DSC 140B

Representation Learning

Lecture 11 | Part 1

Linear Limitations

Linear Predictors

- Last time, we saw linear prediction functions:

$$\begin{aligned} H(\vec{x}; \vec{w}) &= w_0 + w_1 x_1 + \ldots + w_d x_d \\ &= \text{Aug}(\vec{x}) \cdot \vec{w} \end{aligned}$$

Linear Decision Functions

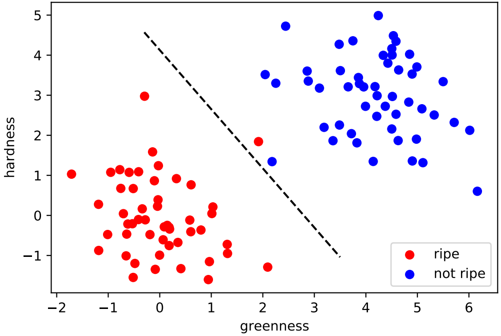
- A linear prediction function H outputs a number.
- What if classes are +1 and -1?
- Can be turned into a **decision function** by taking:

$$\operatorname{sign}(H(\vec{x}))$$

- **Decision boundary** is where $H = 0$
  - Where the sign switches from positive to negative.

Decision Boundaries

- A linear decision function’s decision boundary is linear.
  - A line, plane, hyperplane, etc.



An Example: Parking Predictor

- **Task**: Predict (yes / no): Is there parking available at UCSD right now?

- What training data to collect? What features?

Useful Features

- Time of day?
- Day’s high temperature?
- ...

Exercise

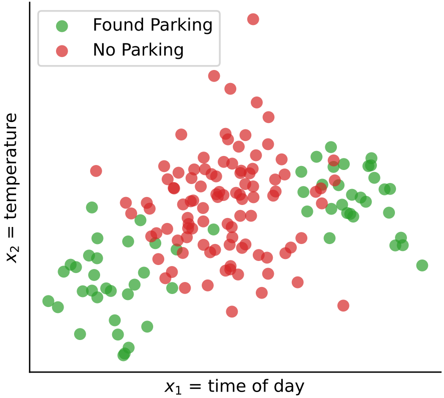
Imagine a scatter plot of the training data with the two features:

- x_1 = time of day
- x_2 = temperature

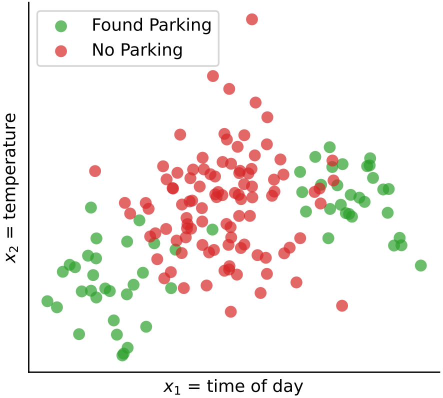
“yes” examples are green, “no” are red.

What does it look like?

Parking Data



Uh oh



- A linear decision function won't work.
- What do we do?

Today’s Question

- How do we learn non-linear patterns using linear prediction functions?

DSC 140B
Representation Learning

Lecture 11 | Part 2

Feature Maps

Representations

- We **represented** the data with two features: time and temperature

- In this **representation**, the trend is **nonlinear**.
  - There is no good linear decision function
  - Learning is “difficult”.

Idea

- **Idea**: We'll make a new **representation** by creating **new features** from the **old features**.

- The "right" representation makes the problem easy again.

- What new features should we create?

New Feature Representation

- Linear prediction functions[1] work well when relationship is linear
  - When x is small we should predict -1
  - When x is large we should predict +1

- But parking's relationship with time is not linear:
  - When time is small we should predict +1
  - When time is medium we should predict -1
  - When time is large we should predict +1

[1] Remember: they are weighted votes.

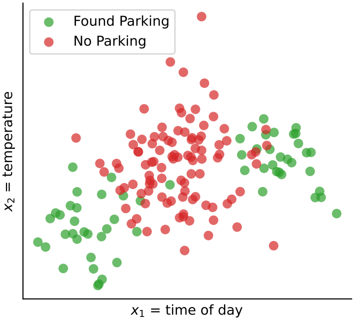
Exercise

How can we “transform” the time of day x_1 to create a new feature x_1' satisfying:

- When x_1' is small, we should predict -1
- When x_1' is large, we should predict +1

What about the temperature, x_2?

