DSC 140A

Probabilistic Modeling & Machine Learning

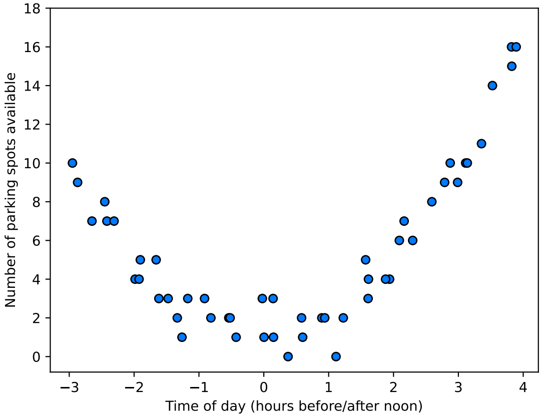
Lecture 8 | Part 1

Feature Maps

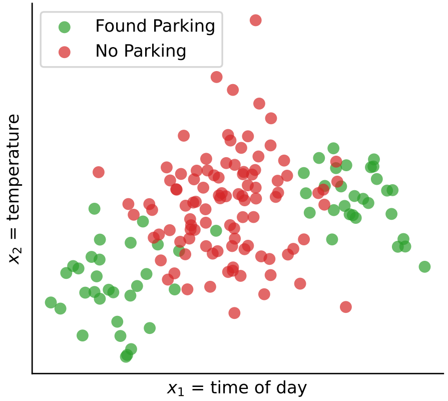
Problem

- Patterns in real world data are often **non-linear**.
- But we only know how to train **linear predictors**.

Example: Regression



Example: Classification



Today

- **Solution**: non-linear **feature maps**.
- Will allow us to:
  - fit complex, non-linear patterns;
  - while still using linear models (least squares, SVM, ...)
- But we'll need to be careful about **overfitting**.

Feature Map

- A **feature map** $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^k$ is a function that takes in a d-dimensional vector and outputs a k-dimensional feature vector.
- I.e., it creates new features from the old ones.
  - Maybe in a non-linear way.

Example

- Define $\vec{\phi} : \mathbb{R}^2 \to \mathbb{R}^5$ as:

$$\vec{\phi}(x_1, x_2) = (x_1,\ x_2,\ x_1^2,\ x_2^2,\ x_1 x_2)^T$$

- If $\vec{x} = (2, 3)^T$, then:

$$\begin{aligned}\vec{\phi}(\vec{x}) &= (2,\ 3,\ 2^2,\ 3^2,\ 2 \times 3)^T \\ &= (2,\ 3,\ 4,\ 9,\ 6)^T\end{aligned}$$

Basis Functions

- A **basis function** is a function $\phi_i : \mathbb{R}^d \to \mathbb{R}$.
- It takes in an old feature vector and outputs a single new feature.
- We can think of a feature map $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^k$ as being made up of k basis functions.

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

Example

- Let $\vec{\phi} : \mathbb{R}^2 \to \mathbb{R}^5$ be defined as:

$$\vec{\phi}(x_1, x_2) = (x_1, x_2, x_1^2, x_2^2, x_1 x_2)^T$$

- The corresponding basis functions are:

$$\phi_1(x_1, x_2) = x_1 \qquad \phi_2(x_1, x_2) = x_2$$
$$\phi_3(x_1, x_2) = x_1^2 \qquad \phi_4(x_1, x_2) = x_2^2$$
$$\phi_5(x_1, x_2) = x_1 x_2$$

A New Data Set

- Say we have a training set with d features:

$$(\vec{x}^{(1)}, y_1), \ldots, (\vec{x}^{(n)}, y_n)$$

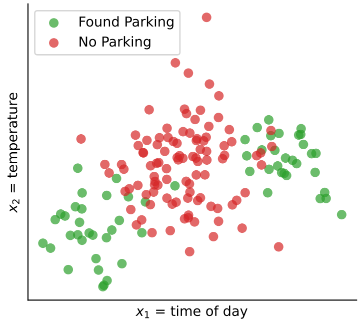
- A feature map $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^k$ gives us a **new** training set with k features:

$$(\vec{\phi}(\vec{x}^{(1)}), y_1), \ldots, (\vec{\phi}(\vec{x}^{(n)}), y_n)$$

Why?

- A (good) feature map can turn **non-linear** patterns in the old data into **linear patterns** in the new data.

