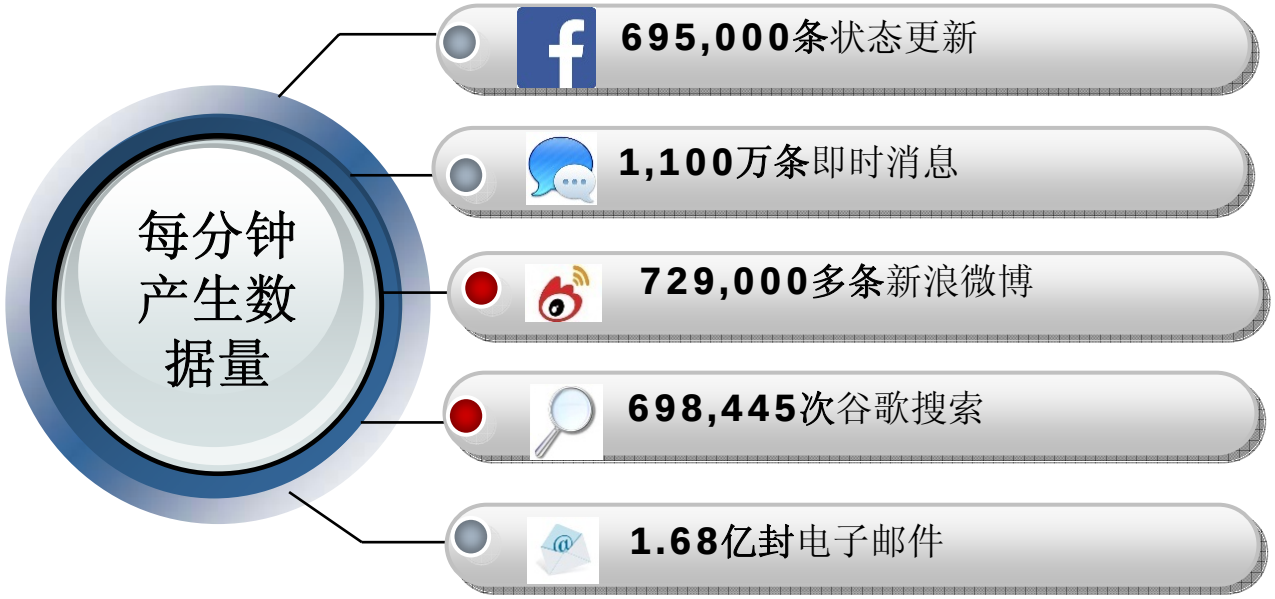
# 大数据时代的机器学习

何晓飞
浙江大学

# 大数据时代的机器学习

- 大数据时代机器学习的特点
- 传统机器学习
- 几个核心问题
  - 深度学习
  - 在线学习
  - 哈希索引
  - 基于树的索引

# 大数据时代的机器学习

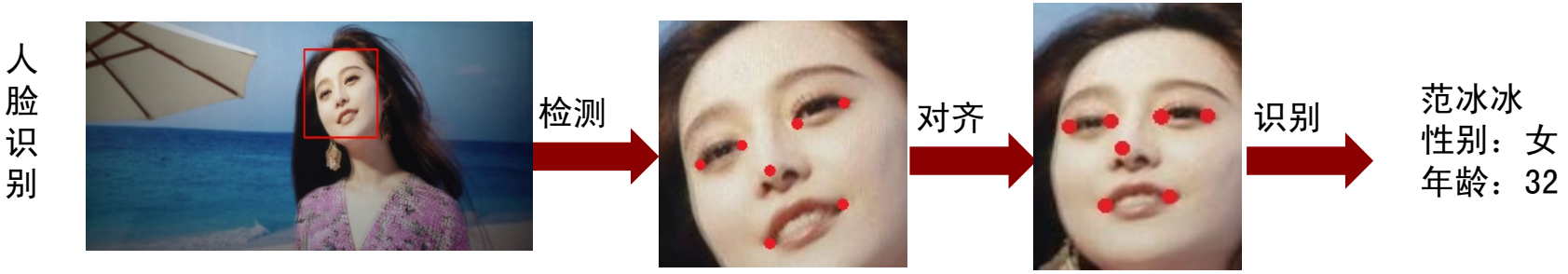



到2015年，全球互联网用户将达到**近30亿**，

全世界的数据量将达到**8ZB**。

# 人脸识别

- 如果在视频中我们看到感兴趣却不认识的人，就需要用人脸识别技术进行识别


但是在大数据时代，我们能得到的信息远远不止这些

# 信息提取

- 现在检索到的相关网页的信息往往是杂乱无章的，我们需要对其进行信息提取


非结构化数据

# 信息提取

- 现在检索到的相关网页的信息往往是杂乱无章的，我们需要对其进行信息提取


非结构化数据

范冰冰

范冰冰，出生于中国山东省青岛市，祖籍烟台，中国著名女演员、歌手，毕业于现今的上海师范大学谢晋影视艺术学院。凭其不断攀升的影响力，范冰冰为中国演艺圈中最具代表性的女明星之一，近年更进军国际影坛。
维基百科

