^{(n)} \in H$ for some Hilbert space H,

$\|\cdot\|$ is the norm corresponding to inner product of H (i.e. $\|w\| = \sqrt{\langle w, w\rangle}$),

$R : [0,\infty) \to \mathbb{R}$ is nondecreasing, and $L : \mathbb{R}^n \to \mathbb{R}$ is arbitrary.

Then, if $J(w)$ has a minimizer, it has a minimizer of the form:

$$w* = \sum_{i=1}^{n} \alpha_i x^{(i)}.$$

Representer Theorem

Proof (Optional)

$$J(w) = R(\|w\|) + L(\langle w, x^{(1)} \rangle, \ldots, \langle w, x^{(n)} \rangle)$$

Idea. If $w*$ is a minimizer, its projection onto the span of the data can only decrease $J(w)$.

Proof. Suppose $w*$ is a minimizer. Let $M = \operatorname{span}(\phi(x^{(1)}), \ldots, \phi(x^{(n)}))$ (the "span of the data").

Let $w = \Pi_M(w*)$, the projection of $w*$ onto M so there exists $\alpha \in \mathbb{R}^n$ s.t. $w = \sum_{i=1}^{n} \alpha_i \phi(x^{(i)})$.

Regularizer can only decrease. $\|w\| \leq \|w*\|$ and $R(\cdot)$ nondecreasing so $R(\|w\|) \leq R(\|w*\|)$.

Loss term stays same. If $w^\perp := w* - w$, then $\langle w*, \phi(x^{(i)}) \rangle = \langle w + w^\perp, \phi(x^{(i)}) \rangle = \langle w, \phi(x^{(i)}) \rangle$ because $w^\perp$ is orthogonal to M.

Therefore, $J(w) \leq J(w*)$ and $w = \sum_{i=1}^{n} \alpha_i \phi(x^{(i)})$ is also a minimizer.

Outline

Feature Maps

Feature Extraction

Kernel Methods: SVM Example

Kernels & Examples of Kernels

Kernel SVM, Kernel Ridge, and Representer Theorem