Idea



- Transform “time” to “absolute time until/since Noon”
- Transform “temp.” to “absolute difference between temp. and 72°”

Basis Functions

- We will transform:
  - the time, x_1, to $|x_1 - \text{Noon}|$
  - the temperature, x_2, to $|x_2 - 72^\circ|$
- Formally, we've designed non-linear **basis functions**:

$$\varphi_1(x_1, x_2) = |x_1 - \text{Noon}|$$
$$\varphi_2(x_1, x_2) = |x_2 - 72^\circ|$$

- In general a basis function φ maps $\mathbb{R}^d \to \mathbb{R}$

Feature Mapping

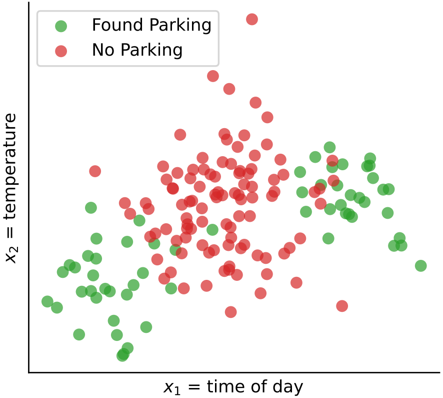
- Define $\vec{\varphi}(\vec{x}) = (\varphi_1(\vec{x}), \varphi_2(\vec{x}))^T$. $\vec{\varphi}$ is a **feature map**
  - Input: vector in "old" representation
  - Output: vector in "new" representation

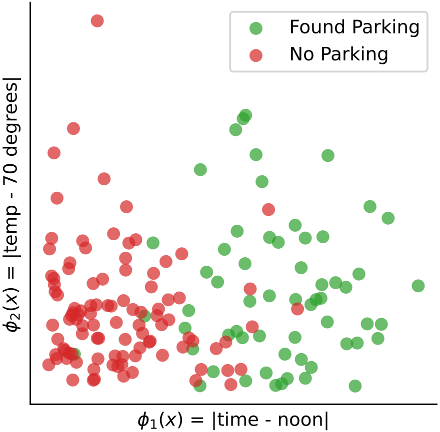
- Example:

$$\vec{\varphi}((10\text{a.m.}, 75^\circ)^T) = (2\text{ hours}, 3^\circ)^T$$

- $\vec{\varphi}$ maps raw data to a **feature space**.

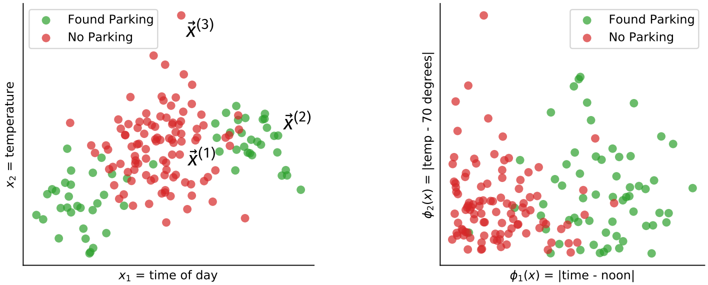
Feature Space, Visualized



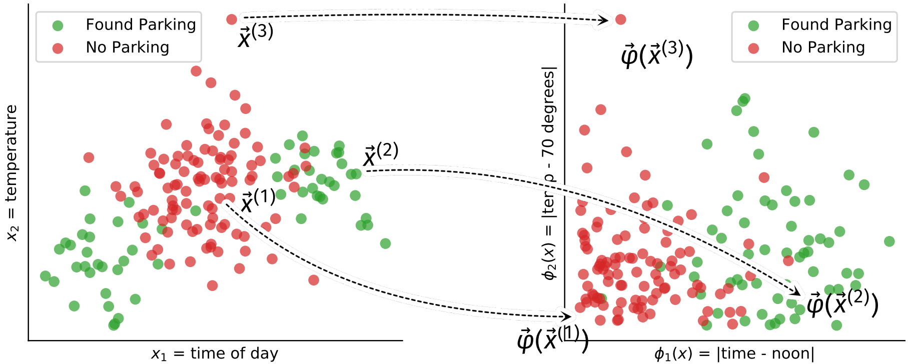


Exercise

Where does $\vec{\varphi}$ map $\vec{x}^{(1)}$, $\vec{x}^{(2)}$, and $\vec{x}^{(3)}$?