**生于：** 1981 年 9 月 16 日（32 岁），青岛市

**身高：** 1.68 米

**即将上映的电影：** X战警：未来昔日

**所获奖项：** 大众电影百花奖最佳女主角

**参演电视剧：** 还珠格格， 封神榜， 小鱼儿与花无缺， 八大豪侠， 秦始皇

电影


苹果
2007 年


X战警：未来昔日
2014 年


钢铁人3
2013 年

登陆之日
2011 年

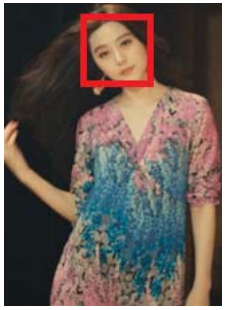

二次曝光
2012 年

结构化数据

# 图像检索

- 如果对其服装感兴趣，我们可以用图像检索查找相似的服装

范冰冰

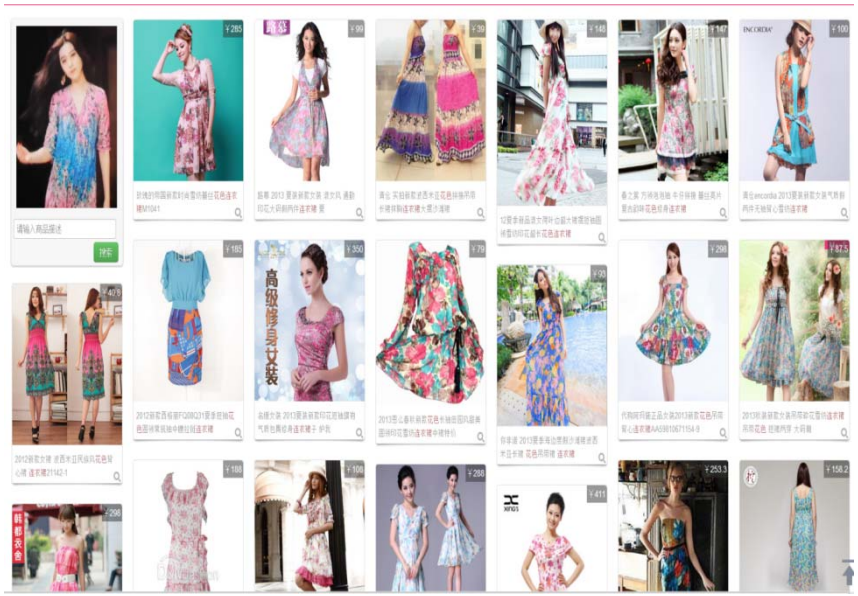


→图像搜索→

淘宝网 Taobao.com

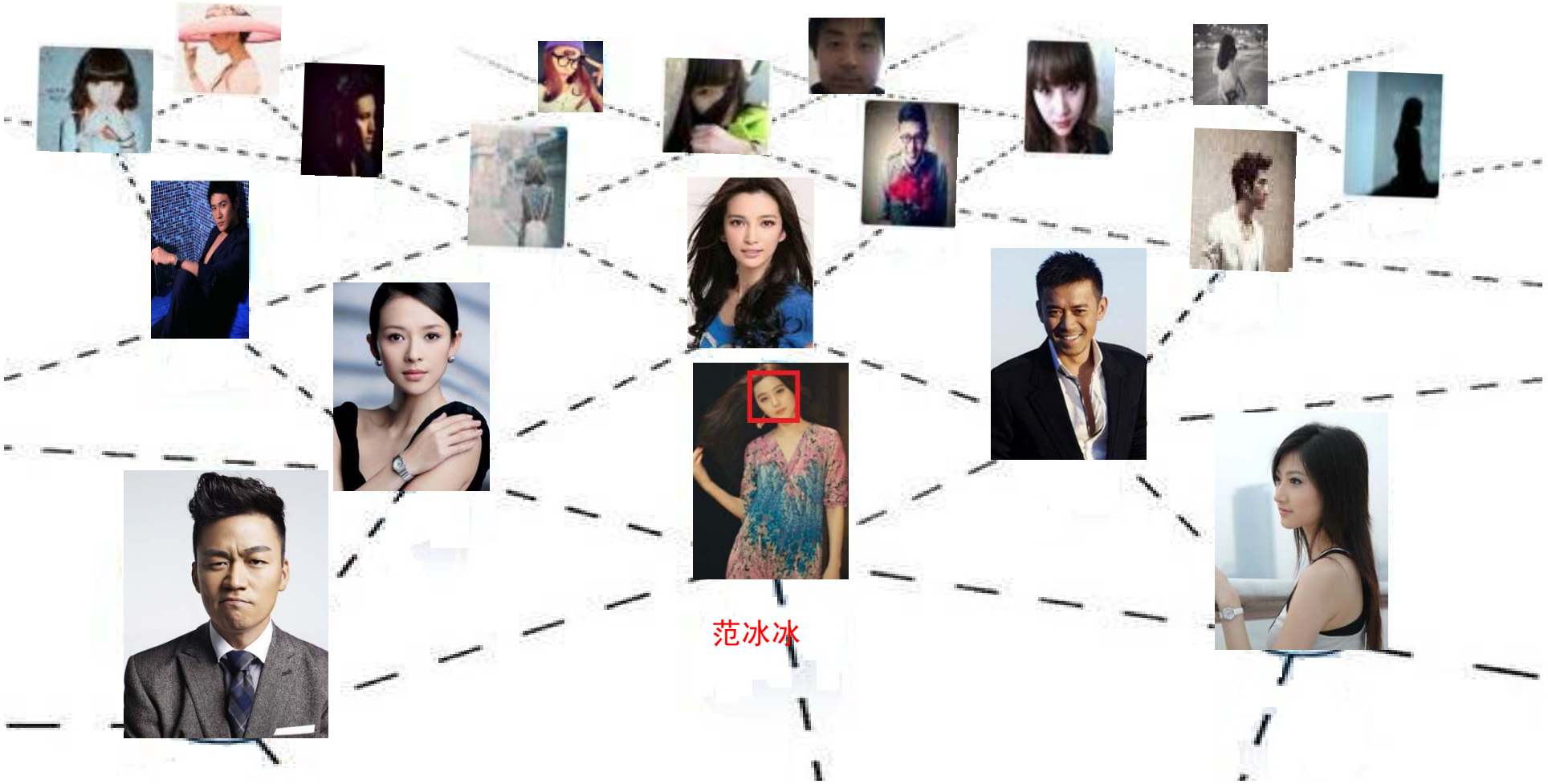
JD.京东 JD.COM

亚马逊 amazon.cn



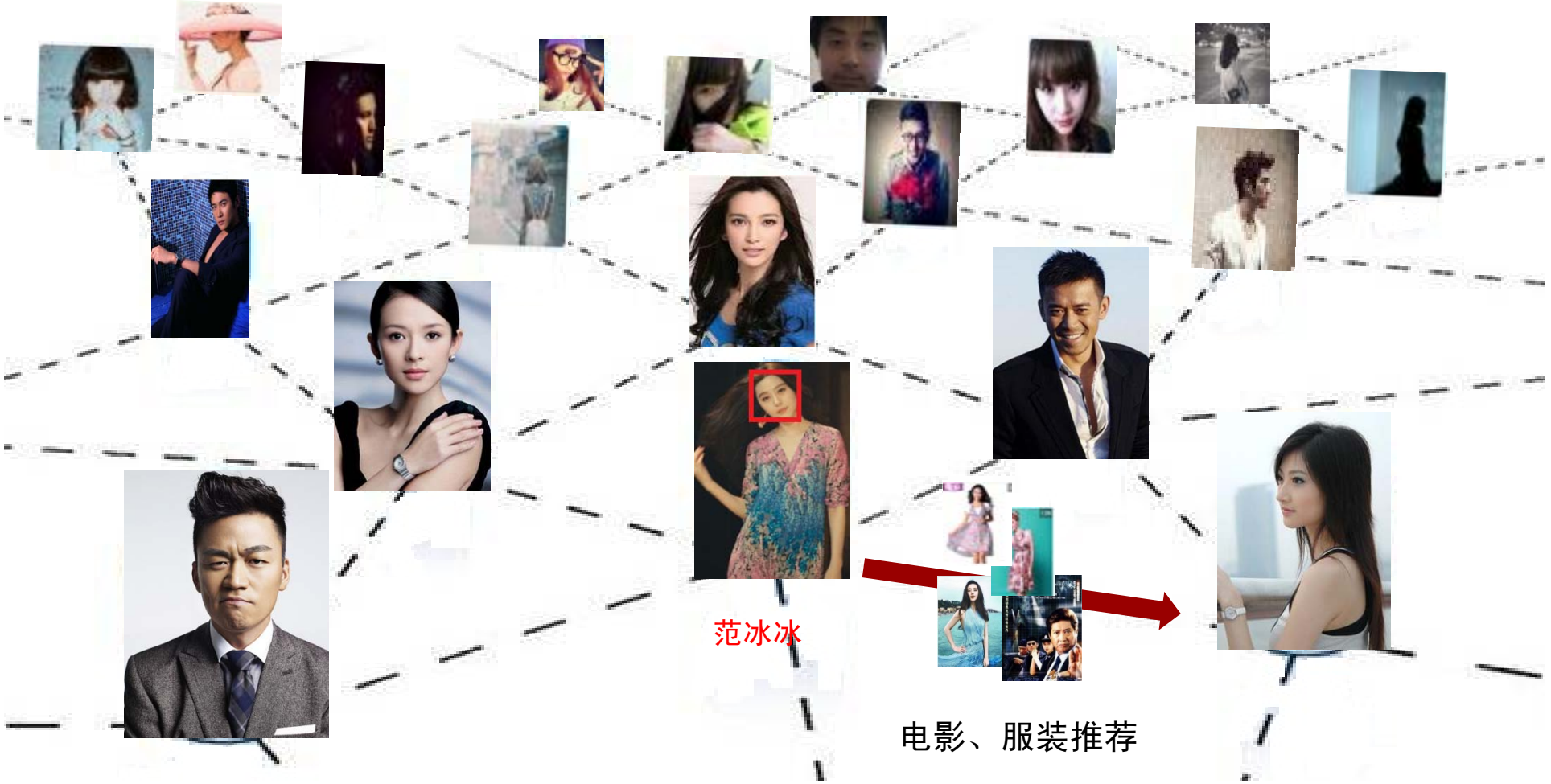
# 社交网络

- 然后，我们可以使用社交网络挖掘其社交关系



# 推荐系统

- 利用挖掘到的信息，我们做多种形式的推荐，
- 比如服装推荐，好友推荐，电影推荐



