Features

David Rosenberg\textsuperscript{1}

New York University

October 29, 2016

\textsuperscript{1}Thanks to Percy Liang for much of the content.
Feature Extraction
Feature Extraction

The Input Space $\mathcal{X}$

- Our general learning theory setup: no assumptions about $\mathcal{X}$
- But $\mathcal{X} = \mathbb{R}^d$ for the specific methods we’ve developed:
  - Ridge regression
  - Lasso regression
  - Linear SVM
The Input Space $X$

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

- But everything in a computer is a sequence of numbers?
  - The $i$th entry of each sequence should have the same “meaning”
  - All the sequences should have the same length
Feature Extraction

Definition

Mapping an input from $\mathcal{X}$ to a vector in $\mathbb{R}^d$ is called **feature extraction** or **featurization**.
Feature Templates
Example: Detecting Email Addresses

- Task: Predict whether a string is an email address
- Could use domain knowledge and write down:

```
abc@gmail.com
```

```
feature extractor

arbitrary!
```

```
length > 10 : 1
fracOfAlpha : 0.85
contains_@ : 1
endsWith_com : 1
endsWith_org : 0
```

- But this was ad-hoc, and maybe we missed something.
- Could be more systematic?
Feature Templates

Definition (informal)
A **feature template** is a group of features all computed in a similar way.

- Input: *abc@gmail.com*

- Length greater than ___
- Last three characters equal ___
- Contains character ___

---

Based on Percy Liang’s "Lecture 3" slides from Stanford’s CS221, Autumn 2014.
Feature Template: Last Three Characters Equal ___ ___ ___

- Don’t think about which 3-letter suffixes are meaningful...
- Just include them all.

abc@gmail.com

endsWith_aaa : 0
endsWith_aab : 0
endsWith_aac : 0
...
endsWith_com : 1
...
endsWith_zzz : 0

- With regularization, our methods will not be overwhelmed.

From Percy Liang’s "Lecture 3" slides from Stanford’s CS212, Autumn 2014.
Feature Template: One-Hot Encoding

Definition

A one-hot encoding is a set of features (e.g. a feature template) that always has exactly one non-zero value.

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

Array representation (good for dense features):

\[[0.85, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]\]

Map representation (good for sparse features):

\{
"fracOfAlpha": 0.85, 
"contains_@": 1
\}
Feature Vector Representations

- Arrays
  - assumed fixed ordering of the features
  - appropriate when significant number of nonzero elements ("dense feature vectors")
  - very efficient in space and speed (and you can take advantage of GPUs)

- Map (a “dict” in Python)
  - best for sparse feature vectors (i.e. few nonzero features)
  - features not in the map have default value of zero
  - Python code for “ends with last 3 characters”:
    
    ```python
    "endsWith " + x[-3:]: 1.
    ```
  - Has overhead compared to arrays, so much slower for dense features
Handling with 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...

From Percy Liang’s "Lecture 3" slides from Stanford’s CS221, Autumn 2014.
Feature Issues for Linear Predictors

For linear predictors, it’s important **how** features are added.

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.
Non-monotonicity: Solution 1

- Transform the input:

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

where 37 is “normal” temperature in Celsius.

- Ok, but requires manually-specified domain knowledge
  
  - Do we really need that?
Non-monotonicity: Solution 2

- 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.

From Percy Liang’s "Lecture 3" slides from Stanford’s CS221, Autumn 2014.
Handling with Nonlinearity with Linear Methods

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:
  \[ \phi(x) = [1, \mathcal{N}(x)] , \]
  where $\mathcal{N}(x) =$ number of people who bought $x$.
- We expect a monotonic relationship between $\mathcal{N}(x)$ and relevance, but...

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

Is relevance linear in $N(x)$?

- Relevance score reflects confidence in relevance prediction.
- Are we 10 times more confident if $N(x) = 1000$ vs $N(x) = 100$?

- Bigger is better... but not that much better.
Handling with Nonlinearity with Linear Methods

Saturation: Solve with nonlinear transform

- Smooth nonlinear transformation:
  \[ \phi(x) = [1, \log\{1 + N(x)\}] \]

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

- *Does it matter what base we use in the log?*

---

From Percy Liang’s "Lecture 3" slides from Stanford’s CS221, Autumn 2014.
Saturation: Solve by discretization

- Discretization (a discontinuous transformation):

\[ \phi(x) = (1(0 \leq N(x) < 10), 1(10 \leq N(x) < 100), \ldots) \]

- Small buckets allow quite flexible relationship
Interactions: The Issue

- Input: Patient information $x$
- Action: Health score $y \in \mathbb{R}$ (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.

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

$$\phi(x) = \begin{bmatrix} 1, h(x), w(x), h(x)^2, w(x)^2, h(x)w(x) \end{bmatrix}$$

- More flexible, no Google, no WolframAlpha.

General Principle

Simpler building blocks replace a single “smart” feature.
Predicate Features and Interaction Terms

Definition

A predicate on the input space $\mathcal{X}$ is a function $P : \mathcal{X} \rightarrow \{\text{True, False}\}$.

- Many features take this form:
  - $x \mapsto s(x) = 1$ (subject is sleeping)
  - $x \mapsto d(x) = 1$ (subject is driving)

- For predicates, interaction terms correspond to **AND** conjunctions:
  - $x \mapsto s(x)d(x) = 1$ (subject is sleeping AND subject is driving)

From Percy Liang's "Lecture 3" slides from Stanford’s CS221, Autumn 2014.
So What’s Linear?

- Non-linear feature map $\phi : \mathcal{X} \rightarrow \mathbb{R}^d$
- Hypothesis space:
  \[ \mathcal{F} = \{ f(x) = w^T \phi(x) \mid w \in \mathbb{R}^d \} . \]
- Linear in $w$? Yes.
- Linear in $\phi(x)$? Yes.
- Linear in $x$? No.
  - Linearity not even defined unless $\mathcal{X}$ is a vector space

Key Idea: Non-Linearity

- Nonlinear $f(x)$ is important for **expressivity**.
- $f(x)$ linear in $w$ and $\phi(x)$: makes finding $f^*(x)$ much easier

From Percy Liang’s "Lecture 3" slides from Stanford’s CS221, Autumn 2014.
Geometric Example: Two class problem, nonlinear boundary

- With linear feature map $\phi(x) = (x_1, x_2)$ and linear models, no hope
- With appropriate nonlinearity $\phi(x) = (x_1, x_2, x_1^2 + x_2^2)$, piece of cake.

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

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.
Consider a linear hypothesis space with a feature map $\phi : \mathcal{X} \rightarrow \mathbb{R}^d$:

$$\mathcal{F} = \{ f(x) = w^T \phi(x) \}$$

**Question:** does $\mathcal{F}$ contain a good predictor?