The unreasonable effectiveness of feature hashing

Gianluca Campanella [dʒanˈluːka kampaˈnɛlla]

14th July 2019

What I do nowadays

I'm a Data Scientist at

Microsoft

in **AzureCAT**

What I do nowadays

I also run my own company

Estimand

that provides

Data Science consulting and training

Contents

Background

Feature hashing

Library support

Acknowledgements

- Sharat Chikkerur
- Markus Cozowicz

Background

Supervised ML

- Observe $\mathbf{X}_{n \times p}$ and $\mathbf{y}_n$ (typically $n \gg p$)
- Find the 'best' estimator of some statistic of $\mathbf{y}$ given $\mathbf{X}$ (for example $\mathbb{E}[\mathbf{y}|\mathbf{X}]$)

Supervised ML

- Observe $\mathbf{X}_{n \times p}$ and $\mathbf{y}_n$ (typically $n \gg p$)
- Find the 'best' estimator of some statistic of $\mathbf{y}$ given $\mathbf{X}$ (for example $\mathbb{E}[\mathbf{y}|\mathbf{X}]$)

Generalised linear models (GLMs)

- $\mathbb{E}[\mathbf{y}|\mathbf{X}] = g^{-1}(\mathbf{X}\beta)$ with g given
- Find the 'best' estimates for β

Features are often categorical

- Discretised continuous features
- One-hot encoding
- Bag-of-words representation

One-hot encoding

country
England
Wales
Scotland
Scotland
Northern Ireland

$$\rightarrow \begin{array}{cccc} \texttt{ENG} & \texttt{NIR} & \texttt{SCT} & \texttt{WLS} \\ \end{array}$$
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Bag-of-words representation

document
Ashley loves cats. She also likes dogs.
Barbara hates cats but she loves dogs.
Carol loves cats and dogs.

$$\downarrow$$

$$\begin{array}{ccccc} \texttt{cat} & \texttt{dog} & \texttt{hate} & \texttt{like} & \texttt{love} \\ \end{array}$$
$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

High-dimensional feature space

What's going on?

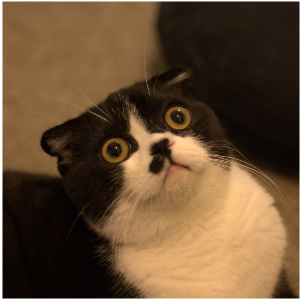
- n is fixed
- p depends on the cardinality of features

High-dimensional feature space

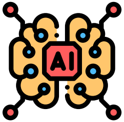
Why is this a problem?

- Online learning
- $p \to n$ breaks GLMs (though regularisation helps)

High-dimensional feature space



$\rightarrow$



$\rightarrow$

$$\begin{array}{c} \begin{matrix} \text{cat} & \text{dog} \end{matrix} \\ \begin{bmatrix} 0.99 & 0.01 \end{bmatrix} \end{array}$$

High-dimensional feature space

Observations in X are high-dimensional but sparse

- Ad tech
- E-commerce
- Social networks

High-dimensional feature space

Can we find a 'smaller' X?

- Time- and space-efficient to compute
- Fixed p
- Sparse

Recap

- Categorical and textual features are common
  - $\rightarrow$ $p \propto$ feature cardinality (often very large)
  - $\rightarrow$ Sparse $\mathbf{X}$
- Want an efficiently computed, fixed-p, sparse representation

Feature hashing

Feature hashing

country = ‘England’

$\downarrow$

Hash value

$h(\texttt{country=England}) = 1462978411$

$\downarrow$

Feature index

$i = h(\texttt{country=England}) \bmod 100 = 11$

Feature hashing

country
England
Wales
Scotland
Scotland
Northern Ireland

$$\rightarrow \begin{array}{c} i \\ \begin{bmatrix} 11 \\ 62 \\ 54 \\ 54 \\ 8 \end{bmatrix} \end{array} \rightarrow \begin{array}{c} \begin{array}{ccccccccc} \dots & \mathbf{x}_8 & \dots & \mathbf{x}_{11} & \dots & \mathbf{x}_{54} & \dots & \mathbf{x}_{62} & \dots \end{array} \\ \begin{bmatrix} & & & 1 & & & & & \\ & & & & & & & 1 & \\ & & & & & 1 & & & \\ & & & & & 1 & & & \\ & 1 & & & & & & & \end{bmatrix} \end{array}$$