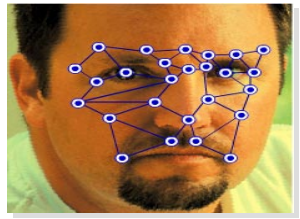
# 大数据时代的机器学习

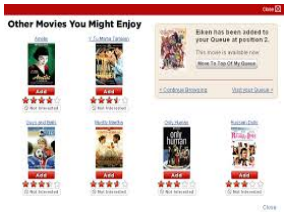
知识图谱

社交网络


图像检索


人脸识别






推荐系统

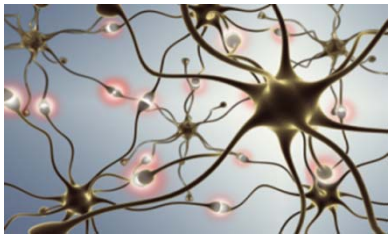
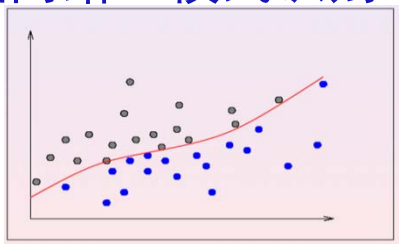
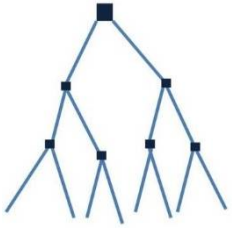
**大数据时代的机器学习特点：各种技术的融合**

