

文本表示学习

陆俞因

2023. 04. 25



独热表示

➤ One-hot Representation

- 使用一个V维向量表示一篇文章，向量的长度V为词汇表的大小
- 1表示存在对应的单词，0表示不存在
 - 注意，one-hot表示仅考虑存在性，只取0/1两种值

例：

- 文档1：不错 酒店 舒服 服务 态度 很好
- 文档2：酒店 服务 热情 希望 服务
- 文档3：苹果 手机 不错

[illegible]



词袋模型

➤ Bag-of-Words(BoW)

- 将文档视为**词汇的集合**，即“**词袋**” (Bag-of-Words)
- 则可使用一个V维向量表示一篇文档，其中每一维的值对应词表的位置上该词语出现的**次数**
 - 注意与one-hot表示文档的区别

例：

- 文档1：不错 酒店 舒服 服务 态度 很好
- 文档2：酒店 服务 热情 希望 服务
- 文档3：苹果 手机 不错

[illegible]



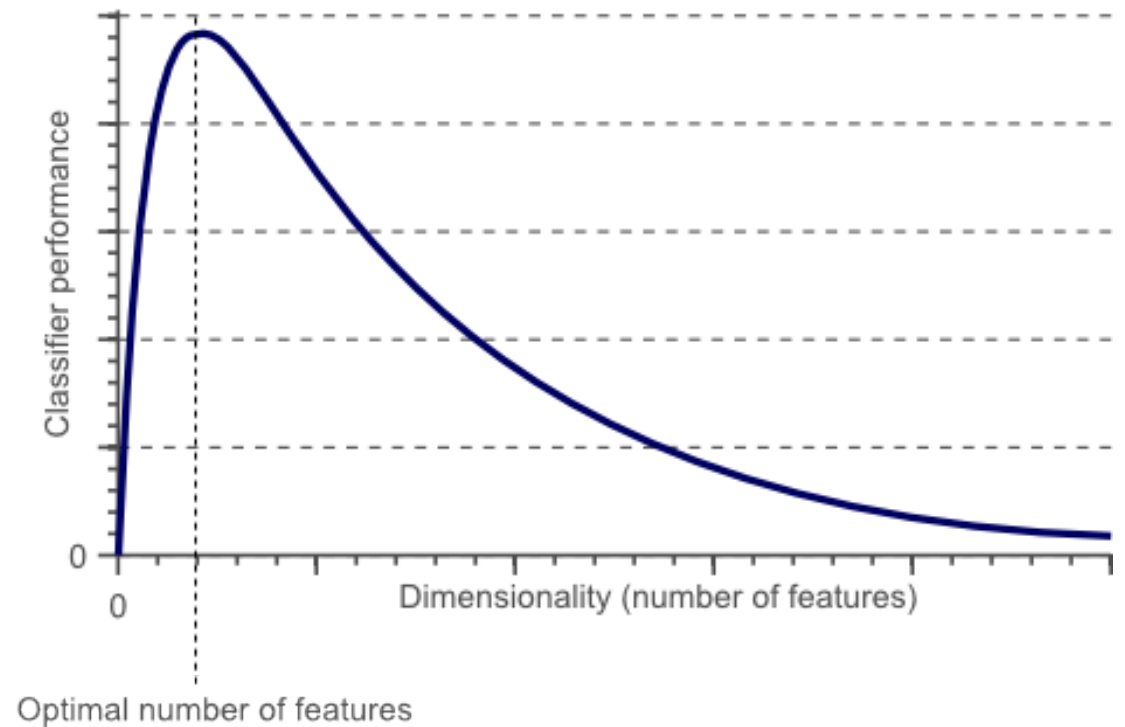
维度灾难

➤ Curse of Dimensionality

假设我们要对猫 (🐱) 和狗 (🐶) 的图片进行分类，为此构建出一些特征，例如：

- R通道
- G通道
- B通道
-

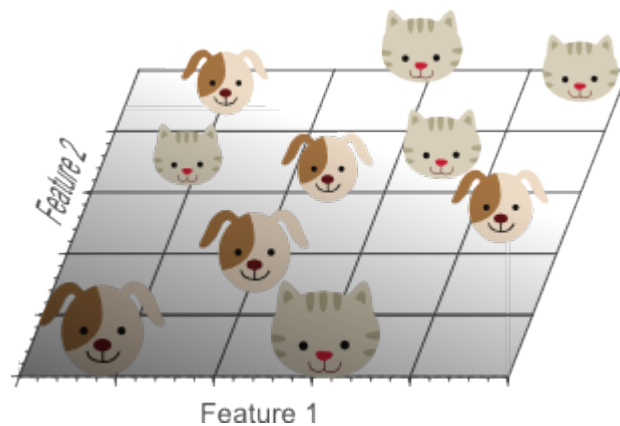
随着特征数的增加，分类器的性能先增强后减弱



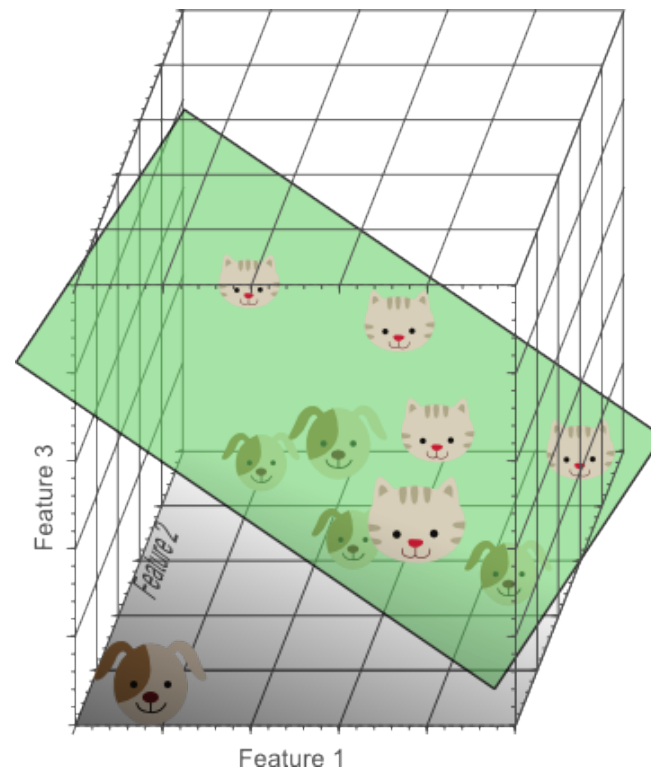
假设我们有10个训练样本，即10张猫和狗的图片



单个特征对训练样本分类效果不佳