Solution



Found Parking
No Parking
$\vec{x}^{(3)}$
$\vec{x}^{(2)}$
$\vec{x}^{(1)}$
x_2 = temperature
x_1 = time of day
$\vec{\varphi}(\vec{x}^{(3)})$
$\vec{\varphi}(\vec{x}^{(2)})$
$\vec{\varphi}(\vec{x}^{(1)})$
Found Parking
No Parking
$\phi_2(x)$ = |temp - 70 degrees|
$\phi_1(x)$ = |time - noon|

After the Mapping

- The basis functions φ_1, φ_2 give us our "new" features.
- This gives us a new **representation**.
- In this representation, learning (classification) is easier.

Training

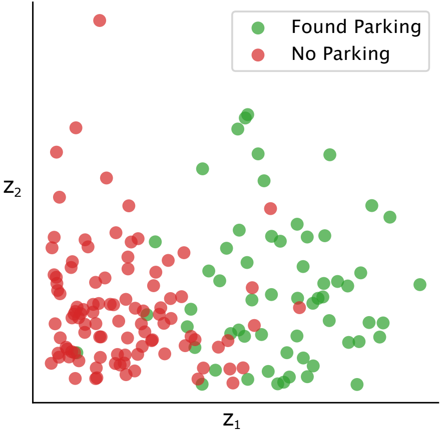
- Map each training example $\vec{x}^{(i)}$ to feature space, creating new training data:

$$\vec{z}^{(1)} = \vec{\varphi}(\vec{x}^{(1)}), \quad \vec{z}^{(2)} = \vec{\varphi}(\vec{x}^{(2)}), \quad \ldots, \quad \vec{z}^{(n)} = \vec{\varphi}(\vec{x}^{(n)})$$

- Fit linear prediction function H in usual way:

$$H_f(\vec{z}) = w_0 + w_1 z_1 + w_2 z_2 + \ldots + w_d z_d$$

Training Data in Feature Space



Prediction

- If we have $\vec{z}$ in feature space, prediction is:

$$H_f(\vec{z}) = w_0 + w_1 z_1 + w_2 z_2 + \ldots + w_d z_d$$

Prediction

- But if we have $\vec{x}$ from original space, we must "convert" $\vec{x}$ to feature space first:

$$\begin{aligned} H(\vec{x}) &= H_f(\vec{\varphi}(\vec{x})) \\ &= H_f(\,(\varphi_1(\vec{x}),\, \varphi_2(\vec{x}),\, \ldots,\, \varphi_d(\vec{x}))^T\,) \\ &= w_0 + w_1\varphi_1(\vec{x}) + w_2\varphi_2(\vec{x}) + \ldots + w_d\varphi_d(\vec{x}) \end{aligned}$$

Overview: Feature Mapping

- A basis function can involve any/all of the original features:

$$\varphi_3(\vec{x}) = x_1 \cdot x_2$$

- We can make more basis functions than original features:

$$\vec{\varphi}(\vec{x}) = (\, \varphi_1(\vec{x}),\ \varphi_2(\vec{x}),\ \varphi_3(\vec{x})\,)^T$$

Overview: Feature Mapping

1. Start with data in original space, $\mathbb{R}^d$.

2. Choose some basis functions, $\varphi_1, \varphi_2, \ldots, \varphi_{d'}$

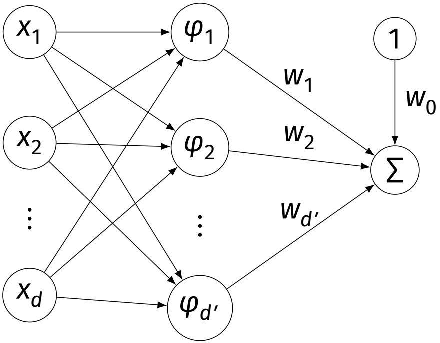
3. Map each data point to **feature space** $\mathbb{R}^{d'}$:

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

4. Fit linear prediction function in new space:

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

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



Today's Question

- Q: How do we learn non-linear patterns using linear prediction functions?