# 大数据时代的机器学习

- 大数据时代机器学习的特点
- 传统机器学习
- 几个核心问题
  - 深度学习
  - 在线学习
  - 哈希索引
  - 基于树的索引

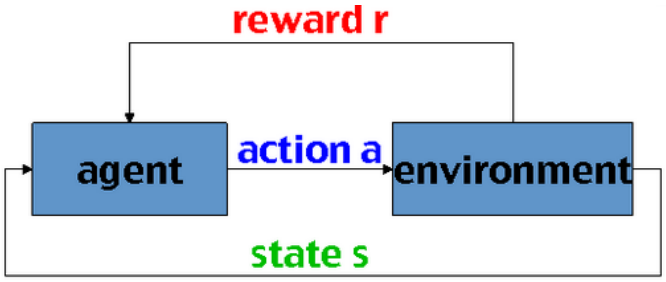
# 机器学习发展历程

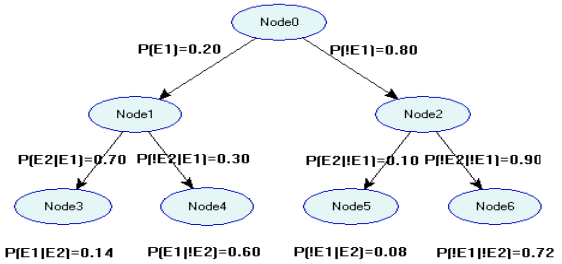
- 1960s-1980s：神经网络、模式识别、决策树 …

…

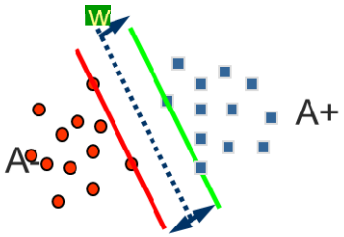