Example: Parking Classification



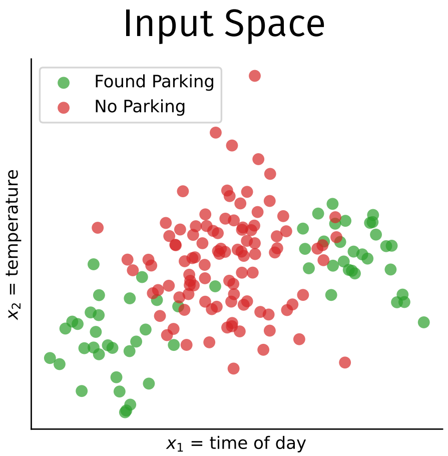
- Original features:

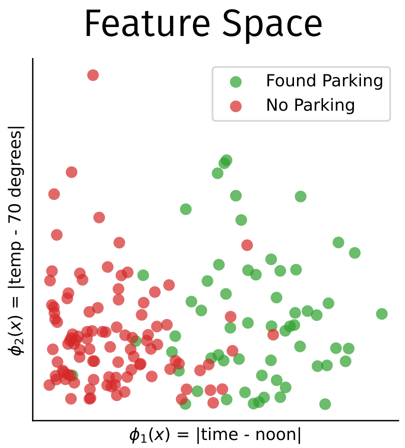
$$\vec{x} = (\text{time}, \text{temp.})^T$$

- Feature map:

$$\vec{\phi}(\vec{x}) = (|\text{time} - \text{Noon}|, |\text{temp.} - 70|)^T$$

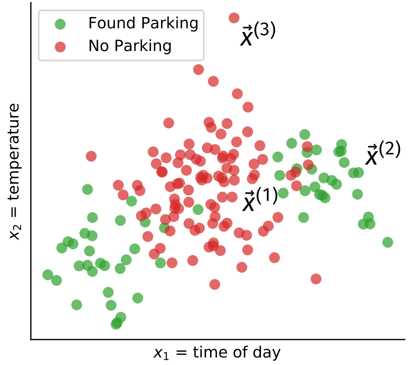
Example: Parking Classification

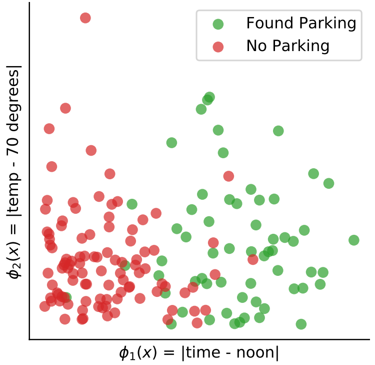




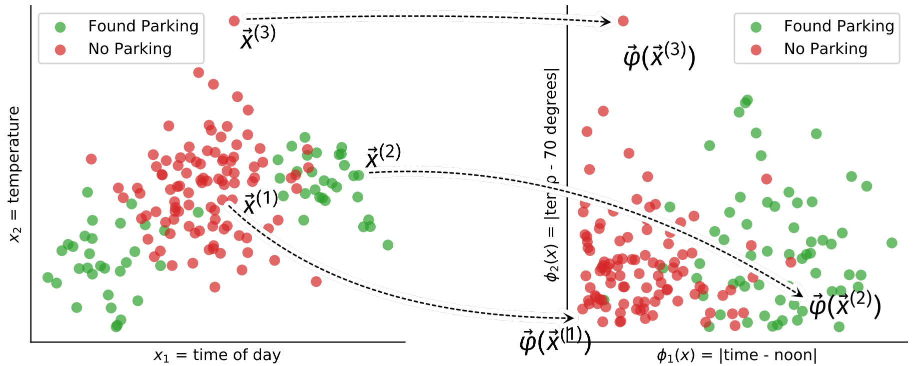
Exercise

(Approximately) where do $\vec{x}^{(1)}$, $\vec{x}^{(2)}$, and $\vec{x}^{(3)}$ get mapped to in feature space?





Solution


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

Idea

- Feature maps turned **non-linear** patterns in input space into **linear** patterns in feature space.
- **Idea:** train a linear model in feature space.