- A: Use non-linear basis functions to map to a feature space.

DSC 140B

Representation Learning

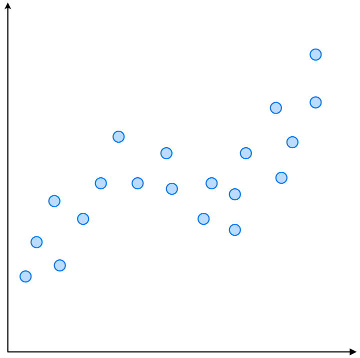
Lecture 11 | Part 3

Basis Functions and Regression

By the way...

- You've (probably) seen basis functions used before.
- Linear regression for non-linear patterns in DSC 40A.

Example



Fitting Non-Linear Patterns

- Fit function of the form

$$H(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$$

- Linear function of $\vec{w}$, non-linear function of x.

The Trick

- Treat x, x^2, x^3, x^4 as **new** features.
- Create design matrix:

$$X = \begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 & x_1^4 \\ 1 & x_2 & x_2^2 & x_2^3 & x_2^4 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 & x_n^4 \end{pmatrix}$$

- Solve $X^T X\vec{w} = X^T\vec{w}$ for $\vec{w}$, as usual.
- Works for more than just polynomials.

Another View

- We have changed the representation of a point:

$$x \mapsto (x, x^2, x^3, x^4)$$

- Basis functions:

$$\varphi_1(x) = x \quad \varphi_2(x) = x^2 \quad \varphi_3(x) = x^3 \quad \varphi_4(x) = x^4$$

DSC 140B

Representation Learning

Lecture 11 | Part 4

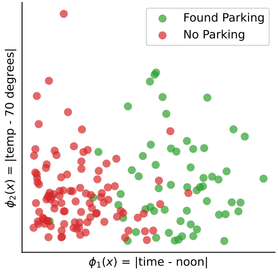
A Tale of Two Spaces

A Tale of Two Spaces

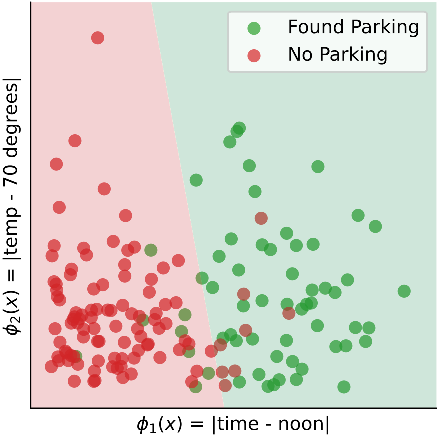
- The **original space**: where the raw data lies.
- The **feature space**: where the data lies after feature mapping $\vec{\varphi}$
- Remember: we fit a linear prediction function in the **feature space**.

Exercise

- In **feature space**, what does the decision boundary look like?
- What does the prediction function surface look like?

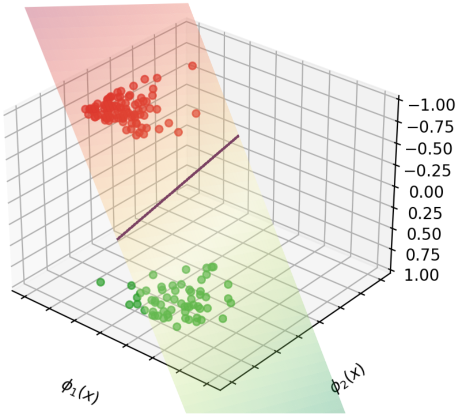


Decision Boundary in Feature Space[2]



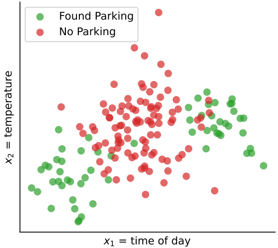
[2] Fit by minimizing square loss

Prediction Surface in Feature Space

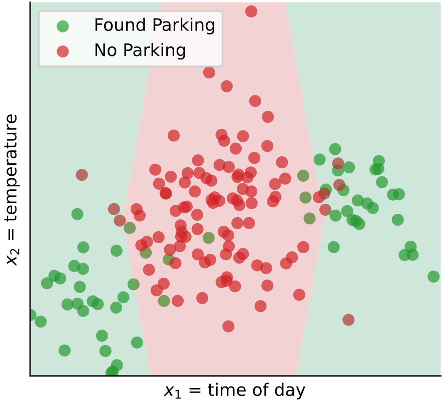


Exercise

- In the **original space**, what does the decision boundary look like?
- What does the prediction function surface look like?