增加第二个特征仍然不能线性分割，即不存在一条直线能够将猫和狗完全分开

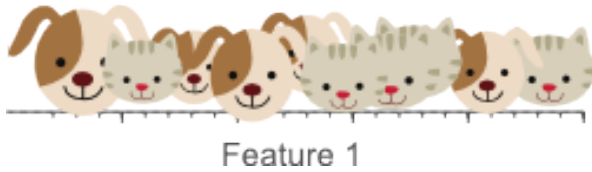


增加第三个特征实现了线性可分，即存在一个平面完全将猫和狗分离开

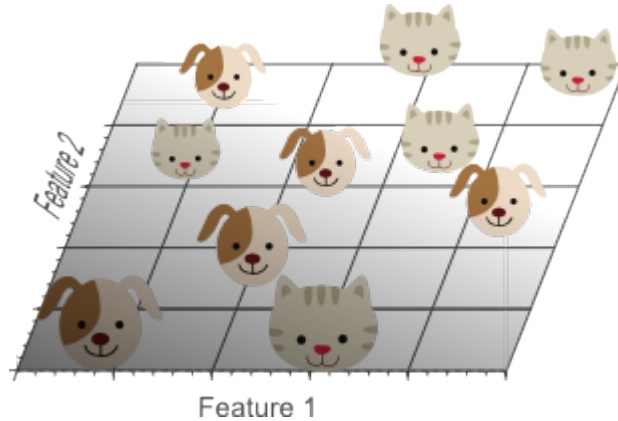
➤为什么增加特征能够使分类器性能增强？

假设特征空间在每个维度上的宽度为5

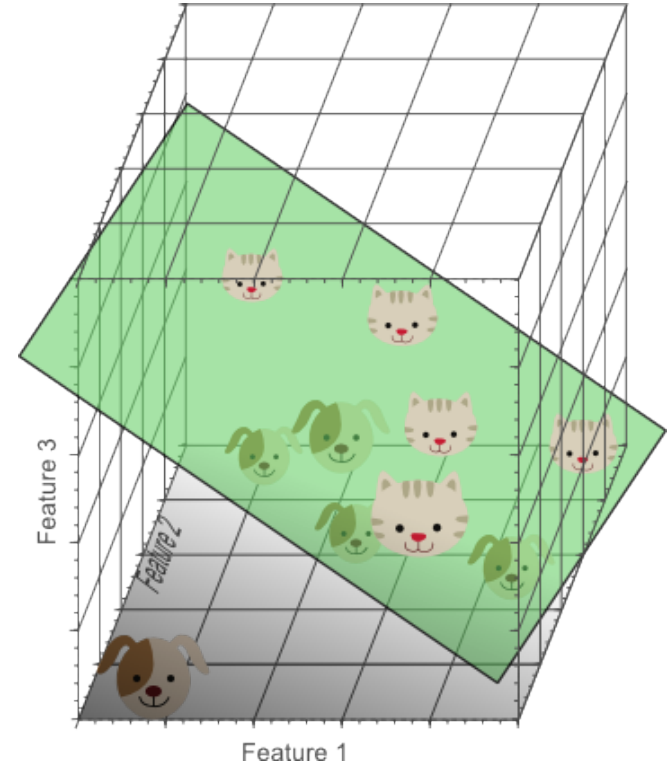
对于10个样本，特征空间中的样本密度分别是：



$$10 / 5 = 2$$



$$10 / (5 * 5) = 0.4$$

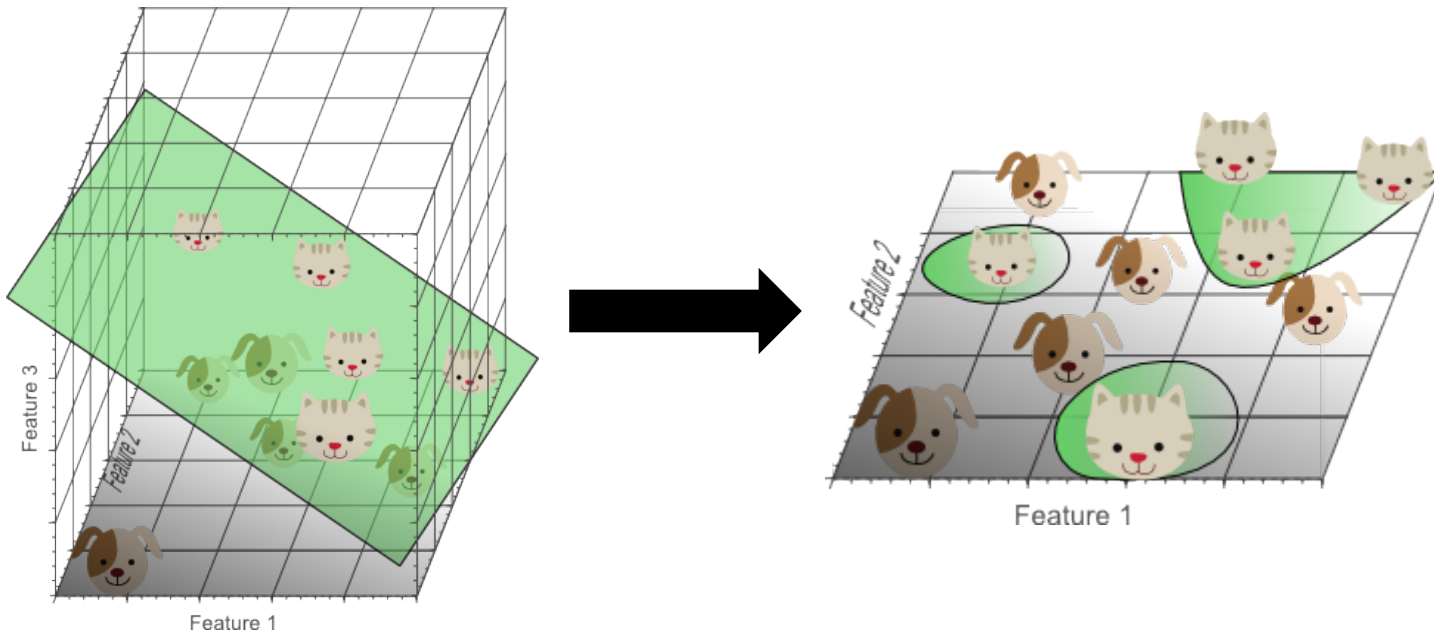


$$10 / (5 * 5 * 5) = 0.08$$