Feature hashing

- Time- (`%timeit`: 468 ns ± 7.2 ns) and space-efficient
- Fixed p
- Sparse

Hash function

h maps any integer input onto integers in some range (e.g. `int32`)

Important properties

- Uniform output
- Avalanche effect
  - $h(\mathrm{a}) \rightarrow 3001393763$
  - $h(\mathrm{c}) \rightarrow 1701913768$

Feature hashing in Python

```python
import mmh3

s = "country=England"

h = mmh3.hash(s, seed=42, signed=False)
print(h)  # => 1462978411

i = h % 100
print(i)  # => 11
```

Hashing of Unicode strings

- Strings are just (very large) integers
- Be careful when handling Unicode

```
s1 = "sch\u00f6n"
s2 = "scho\u0308n"

print(s1)          # => schön
print(s2)          # => schön
print(s1 == s2)    # => False
```

Unicode normalisation

```python
import unicodedata

s1 = unicodedata.normalize("NFKD", "sch\u00f6n")
s2 = unicodedata.normalize("NFKD", "scho\u0308n")

print(s1)          # => schön
print(s2)          # => schön
print(s1 == s2)    # => True
```

Projection

- The modulo operator is expensive
- If the hash size is a power of two, we can do better

```
%timeit h % 128
# => 81.3 ns ± 1.92 ns per loop

%timeit h & 127
# => 56.9 ns ± 1.4 ns per loop
```

Collisions

```python
def h(x):
    return mmh3.hash(x, seed=42, signed=False) & 127

print(h("country=United Kingdom"))  # => 6
print(h("country=Bulgaria"))        # => 6
```

- Smaller hash size → more collisions
- Impact on statistical performance and interpretability

Sign function ξ

- Use another function ξ to determine the sign
- Collisions cancel out (in expectation)

```python
def h(x):
    hash_ = mmh3.hash(x, seed=42, signed=True)
    return abs(hash_) & 127, (hash_ >= 0) * 2 - 1

print(h("country=United Kingdom"))  # => (  6,  1)
print(h("country=Bulgaria"))        # => (122, -1)
```

Recap

- h maps any integer input onto integers in some range
- h deterministically scrambles arbitrary-length bitmaps, producing fixed-length bitmaps
- The impact of collisions on statistical performance is small

Feature hashing

Example

Original data

user	movie	has_cats	has_dogs	rating
Ashley	A Dog's Journey	0	1	4
Ashley	Catwoman	1	0	5
Ashley	The Aristocats	1	1	5
Ashley	The Queen's Corgi	0	1	3
Barbara	A Dog's Journey	0	1	5
Barbara	The Aristocats	1	1	2
Barbara	The Queen's Corgi	0	1	5
Carol	Catwoman	1	0	5
Carol	The Aristocats	1	1	5
Carol	The Queen's Corgi	0	1	4

After feature hashing

user	movie	has_cats	has_dogs
2^-	5^-		110^-
2^-	1^-	9^+	
2^-	104^-	9^+	110^-
2^-	67^+		110^-
19^+	5^-		110^-
19^+	104^-	9^+	110^-
19^+	67^+		110^-
77^+	1^-	9^+	
77^+	104^-	9^+	110^-
77^+	67^+		110^-