- 1990s：数据挖掘、强化学习、贝叶斯网络 、Boosting…



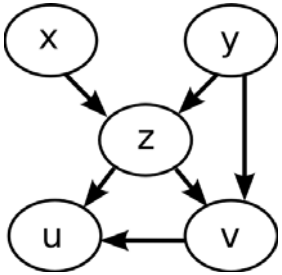



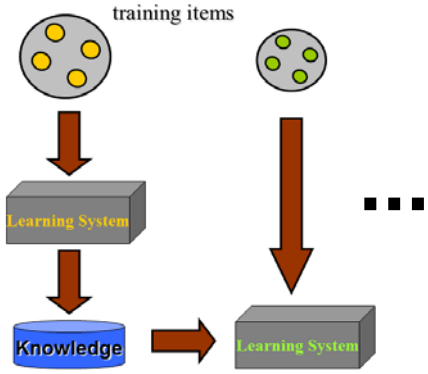

…

- 2000s:SVM、核方法、概率图模型、迁移学习…



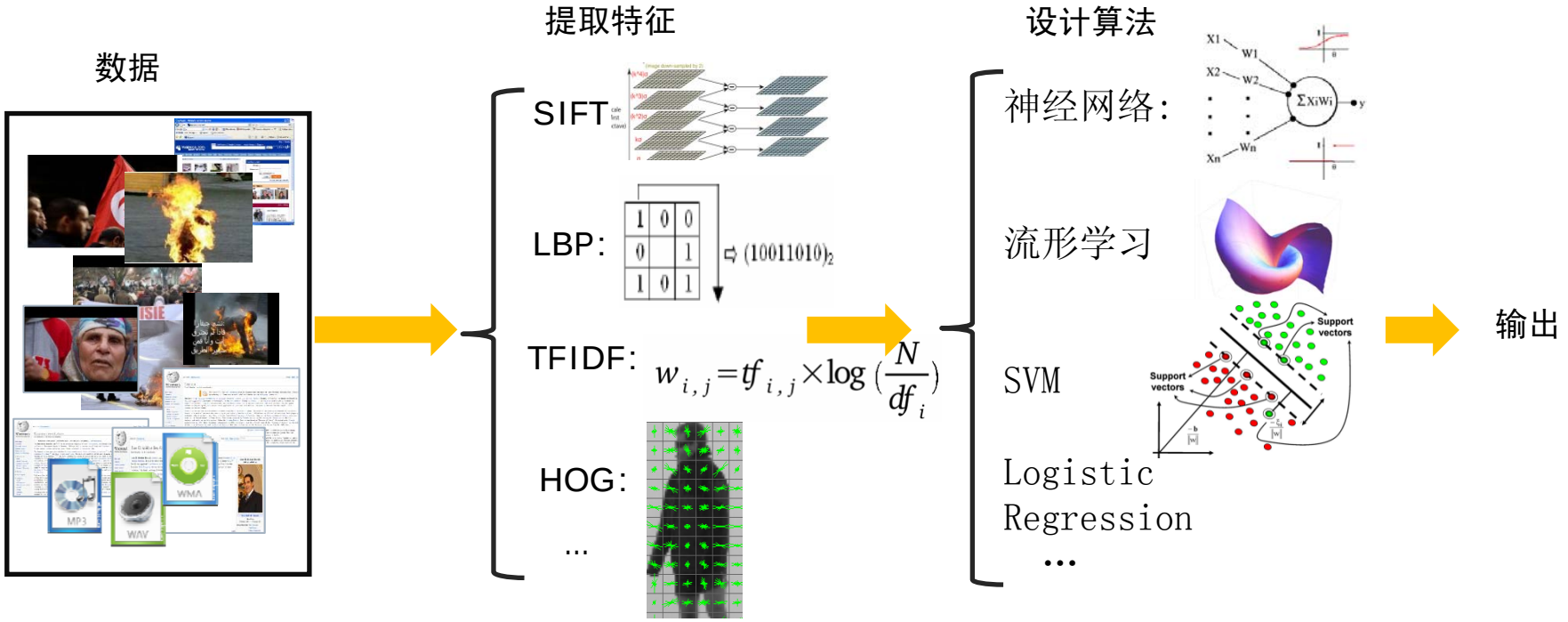

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b$$