Procedure: Learning with Feature Maps

- First, pick a feature map $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^k$.
- **To train**:
  - Given training set $(\vec{x}^{(1)}, y_1), \ldots, (\vec{x}^{(n)}, y_n)$.
  1. Map each $\vec{x}^{(i)}$ to feature space, creating a new data set $(\vec{\phi}(\vec{x}^{(1)}), y_1), \ldots, (\vec{\phi}(\vec{x}^{(n)}), y_n)$.
  2. Train linear model (least squares, SVM, perceptron...) on the new data in feature space to get $\vec{w}^*$.
- **To predict:**
  - Given new input $\vec{x}$.
  1. Map $\vec{x}$ to feature space: $\vec{\phi}(\vec{x})$.
  2. Predict $H(\vec{x}; \vec{w}^*) = \vec{w}^* \cdot \operatorname{Aug}(\vec{\phi}(\vec{x}))$.

Exercise

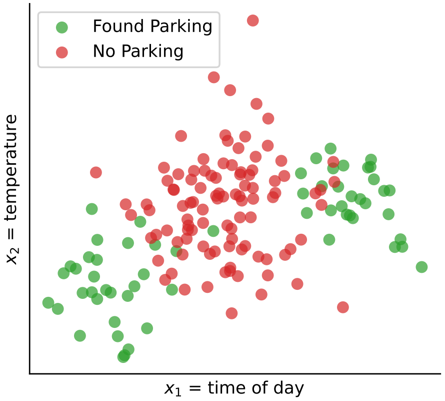
Suppose the original feature vectors are in $\mathbb{R}^2$ and the feature map is defined as

$$\vec{\phi}(x_1, x_2) = (x_1, x_2, x_1^2, x_2^2, x_1 x_2)^T$$

We train an SVM in feature space. What is the dimensionality of $\vec{w}^*$?

Example: Least Squares

- Let’s train a least squares classifier using a feature map.



Step 1: Pick a Feature Map

- In the input space, we have features (x_1, x_2).

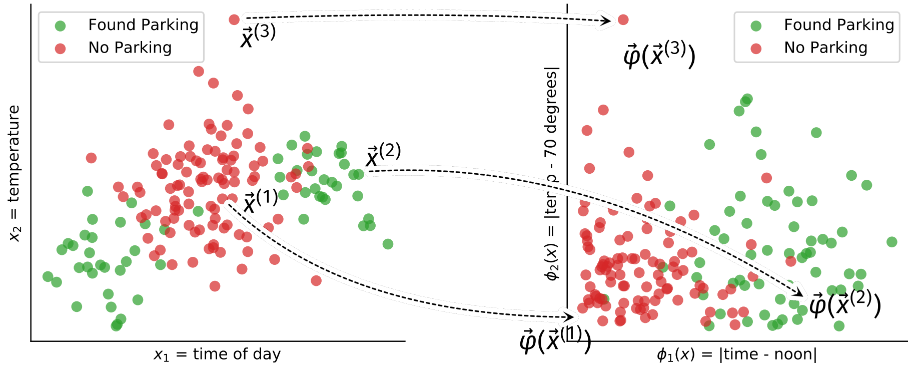
$$x_1 = \text{time}, \quad x_2 = \text{temperature}.$$

- We'll use the same feature map as before:

$$\vec{\phi}(x_1, x_2) = (|x_1 - 12|, |x_2 - 70|)^T$$

Step 2(a): Map to Feature Space

- Map every data point to feature space.



Step 2(b): Train in Feature Space

- Recall: we train a least squares classifier in **input space** by computing:

$$\vec{w}^* = (\vec{X}^T \vec{X})^{-1} \vec{X}^T \vec{y}$$

- Here, X is the (augmented) $(n \times d)$ design matrix:

$$X = \begin{pmatrix} \text{Aug}(\vec{x}^{(1)})^T \longrightarrow \\ \text{Aug}(\vec{x}^{(2)})^T \longrightarrow \\ \vdots \\ \text{Aug}(\vec{x}^{(n)})^T \longrightarrow \end{pmatrix} = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} \\ \vdots & \vdots & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} \end{pmatrix}$$

Step 2(b): Train in Feature Space

- In feature space, our feature vectors are $\vec{\phi}(\vec{x}^{(1)}), \ldots, \vec{\phi}(\vec{x}^{(n)})$.
- So the design matrix becomes the $(n \times k)$ matrix:

