Feature Maps

He He

Slides based on Lecture 4d from David Rosenberg's course material.

CDS, NYU

March 2, 2021

The Input Space $\mathcal{X}$

- Our general learning theory setup: no assumptions about $\mathcal{X}$

- But $\mathcal{X} = \mathsf{R}^d$ for the specific methods we've developed:
    - Ridge regression
    - Lasso regression
    - Support Vector Machines

- Our hypothesis space for these was all affine functions on R^d:

$$\mathcal{F} = \left\{ x \mapsto w^T x + b \mid w \in \mathsf{R}^d, b \in \mathsf{R} \right\}.$$

- What if we want to do prediction on inputs not natively in R^d?

The Input Space $\mathfrak{X}$

- Often want to use inputs not natively in R^d:
  - Text documents
  - Image files
  - Sound recordings
  - DNA sequences

- But everything in a computer is a sequence of numbers
  - The ith entry of each sequence should have the same "meaning"
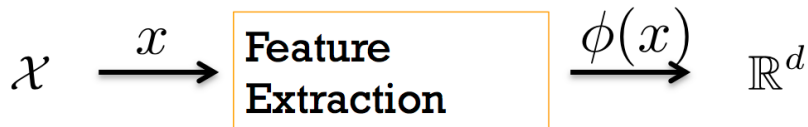 - All the sequences should have the same length

Feature Extraction

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

Raw Input   **Feature Vector**

$$\mathcal{X} \xrightarrow{\;x\;} \boxed{\begin{array}{l}\textbf{Feature}\\\textbf{Extraction}\end{array}} \xrightarrow{\;\phi(x)\;} \mathbb{R}^d$$

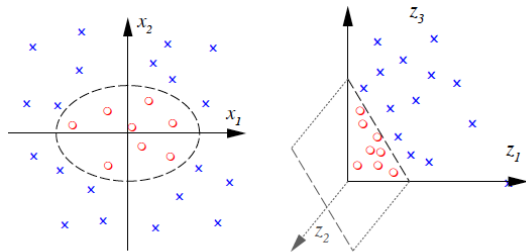# Linear Models with Explicit Feature Map

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

- Introduce **feature map** $\phi : \mathcal{X} \to \mathsf{R}^d$

- The feature map maps into the **feature space** R^d.

- Hypothesis space of affine functions on feature space:

$$\mathcal{F} = \left\{ x \mapsto w^T \phi(x) + b \mid w \in \mathsf{R}^d, b \in \mathsf{R} \right\}.$$

Geometric Example: Two class problem, nonlinear boundary

$$\Phi : R^2 \to R^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{(2)}x_1 x_2, x_2^2)$$

- With identity feature map $\phi(x) = (x_1, x_2)$ and linear models, can't separate regions

- With appropriate featurization $\phi(x) = (x_1, x_2, x_1^2 + x_2^2)$, becomes linearly separable .

- Video: http://youtu.be/3liCbRZPrZA

Expressivity of Hypothesis Space

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

- Sometimes we say a larger hypothesis is more expressive.
  - (can fit more relationships between input and action)

- Many ways to create new features.

Handling Nonlinearity with Linear Methods

Example Task: Predicting Health

- General Philosophy: Extract every feature that might be relevant

- Features for medical diagnosis
    - height
    - weight
    - body temperature
    - blood pressure
    - etc...

Feature Issues for Linear Predictors

- For linear predictors, it's important **how** features are added
  - The relation between a feature and the label may not be linear
  - There may be complex dependence among features

- Three types of nonlinearities can cause problems:
  - Non-monotonicity
  - Saturation
  - Interactions between features

Non-monotonicity: The Issue

- Feature Map: $\phi(x) = [1, \text{temperature}(x)]$

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

- Hypothesis Space $\mathcal{F} = \{\text{affine functions of temperature}\}$

- Issue:
    - Health is not an affine function of temperature.
    - Affine function can either say
        - Very high is bad and very low is good, or
        - Very low is bad and very high is good,
        - But here, 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 "normal" temperature in Celsius.

- Ok, but requires manually-specified domain knowledge
  - Do we really need that?
  - What does $w^T \phi(x)$ look like?

- Think less, put in more:

$$\phi(x) = \left[1, \text{temperature}(x), \{\text{temperature}(x)\}^2\right].$$

- More expressive than Solution 1.

General Rule

Features should be simple building blocks that can be pieced together.

Saturation: The Issue

- Setting: Find products relevant to user's query

- Input: Product x

- Action: Score the relevance of x to user's query

- Feature Map:

$$\phi(x) = [1, N(x)],$$

where $N(x) =$ number of people who bought x.

- We expect a monotonic relationship between $N(x)$ and relevance, but also expect diminishing return.

Saturation: Solve with nonlinear transform

- Smooth nonlinear transformation:

$$\phi(x) = [1, \log\{1 + N(x)\}]$$

  - $\log(\cdot)$ good for values with large dynamic ranges

- Discretization (a discontinuous transformation):

$$\phi(x) = (1(0 \leqslant N(x) < 10), 1(10 \leqslant N(x) < 100), \ldots)$$

  - Small buckets allow quite flexible relationship

Interactions: The Issue

- Input: Patient information x

- Action: Health score $y \in \mathsf{R}$ (higher is better)

- Feature Map

$$\phi(x) = [\text{height}(x), \text{weight}(x)]$$

- Issue: It's the weight *relative* to the height that's important.

- Impossible to get with these features and a linear classifier.

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

Interactions: Approach 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(h) and ideal weight(w)

$$f(x) = (52 + 1.9\,[h(x) - 60] - w(x))^2$$

- WolframAlpha for complicated Mathematics:

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

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

Interactions: Approach 2

- Just include all second order features:

$$\phi(x) = \left[1, h(x), w(x), h(x)^2, w(x)^2, \underbrace{h(x)w(x)}_{\text{cross term}}\right]$$

- More flexible, no Google, no WolframAlpha.

General Principle

Simpler building blocks replace a single "smart" feature.

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

Monomial Interaction Terms

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

- Suppose we start with $x = (1, x_1, \ldots, x_d) \in \mathsf{R}^{d+1} = \mathcal{X}$.

- Consider adding all **monomials** of degree M: $x_1^{p_1} \cdots x_d^{p_d}$, with $p_1 + \cdots + p_d = M$.
  - Monomials with degree 2 in 2D space: 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
  - For $d = 40$ and $M = 8$, we get 314457495 features.

Big Feature Spaces

Very large feature spaces have two potential issues:

- Overfitting

- Memory and computational costs

Solutions:

- Overfitting we handle with regularization.

- **Kernel methods** can help with memory and computational costs when we go to high (or infinite) dimensional spaces.