After feature hashing

$$
\begin{array}{c}
\begin{array}{ccccccccccccccccccc}
\cdots & \mathbf{x}_1 & \cdots & \mathbf{x}_2 & \cdots & \mathbf{x}_5 & \cdots & \mathbf{x}_9 & \cdots & \mathbf{x}_{19} & \cdots & \mathbf{x}_{67} & \cdots & \mathbf{x}_{77} & \cdots & \mathbf{x}_{104} & \cdots & \mathbf{x}_{110} & \cdots
\end{array} \\
\begin{bmatrix}
 & & & -1 & & -1 & & & & & & & & & & & & -1 & \\
 & -1 & & -1 & & & & 1 & & & & & & & & & & & \\
 & & & -1 & & & & 1 & & & & & & & & -1 & & -1 & \\
 & & & -1 & & & & & & & & 1 & & & & & & -1 & \\
 & & & & & -1 & & & & 1 & & & & & & & & -1 & \\
 & & & & & & & 1 & & 1 & & & & & & -1 & & -1 & \\
 & & & & & & & & & 1 & & 1 & & & & & & -1 & \\
 & -1 & & & & & & 1 & & & & & & 1 & & & & & \\
 & & & & & & & 1 & & & & & & 1 & & -1 & & -1 & \\
 & & & & & & & & & & & 1 & & 1 & & & & -1 &
\end{bmatrix}
\end{array}
$$

Model

$$\texttt{rating} \sim \texttt{user} + \texttt{movie} + \texttt{has\_cats} + \texttt{has\_dogs}$$

- Average effect for users
- Average effect for movies
- Average effect for movie-related features

Model

$$\texttt{rating} \sim \texttt{user} + \texttt{movie} + \texttt{has\_cats} + \texttt{has\_dogs}$$

- Average effect for users
- Average effect for movies
- Average effect for movie-related features

Model

$$\texttt{rating} \sim \texttt{user} + \texttt{movie} + \texttt{has\_cats} + \texttt{has\_dogs}$$

- Average effect for users
- Average effect for movies
- Average effect for movie-related features

Model

$$\texttt{rating} \sim \texttt{user} + \texttt{movie} + \texttt{has\_cats} + \texttt{has\_dogs}$$

- Average effect for users
- Average effect for movies
- Average effect for movie-related features

Interactions

$$\texttt{rating} \sim \texttt{user} + \texttt{movie} + \texttt{user} \times (\texttt{has\_cats} + \texttt{has\_dogs})$$

- Average effect for users
- Average effect for movies
- Average effect for movie-related features
- User correction

Interactions with feature hashing

$$h(\texttt{user=Ashley}) = 2^- \qquad h(\texttt{has\_cats}) = 9^+$$

$$\searrow \qquad\qquad \swarrow$$

```python
def interact(h1, h2):
    i1, s1 = h1
    i2, s2 = h2
    return ((i1 ^ i2) * 16777619) & 127, s1 * s2
```

$$\downarrow$$

$$h(\texttt{user=Ashley} \times \texttt{has\_cats}) = 81^-$$

After feature hashing with interactions

user	movie	has_cats	has_dogs	likes_cats	likes_dogs
2^-	5^-		110^-		4^+
2^-	1^-	9^+		81^-	
2^-	104^-	9^+	110^-	81^-	4^+
2^-	67^+		110^-		4^+
19^+	5^-		110^-		71^-
19^+	104^-	9^+	110^-	110^+	71^-
19^+	67^+		110^-		71^-
77^+	1^-	9^+		12^+	
77^+	104^-	9^+	110^-	12^+	25^-
77^+	67^+		110^-		25^-