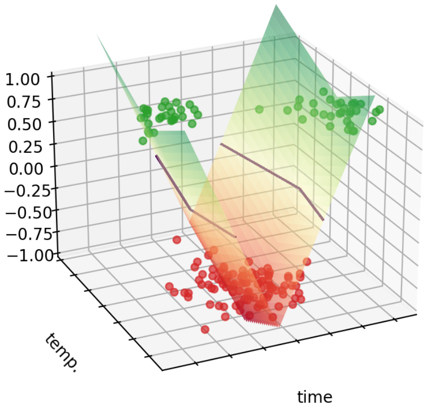


Decision Boundary in Original Space[3]



[3] Fit by minimizing square loss

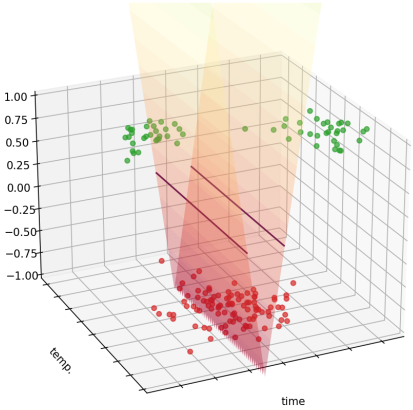
Prediction Surface in Original Space



Insight

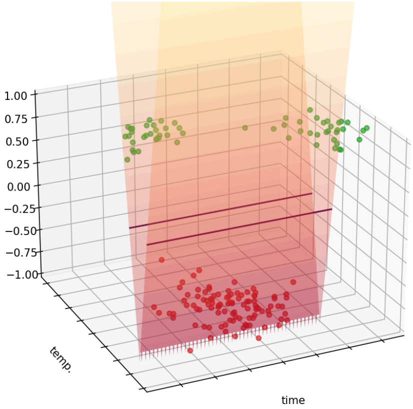
- H is a sum of basis functions, φ_1 and φ_2.
    - $H(\vec{x}) = w_0 + w_1\varphi_1(\vec{x}) + w_2\varphi_2(\vec{x})$
- The prediction surface is a sum of other surfaces.
- Each basis function is a “building block”.

Visualizing the Basis Function φ_1



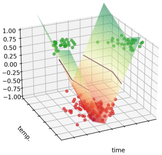
- $w_0 + w_1 |x_1 - \text{noon}|$

Visualizing the Basis Function φ_2

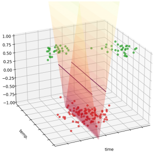


- $w_0 + w_2|x_2 - 72^\circ|$

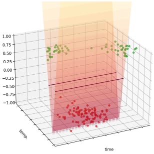
Visualizing the Prediction Surface



=

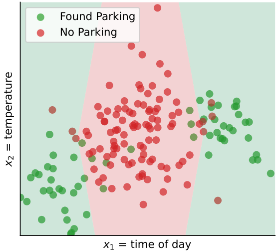


+



Exercise

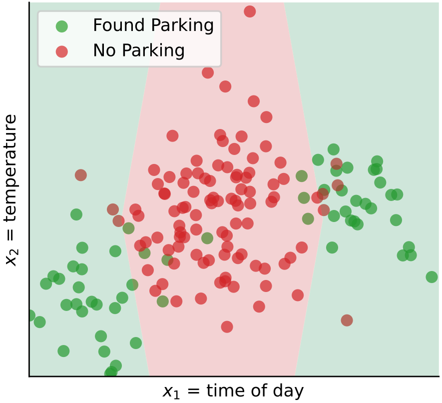
The decision boundary has a single “pocket” where it is negative. Can it have more than one, assuming we use basis functions of the same form? What if we use more than two basis functions?



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.

View: Function Approximation



- Find a function that is ≈ 1 near green points and ≈ −1 near red points.

What's Wrong?

- We've discovered how to learn non-linear patterns using linear prediction functions.
  - Use non-linear basis functions to map to a feature space.
- Something should bug you, though...