A short introduction to `function_score`

Britta Weber

elasticsearch

Agenda

PART 1: Text scoring for human beings and the downside for tags

PART 2: Scoring numerical fields

How does scoring of text work?

Relevancy

Step	Query	Doc 1	Doc 2
The text	brown fox	The quick brown fox likes brown nuts	The red fox
The terms	(brown, fox)	(brown, brown, fox, likes, nuts, quick)	(fox, red)
A frequency vector	(1, 1)	(2, 1)	(0, 1)
Relevancy	-	3?	1?

So...more matching words mean higher score, right?

Text scoring oddities

https://gist.github.com/brwe/7229896

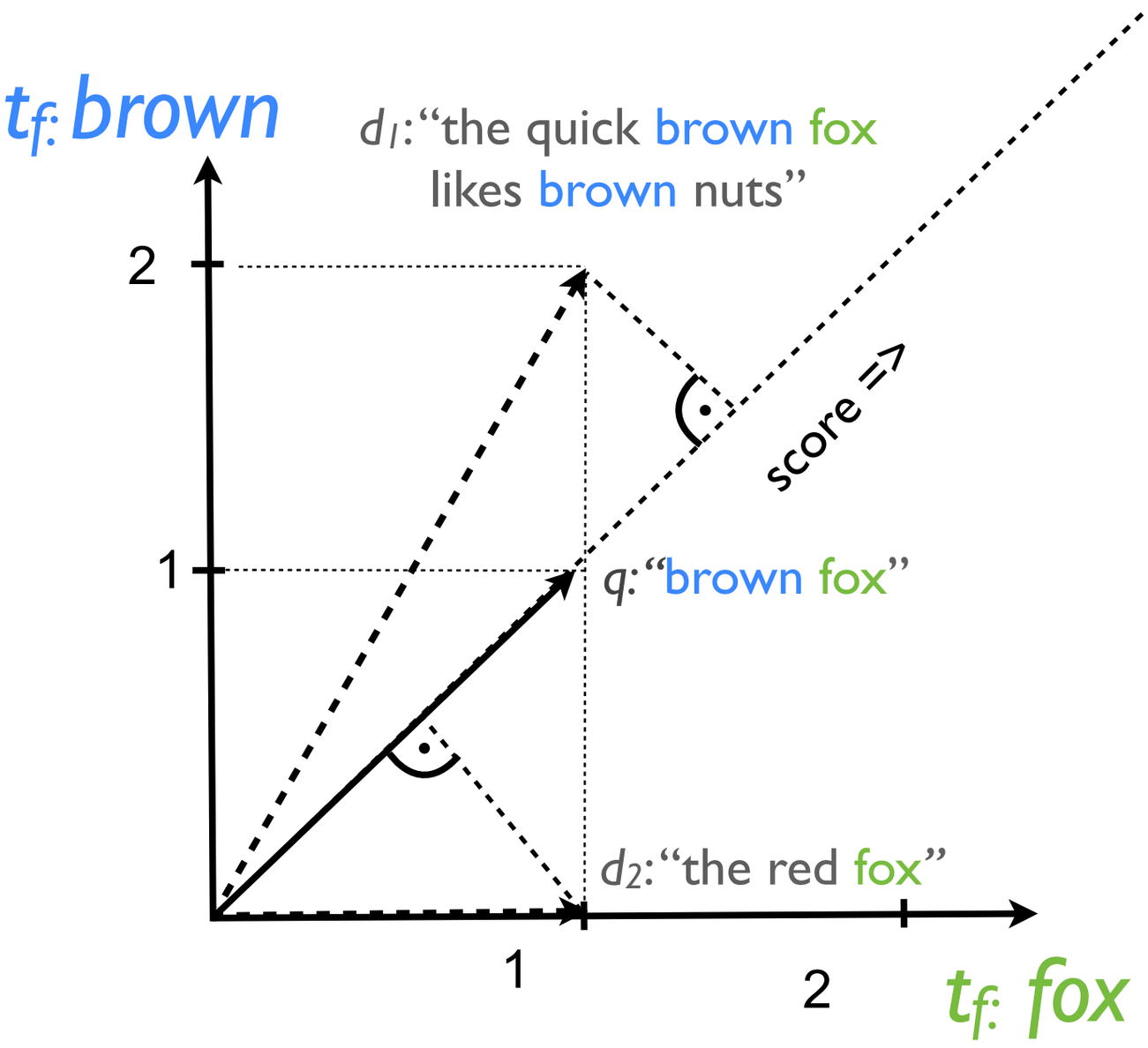
Relevancy

Step	Query	Doc 1	Doc 2
The text	brown fox	The quick brown fox likes brown nuts	The red fox
The terms	(brown, fox)	(brown, brown, fox, likes, nuts, quick)	(fox, red)
A frequency **vector**	(1, 1)	(2, 1)	(0, 1)
Relevancy	-	3?	1?

Relevancy - Vector Space Model

Distance of docs and query:

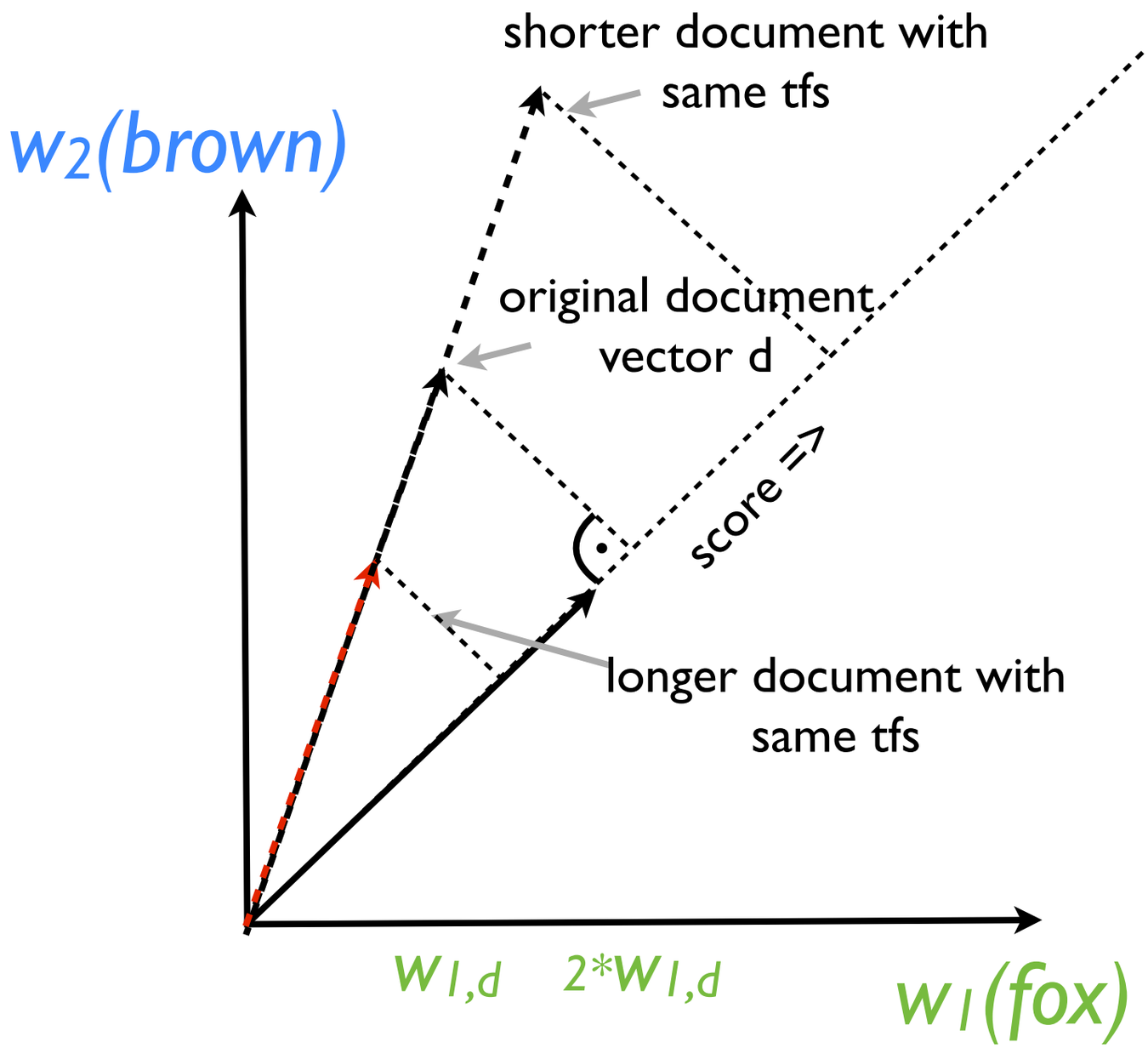
Project document vector on query *axis*.