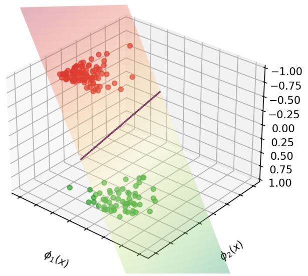
$$\Phi = \begin{pmatrix} \vec{\phi}(\vec{x}^{(1)})^T \longrightarrow \\ \vec{\phi}(\vec{x}^{(2)})^T \longrightarrow \\ \vdots \\ \vec{\phi}(\vec{x}^{(n)})^T \longrightarrow \end{pmatrix} = \begin{pmatrix} 1 & |x_1^{(1)} - 12| & |x_2^{(1)} - 70| \\ 1 & |x_1^{(2)} - 12| & |x_2^{(2)} - 70| \\ \vdots & \vdots & \vdots \\ 1 & |x_1^{(n)} - 12| & |x_2^{(n)} - 70| \end{pmatrix}$$

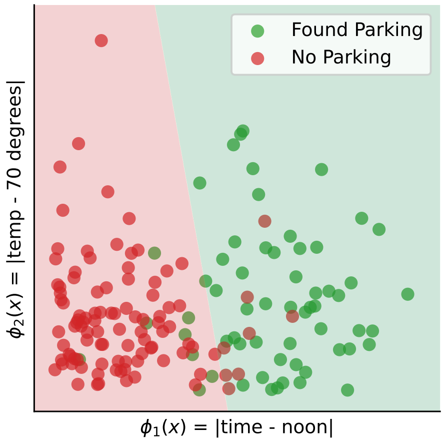
Step 2(b): Train in Feature Space

- The least squares solution in feature space is:

$$\vec{w}^* = (\Phi^T \Phi)^{-1} \Phi^T \vec{y}$$

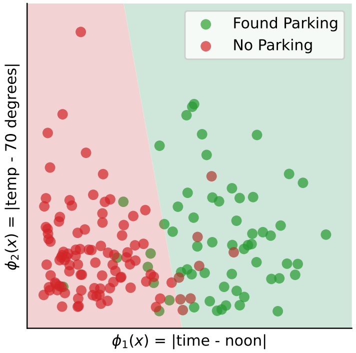
Solution in Feature Space





Step 3: Predict

- Given a new example $\vec{x}$ in input space:
    1. Map $\vec{x}$ to feature space: $\vec{\phi}(\vec{x})$.
    2. Predict $\text{sign}(\vec{w}^* \cdot \text{Aug}(\vec{\phi}(\vec{x})))$.



Exercise

Let $\vec{\phi}(x_1, x_2) = (|x_1 - 12|, |x_2 - 70|)^T$. Suppose we train a least squares classifier in feature space and find $\vec{w}^* = (3, -1, 2)^T$.

Given a new point $\vec{x} = (10, 65)^T$ in input space, what is the prediction, $H(\vec{x})$?

The Prediction Function(s)

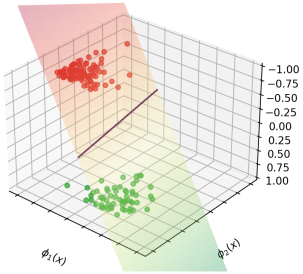
- There are, in a sense, **two** prediction functions to consider.
- First, the prediction function in feature space:

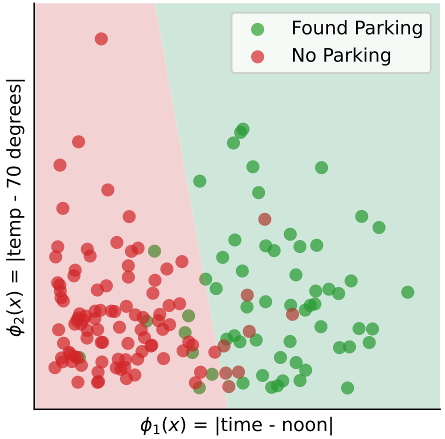
$$\begin{aligned} H_\phi(\vec{z}) &= \vec{w} \cdot \text{Aug}(\vec{z}) \\ &= w_0 + w_1 z_1 + w_2 z_2 + \ldots + w_k z_k \end{aligned}$$

- This function takes in a vector $\vec{z}$ that is already in feature space.