随着特征数的增加，特征空间越来越稀疏，因此容易找到一个“完美区分”训练样本的超平面

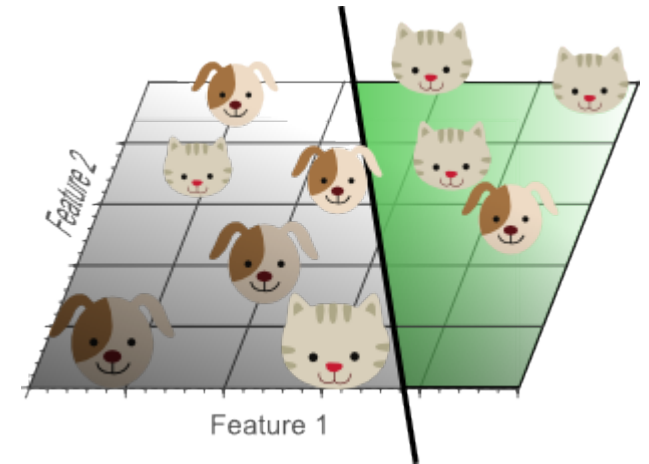
➤大量特征会导致过拟合

将3D的分类结果投影到2D特征空间



分类器学习了训练数据的噪声和异常，而对样本外的数据拟合效果并不理想，甚至很差

2D特征空间中的一个线性分类器



尽管训练样本不能全都分类正确，但这个分类器的泛化能力更好

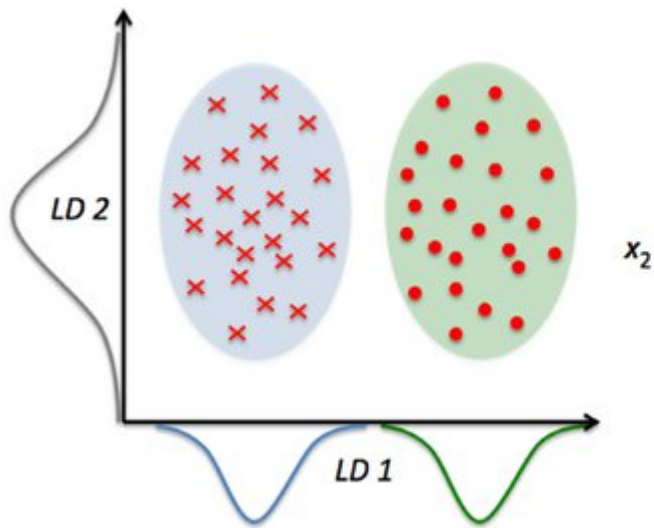


数据降维

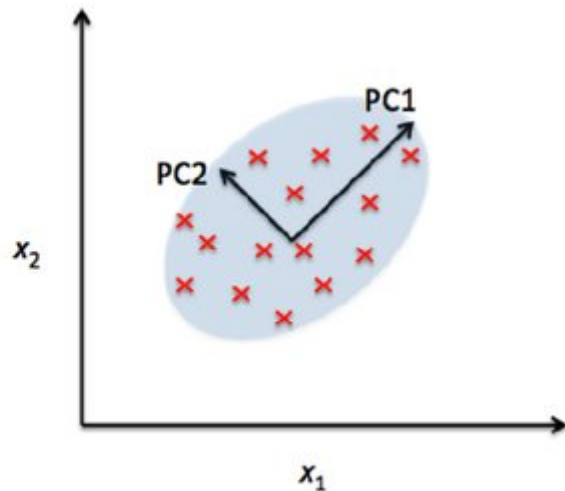
➤ Dimensionality reduction

- 特征选择 (feature selection)