I: Field length

Shorter text is more relevant than longer text.

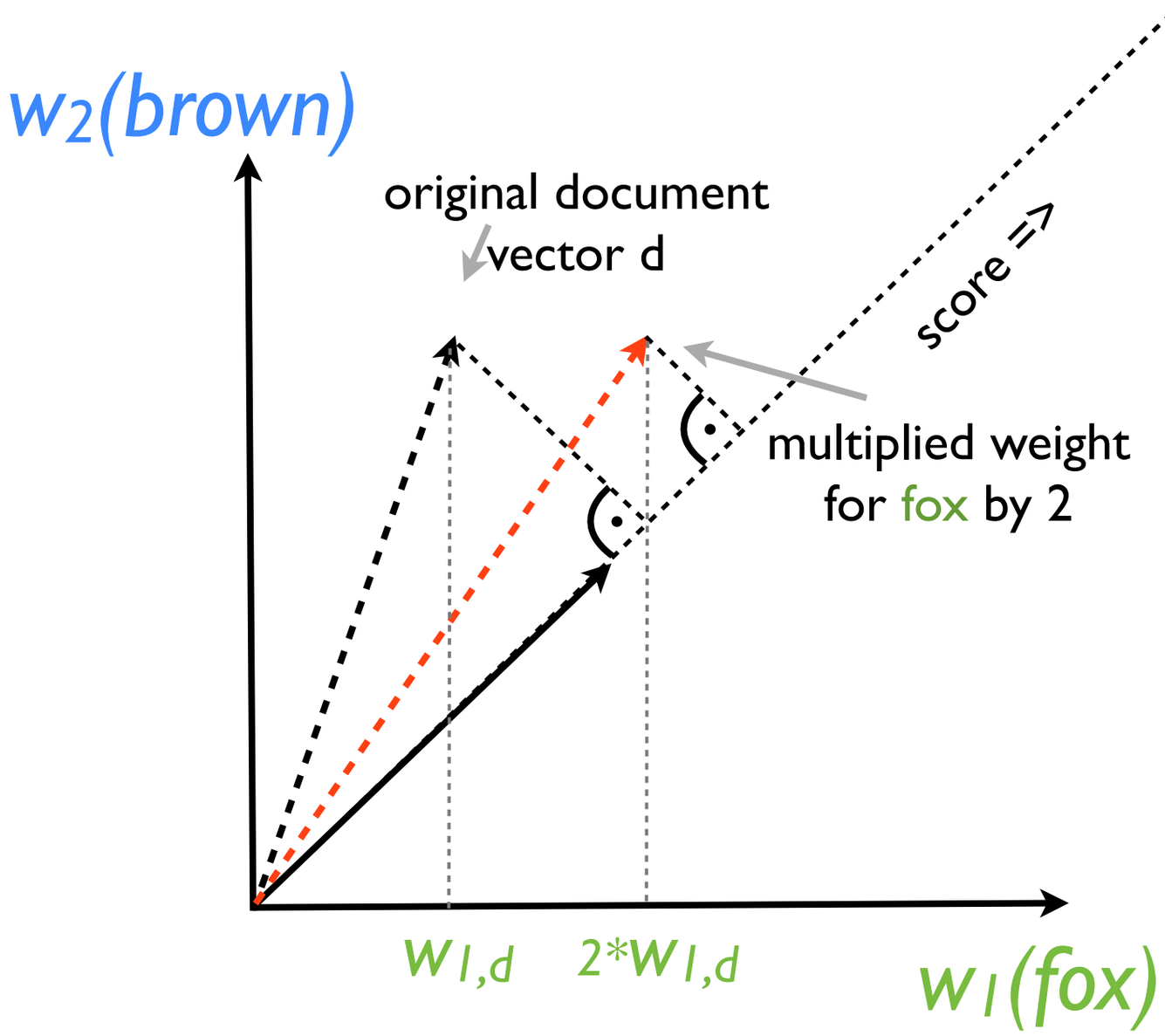
Field length - Vector Space Model



II: Document frequency

Words that appear more often in documents are less important that words that appear less often.

Term weight - Vector Space Model



How many of these factors are there?

Lucene Similarity

inverted document frequency for term *t*

field length, some function turning the number of tokens into a float, roughly:

$$1/\sqrt{\text{num terms in field}}$$

score of a document *d* for a given query *q*

$$\text{score}_{q,d} = \text{norm}(q) \times \sum_{t \text{ in } q} \sqrt{\text{tf}_{t,d}} \times \text{idf}_t^2 \times \text{norm}(d, field) \times \text{boost}(t)$$

$$\text{idf}_f = 1 + \frac{|D|}{1 + |\{d' \in D | t \in d'\}|}$$

query norm, does not fit on this slide

core TF/IDF weight

boost of query term *t*

http://lucene.apache.org/core/4_0_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html

Explain api

If you do not understand the score:

```
curl -XPOST "http://localhost:9200/idfidx/test/_search" -d'
{
    "query": {
        "match": {
            "location": "berlin kreuzberg"
        }
    },
    "explain": true
}'
```

The point is...

- Text scoring per default is tuned for natural language text.
- Empirical scoring formula works well for articles, mails, reviews, etc.
- This way to score might be undesirable if the text represents *tags*.

function_score

- *Tags* should be scored different from *text*
- Numerical field values should result in a score and not only score 0/1
- Sometimes we want to write our own scoring function!

(Disclaimer: Not all of this is new.)

function_score - basic structure

```
"function_score": {
      "(query|filter)": {},
      "functions": [
          {
              "filter": {},
              "FUNCTION": {}
          },
          ...
      ]
}
```

query or filter

Apply score computation only to docs matching a specific filter (default “match_all”)