H_ϕ in Feature Space

$$H_\phi(\vec{z}) = w_0 + w_1 z_1 + w_2 z_2$$





The Prediction Function

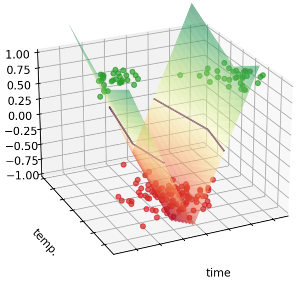
- There is also the prediction function $H(\vec{x})$ that takes in vectors in input space.

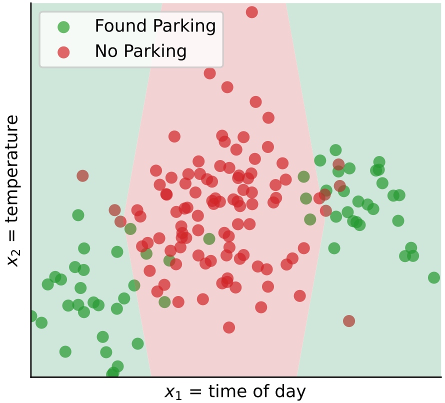
$$\begin{aligned} H(\vec{x}) &= H_\phi(\vec{\phi}(\vec{x})) \\ &= \vec{w} \cdot \text{Aug}(\vec{\phi}(\vec{x})) \\ &= w_0 + w_1\phi_1(\vec{x}) + w_2\phi_2(\vec{x}) + \ldots + w_k\phi_k(\vec{x}) \end{aligned}$$

- When plotted, this function will look **non-linear**.

H in Input Space

$$H(\vec{x}) = w_0 + w_1|x_1 - 12| + w_2|x_2 - 70|$$





Exercise

Let $\vec{\phi}(x_1, x_2) = (|x_1 - 12|, |x_2 - 70|)^T$. Suppose we train a least squares classifier in feature space and find $\vec{w}^* = (3, -1, 2)^T$.

Given a new point $\vec{x} = (10, 65)^T$ in input space, what is the prediction, $H(\vec{x})$? This time, compute the answer *without* explicitly computing $\vec{\phi}(\vec{x})$.

DSC 140A

Probabilistic Modeling & Machine Learning

Lecture 8 | Part 2

Example: Non-Linear Regression

Non-Linear Regression

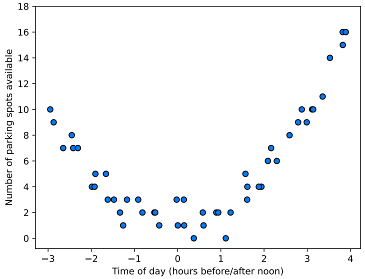
- With a feature map $\vec{\phi}(\vec{x}) = (\phi_1(\vec{x}), \dots, \phi_k(\vec{x}))^T$, our prediction function becomes:

$$H(\vec{x}) = w_0 + w_1\phi_1(\vec{x}) + w_2\phi_2(\vec{x}) + \dots + w_k\phi_k(\vec{x})$$

- In other words, we're not constrained to only fitting straight lines/planes:

$$H(x) = w_0 + w_1 x$$

Example: Parking Regression



- Data looks like a quadratic function.

- Idea: fit a function of the form:

$$H(t) = w_0 + w_1 t + w_2 t^2$$

Exercise

Suppose we wish to fit a function of the form $H(t) = w_0 + w_1 t + w_2 t^2$ to the data.

What feature map $\vec{\phi}$ should we use to get this form of prediction function?

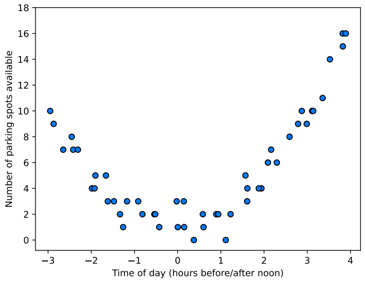
Answer

- Use $\vec{\phi}(t) = (t, t^2)^T$.