选出具有代表性的特征子集



✓ 线性判别分析 (LDA)



✓ 主成分分析 (PCA)

- 特征投影 (feature projection)

将高维特征空间中的样本投影到一个低维特征空间。例如：

- ✓ 主成分分析 (PCA)

- ✓ 线性判别分析 (LDA)

- ✓ Word2Vec

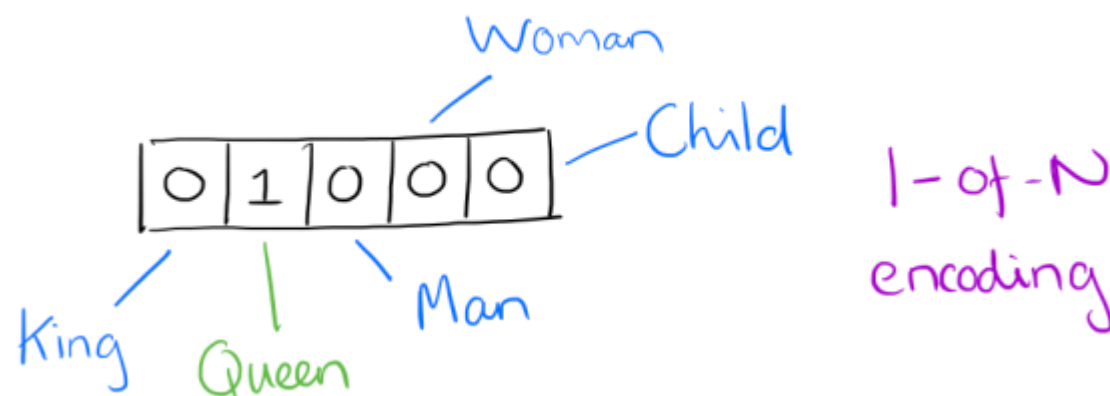
- ✓



词独热表示

➤ One-hot Representation

长度为（不重复）词表大小，每一维表示一个单词





Word2Vec

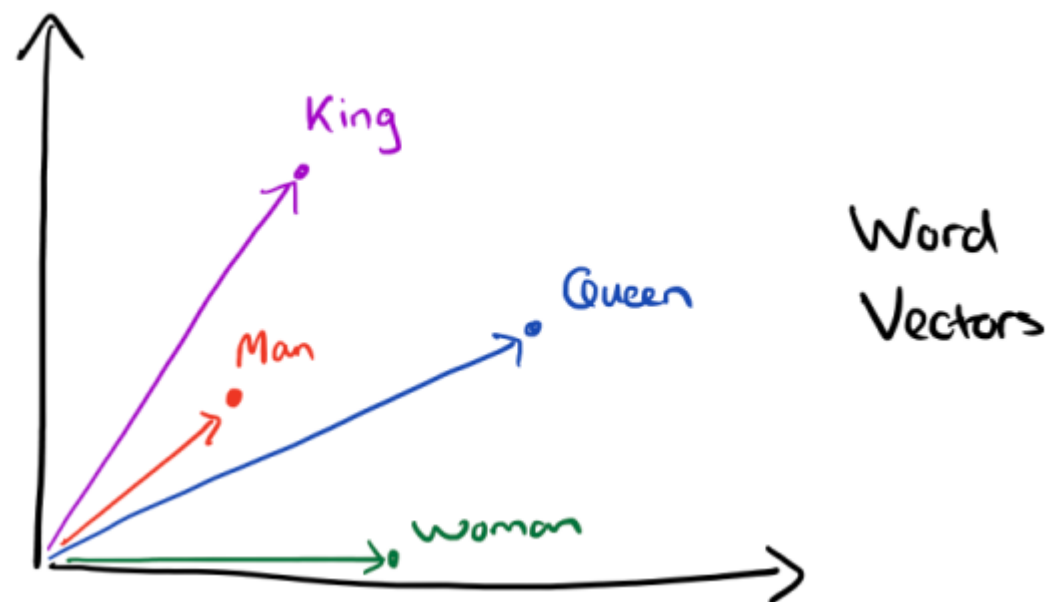
➤ Word Embedding 词嵌入

将高维稀疏的词独热表示映射为低维紧密的向量表示

从几何角度理解,

- ✓ 每个单词映射为低维语义空间中的一个点,
词嵌入即点坐标
- ✓ 可以用点之间的距离表示单词之间的语义相似性

	King	Queen	Woman	Princess
Royalty	0.99	0.99	0.02	0.98
Masculinity	0.99	0.05	0.01	0.02
Femininity	0.05	0.93	0.999	0.94
Age	0.7	0.6	0.5	0.1
...





分布式假说


➤ Distributional Hypothesis

"a word is characterized by the company it keeps"