Apply this function to matching docs

Example for function score

https://gist.github.com/brwe/7049473

Decay Functions

JSON structure

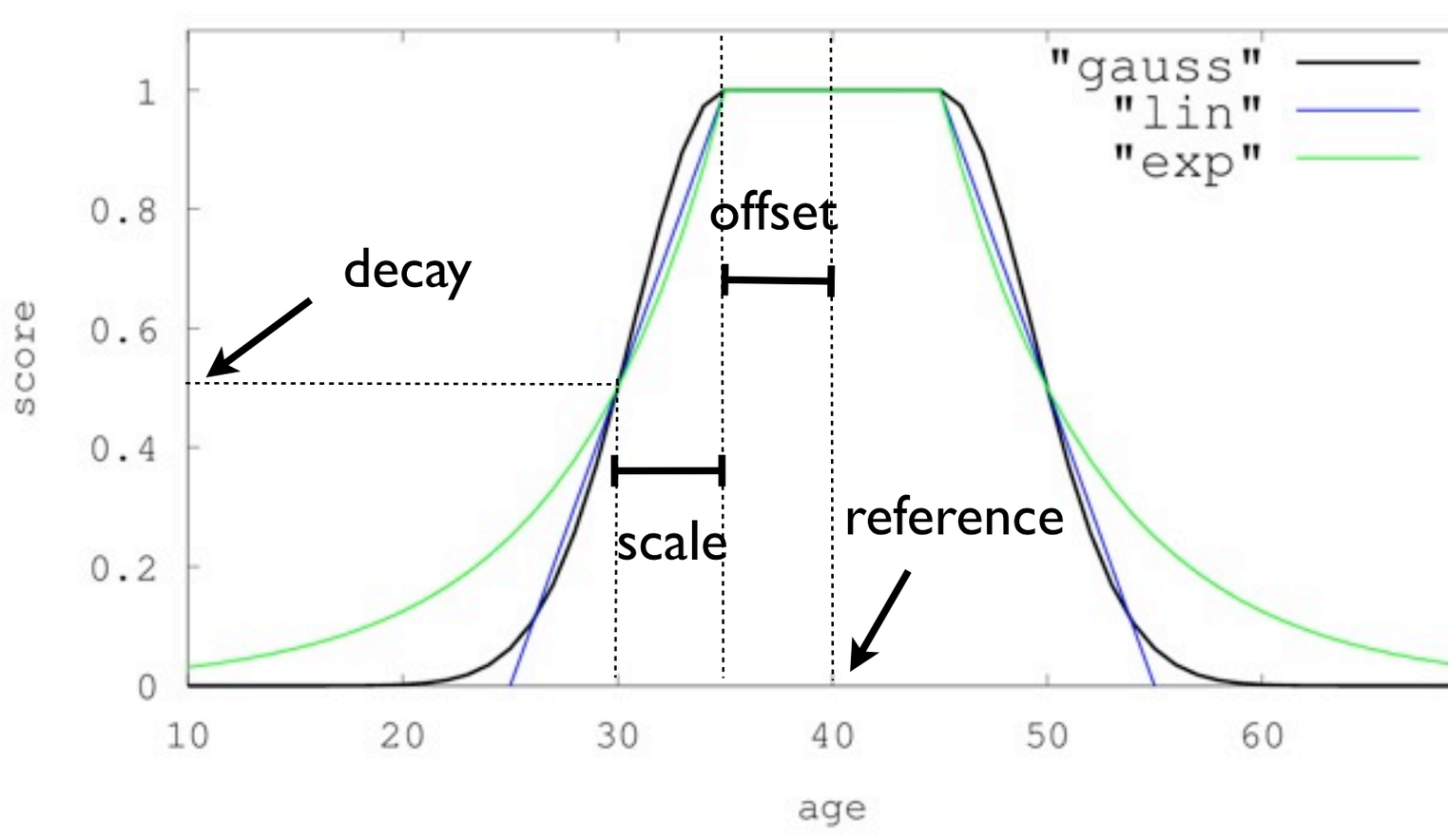
```
"gauss": {
 "age": {
     "reference": 40,
     "scale": 5,
   "decay": 0.5,
   "offset": 5
     }
 }
```