- Then the prediction function is:

$$\begin{aligned} H(t) &= \vec{w} \cdot \text{Aug}(\vec{\phi}(t)) \\ &= (w_0, w_1, w_2) \cdot (1, t, t^2)^T \\ &= w_0 + w_1 t + w_2 t^2 \end{aligned}$$

Example: Parking Regression



- Original features:

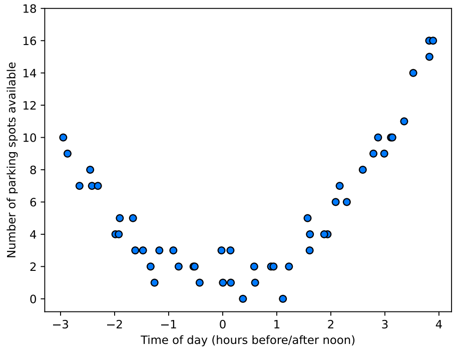
$$\vec{x} = (\text{time})^T$$

- Feature map:

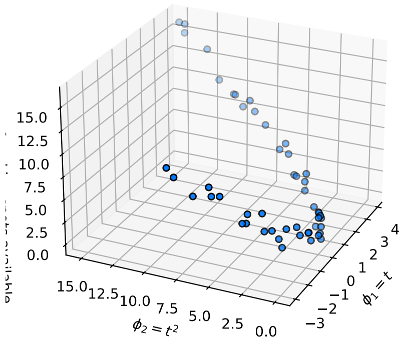
$$\vec{\phi}(\vec{x}) = (\text{time}, \text{time}^2)^T$$

Example: Parking Regression

Input Space



Feature Space



Least Squares

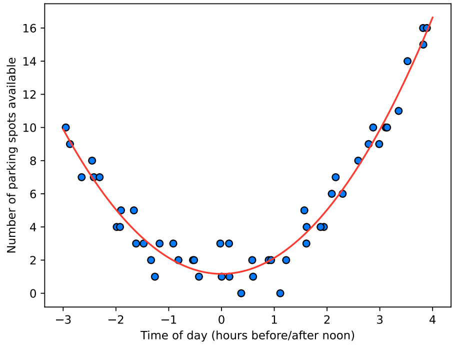
- After mapping to feature space, we fit a plane with least squares.

- The design matrix becomes:

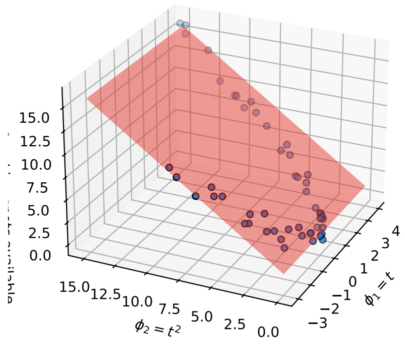
$$\Phi = \begin{pmatrix} \mathrm{Aug}(t^{(1)})^T \longrightarrow \\ \mathrm{Aug}(t^{(2)})^T \longrightarrow \\ \vdots \\ \mathrm{Aug}(t^{(n)})^T \longrightarrow \end{pmatrix} = \begin{pmatrix} 1 & t^{(1)} & (t^{(1)})^2 \\ 1 & t^{(2)} & (t^{(2)})^2 \\ \vdots & \vdots & \vdots \\ 1 & t^{(n)} & (t^{(n)})^2 \end{pmatrix}$$

Example: Parking Regression

Input Space



Feature Space



DSC 140A

Probabilistic Modeling & Machine Learning

Lecture 8 | Part 3

ERM with Feature Maps

Learning with Feature Maps

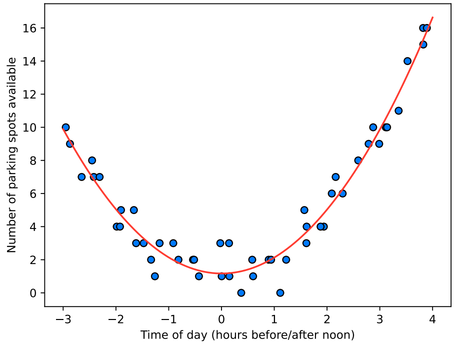
- We’ve developed a procedure for fitting non-linear patterns with linear models.
  - Map to feature space, learn there.
- Is this the “best” approach?

Empirical Risk Minimization

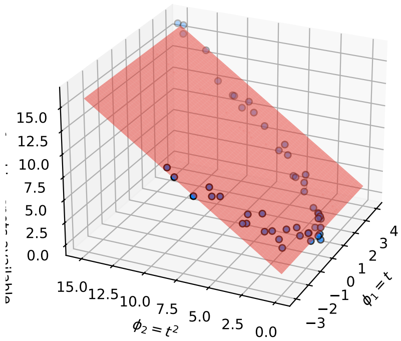
- Step 1: choose a **hypothesis class**
  - Functions of the form $H(\vec{x}) = \vec{w} \cdot \text{Aug}(\vec{\phi}(\vec{x}))$.
- Step 2: choose a **loss function**
  - Square loss, perceptron loss, hinge loss, etc.
- Step 3: find H minimizing **empirical risk**
  - Do we get the same H if we train in feature space?