“一个单词的上下文决定它的语义”

学习新单词:

I had a delicious **dinner** last night

A diagram consisting of two blue curved arrows. The first arrow starts from the word 'delicious' and points to the word 'dinner'. The second arrow starts from the phrase 'last night' and also points to the word 'dinner', illustrating how the surrounding context helps in learning the meaning of the word.

完形填空:

I had a delicious last night

dinner meal steak



Word2Vec

➤ Word Embedding 词嵌入

一个固定大小的滑动窗口内的单词被定位为上下文词，
进而构建训练样本对 (上下文词 x ，目标词 y)

I had a delicious **dinner** last night

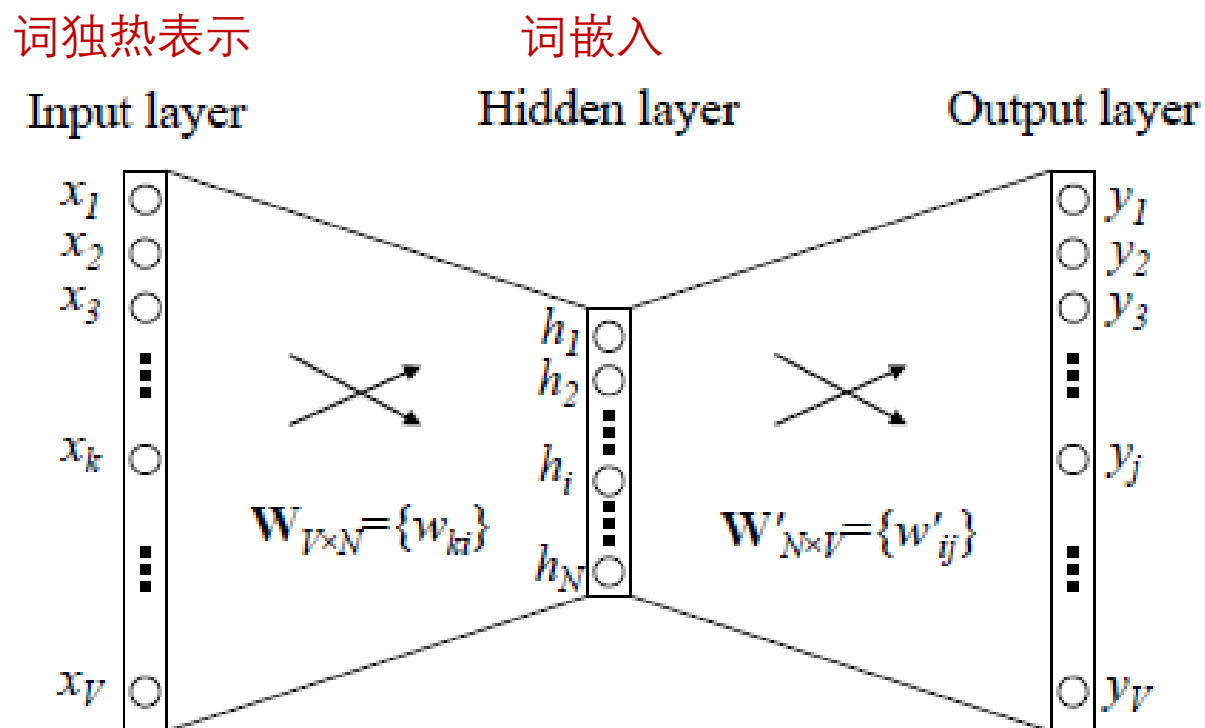
(a, dinner) (delicious, dinner) (last, dinner) (night, dinner)