After feature hashing with interactions

$$
\begin{array}{c}
\begin{array}{cccccccccccccc}
\mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_4 & \mathbf{x}_5 & \mathbf{x}_9 & \mathbf{x}_{12} & \mathbf{x}_{19} & \mathbf{x}_{25} & \mathbf{x}_{67} & \mathbf{x}_{71} & \mathbf{x}_{77} & \mathbf{x}_{81} & \mathbf{x}_{104} & \mathbf{x}_{110}
\end{array}\\
\left[\begin{array}{cccccccccccccc}
 & -1 & 1 & -1 & & & & & & & & & & -1 \\
-1 & -1 & & & 1 & & & & & & & -1 & & \\
 & -1 & 1 & & 1 & & & & & & & -1 & -1 & -1 \\
 & -1 & 1 & & & & & & 1 & & & & & -1 \\
 & & & -1 & & & 1 & & & -1 & & & & -1 \\
 & & & & 1 & & 1 & & & -1 & & & -1 & {\color{orange}0} \\
 & & & & & & 1 & & 1 & -1 & & & & -1 \\
-1 & & & & 1 & 1 & & & & & 1 & & & \\
 & & & & 1 & 1 & & -1 & & & 1 & & -1 & -1 \\
 & & & & & & & -1 & 1 & & 1 & & & -1
\end{array}\right]
\end{array}
$$

Pros and cons

Pros

- Time- and space-efficient $\rightarrow$ online learning
- Sparsity-preserving
- Implicit handling of missing data

Cons

- Collisions $\rightarrow$ statistical performance
- Inverse mapping $\rightarrow$ interpretability

Feature hashing

Use case

Scenario

- Business directory service (think Yelp)
- Want to improve relevance of and personalise search results

Scenario

- Business directory service (think Yelp)
- Want to improve relevance of and personalise search results

CTR optimisation

Maximise probability of click on top results

Personalisation

Take into account past user interactions

Data

user	query	business	click
A	Italian restaurant	Bocca di Lupo	1
A	Italian restaurant	Brasserie Zédel	0
A	Italian restaurant	Emilia's Crafted Pasta	0
A	Italian restaurant	Fucina	1
A	Italian restaurant	Trullo	0

- 3.5×10^8 observations (250 days)
- 4.5×10^6 businesses with many (sparse) attributes

Model

$$\texttt{click} \sim \texttt{user} + \texttt{query} + \texttt{business} + \\ \texttt{query} \times \texttt{business} + \texttt{user} \times \texttt{business}$$

- Average effects
- Relevance correction
- User correction

Model

$$\texttt{click} \sim \texttt{user} + \texttt{query} + \texttt{business} + \\ \texttt{query} \times \texttt{business} + \texttt{user} \times \texttt{business}$$

- Average effects
- Relevance correction
- User correction

Model

$$\begin{aligned}\texttt{click} \sim \texttt{user} + \texttt{query} + \texttt{business} +\\ \texttt{query} \times \texttt{business} + \texttt{user} \times \texttt{business}\end{aligned}$$

- Average effects
- Relevance correction
- User correction

Model

$$\texttt{click} \sim \texttt{user} + \texttt{query} + \texttt{business} + \texttt{query} \times \texttt{business} + \texttt{user} \times \texttt{business}$$

- Average effects
- Relevance correction
- User correction

Model

- Fixed $p = 2^{23} \approx 8.4 \times 10^6$
- Regularised logistic regression (Spark ML)
- Grid search over elastic net hyperparameters

Results

- Estimated $\approx 5\%$ increase in CTR based on train/test split
- A/B testing showed $\approx 10\%$ increase in CTR

Library support

Vowpal Wabbit (VW)

- C++ with Python bindings
- Linear, logistic and Poisson regression with interactions
- Extremely fast and scalable

scikit-learn

- Some support for feature hashing: `feature_extraction.FeatureHasher` and `feature_extraction.text.HashingVectorizer`
- No interactions

Apache Spark

- Some support for feature hashing in MLlib: `ml.feature.FeatureHasher` and `ml.feature.HashingTF`
- PySpark transformers (hashing and interactions)
- VW bindings coming soon to MMLSpark

Recap

Feature hashing is...

- Great for online training on lots of data
- Well-suited for high-cardinality features (sparse $\mathbf{X}$)
- Even better with interactions

Thank you!

If you want to keep in touch…

@ gianluca@campanella.org gcampanella gcampanella