Example: Parking Regression

Input Space



Feature Space



Yes

- The H_ϕ that minimizes risk in feature space is the same as the H that minimizes risk in input space.
  - As long as H is a linear function of the **parameters**.

Argument

- Take, for example, square loss.
- The risk is:

$$R(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \vec{w} \cdot \text{Aug}(\vec{\phi}(\vec{x}^{(i)})))^2$$

- Minimizer is $\vec{w}^* = (\Phi^T\Phi)^{-1}\Phi^T\vec{y}$.

In General

- Assume prediction function is of the form:

$$H(\vec{x}) = w_0 + w_1\phi_1(\vec{x}) + w_2\phi_2(\vec{x}) + \ldots + w_k\phi_k(\vec{x})$$

- To find $\vec{w}$ that minimizes risk:
  - Map data to feature space;
  - Train a linear model in feature space.

- Works for least squares, perceptron, SVM, etc.

Takeaway

- The “linear” in “linear prediction function” refers to the **parameters**, not the features!
- We can fit any function of the form:

$$H(x) = w_1\phi_1(x) + w_2\phi_2(x) + \ldots + w_k\phi_k(x)$$

DSC 140A

Probabilistic Modeling & Machine Learning

Lecture 8 | Part 4

Gaussian Radial Basis Functions

General Basis Functions

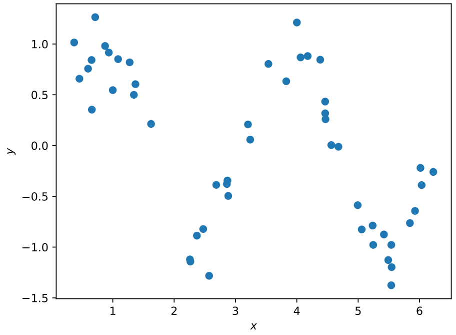
- We can fit any function of the form:

$$H(x) = w_1\phi_1(x) + w_2\phi_2(x) + \ldots + w_k\phi_k(x)$$

- Before: we chose ϕ_i carefully based on the problem.

- Is there an easier way?
  - Are there basis functions that work well for many problems?

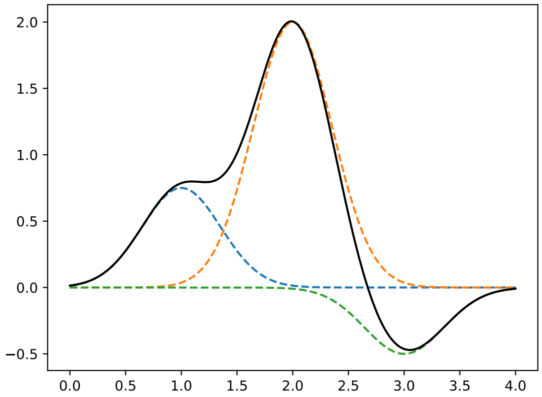
Example



- Suppose we want to fit a function *H* to this data.
- Locally, each part of the curve looks like a “bump”.
- **Idea:** let *H* be a sum of bumps.

A Sum of Bumps

$$H(x) = w_1\,\text{bump}_1(x) + w_2\,\text{bump}_2(x) + w_3\,\text{bump}_3(x)$$



Gaussian Basis Functions

- One way to make a bump: a **Gaussian**

$$\phi_i(x) = \exp\left(-\frac{(x-\mu_i)^2}{\sigma_i^2}\right)$$

- Must specify[1] **center** μ_i and **width** σ_i for each Gaussian basis function.

[1]You pick these; they are not learned!