Word2Vec

➤ Word Embedding 词嵌入

Word2Vec 的训练模型本质上是只具有一个隐含层的神经网络



以对数极大似然作为目标函数

$$\mathcal{L} = \sum_{w \in C} \log[p(w|Context(w))]$$

$$h = W^T x, x_i = 1, h = W[:, i]$$

$$y = W'^T h$$

归一化 $p(w|Context(w)) = \frac{e^{y_{w,i_w}}}{\sum_{j=1}^N e^{y_{w,j}}}$



Word2Vec

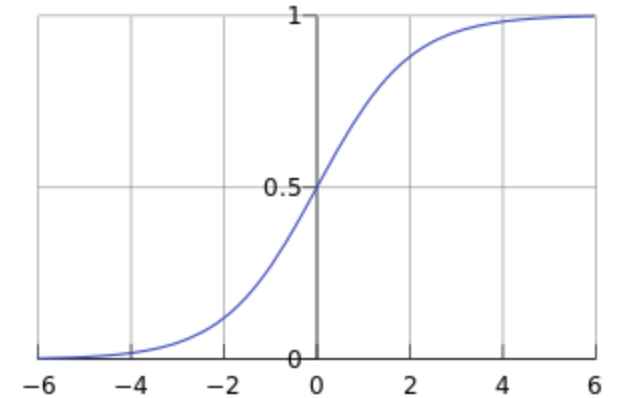
➤ Negative Sampling 负采样

将“预测目标词”看作一个二分类问题，与逻辑回归类似

预测为正例的概率 $p(w|j) = \sigma(y_{w,j}) = \sigma(h_w^T h_j)$

预测为负例的概率 $1 - p(w|j) = 1 - \sigma(h_w^T h_j)$

Sigmoid激活函数



损失函数
$$\mathcal{L}(w, j) = \underbrace{-d_{w,j} \cdot \log[\sigma(h_w^T h_j)]}_{\text{正例 } d_{w,j} = 1} - \underbrace{(1 - d_{w,j}) \cdot \log[1 - \sigma(h_w^T h_j)]}_{\text{负例 } d_{w,j} = 0}$$



Word2Vec

➤更多细节与数学原理

□如何选择负例？

□高效的代码实现？

□与预训练词嵌入模型的异同？

[word2vec 中的数学原理详解](#)



Word2Vec

➤ 开源代码

❑ C语言源码 <https://code.google.com/p/word2vec/>

❑ Python库 `gensim` <https://radimrehurek.com/gensim/models/word2vec.html>

```
>>> from gensim.test.utils import common_texts
>>> from gensim.models import Word2Vec
>>>
>>> model = Word2Vec(sentences=common_texts, vector_size=100, window=5, min_count=1, workers=4)
>>> model.save("word2vec.model")
```



```
[=====] 100.0% 1662.8
```

```
Finding Capital of Britain: (Paris - France) + Britain
```

```
[('London', 0.7541897892951965)]
```

```
Finding Capital of India: (Berlin - Germany) + India
```

```
[('Delhi', 0.72683185338974)]
```

```
5 similar words to BMW:
```

```
('Audi', 0.7932199239730835)
```

```
('Mercedes_Benz', 0.7683467864990234)
```

```
('Porsche', 0.727219820022583)
```

```
('Mercedes', 0.7078384757041931)
```

```
('Volkswagen', 0.695941150188446)
```

```
3 similar words to beautiful:
```

```
('gorgeous', 0.8353004455566406)
```

```
('lovely', 0.810693621635437)
```

```
('stunningly_beautiful', 0.7329413890838623)
```

```
Cosine similarity between fight and battle: 0.7021284
```

```
Cosine similarity between fight and love: 0.13506128
```