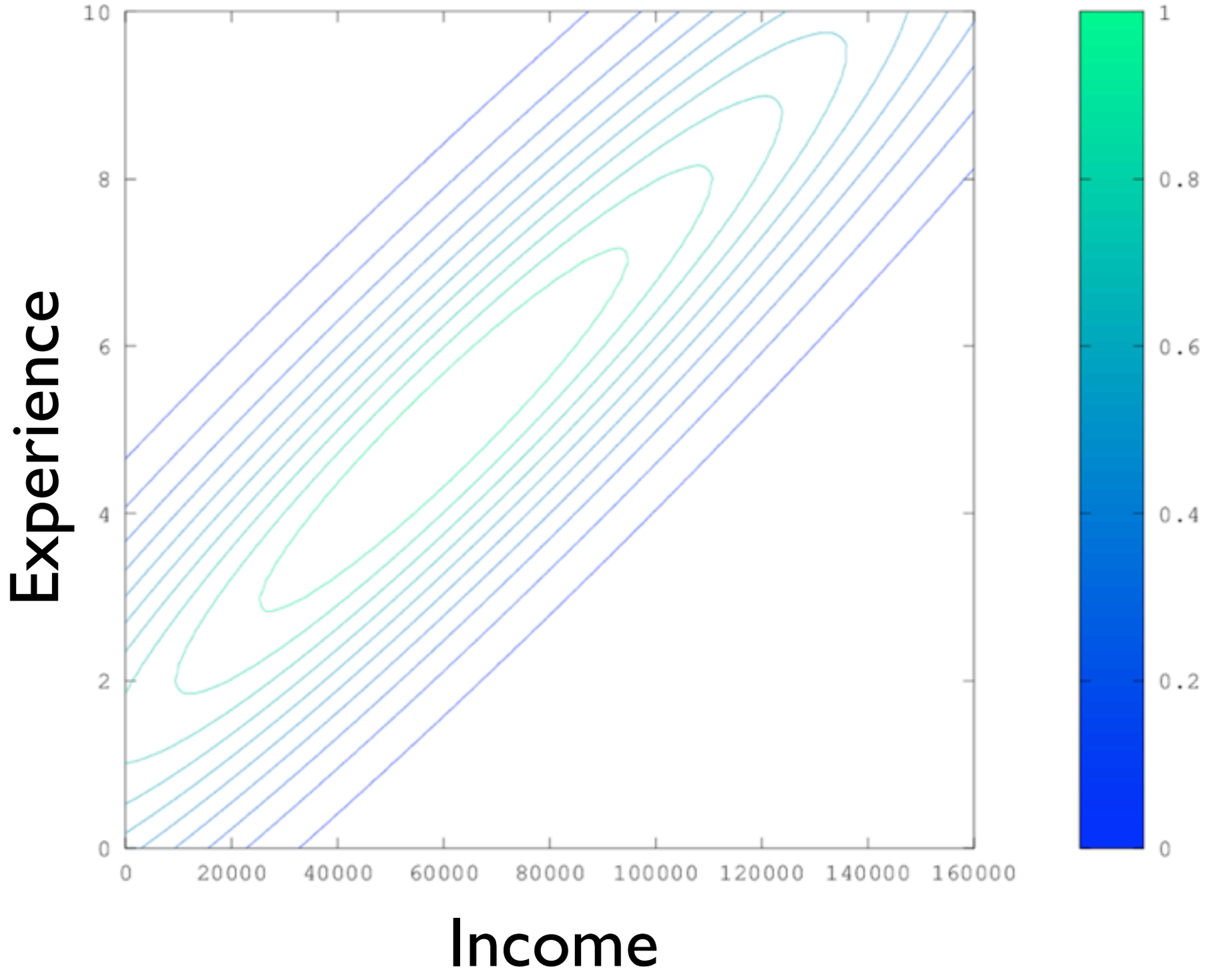
shape of decay curve

field name

Decay functions

- “gauss”
- “exp”
- “lin”




Experience
10
8
6
4
2
0
0
20000
40000
60000
80000
100000
120000
140000
160000
Income
1
0.8
0.6
0.4
0.2
0

function_score - even more parameters!

```
"function_score": {
      "(query|filter)": {},
      "boost": 2,
      "functions": [
          {
              "filter": {},
              "FUNCTION": {}
          },
          ...
      ],
      "max_boost": 10.0,
      "score_mode": "(mult|max|...)",
      "boost_mode": "(mult|replace|...)"
}
```

query score

Apply score computation only to doc matching a specific filter (default "match_all")

Apply this function to matching docs

limit boost to 10

Result of the different filter/ function pairs should be summed, multiplied,....

Merge with query score by multiply, add, ...

The downside

`function_score` functions and their combination can be arbitrarily complex => hard to tune the parameters.

Coming up next in elasticsearch...

Script scoring with term vectors - build your own fancy natural language scoring model!

#3772