Exercise

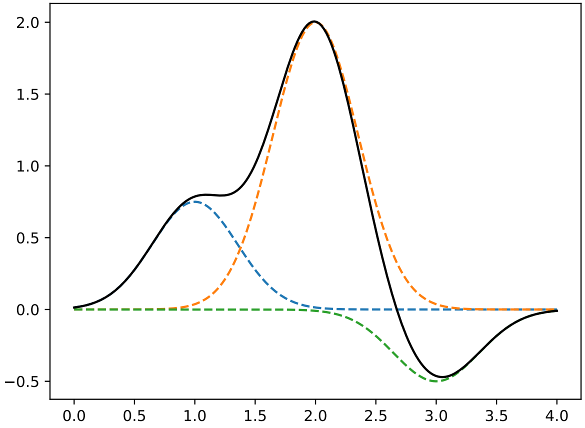
Suppose we have a Gaussian of the form:

$$\phi(x) = \exp\left(-\frac{(x-2)^2}{3}\right)$$

What is the value of $\phi(2)$? What is the value of $\phi(100)$, approximately?

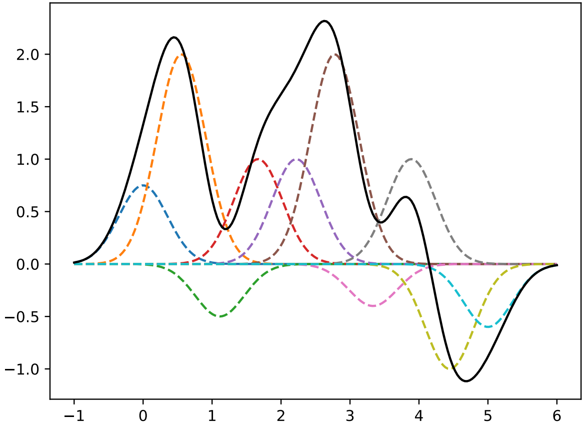
Example: $k = 3$

- A function of the form: $H(x) = w_1\phi_1(x) + w_2\phi_2(x) + w_3\phi_3(x)$, using 3 Gaussian basis functions.



Example: $k = 10$

- The more basis functions, the more complex H can be.



Learning with Gaussian Basis Functions

- Gaussians make for very general basis functions.
- By adjusting $w_1, \ldots, w_k$, we can fit complex patterns.

`https://dsc140a.com/static/vis/gaussian-basis-functions-1d`

Procedure: Learning with Gaussian Basis Functions

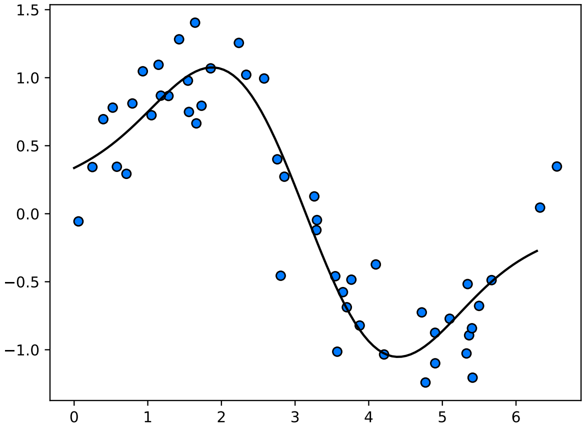
1. Pick number and location of Gaussians.
   - $\mu_1, \ldots, \mu_k$ and $\sigma_1, \ldots, \sigma_k$.

2. Make k basis functions:
   - $\phi_i(x) = \exp\left(-\frac{(x-\mu_i)^2}{\sigma_i^2}\right)$.

3. Map data to feature space and train a linear model as before.

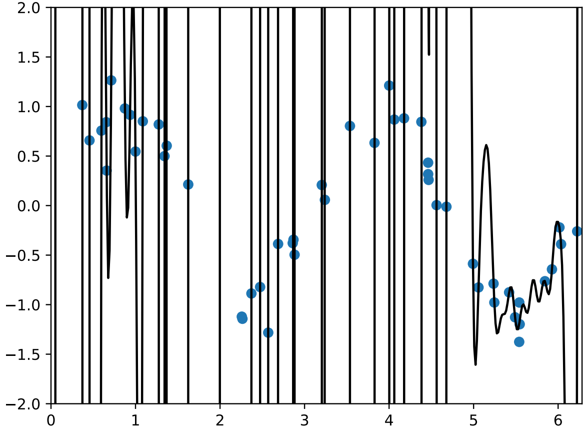
Demo: Sinusoidal Data

- Fit curve to 50 noisy data points.
- Use $k = 4$ Gaussian basis functions.



Demo: Sinusoidal Data

- Fit curve to 50 noisy data points.
- Use $k = 50$ Gaussian basis functions.



Next Time

- How to control **overfitting**.