…

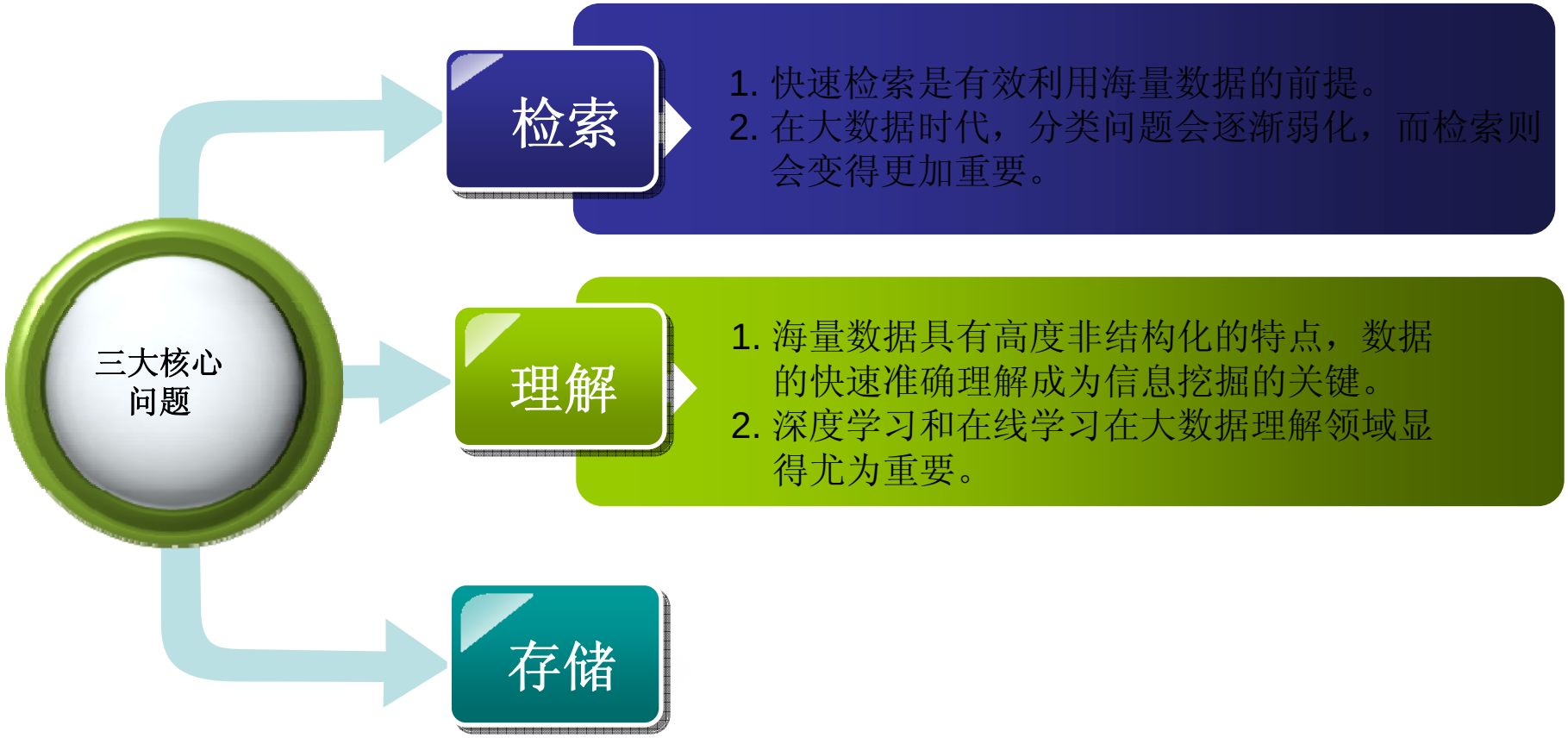
# 传统机器学习




- 缺陷：
  - 特征多为人工设计，耗时耗力且需要领域知识
  - 数据量小，可以一次性全部处理
  - 注重算法的精度，对于算法效率关注不够

# 大数据时代的机器学习

- 大数据时代机器学习的特点
  - 数据理解是难点
  - 分类会逐渐弱化，检索会更加重要

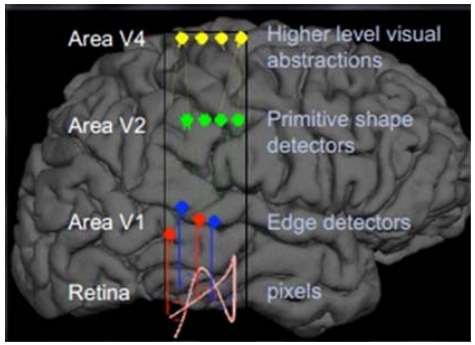


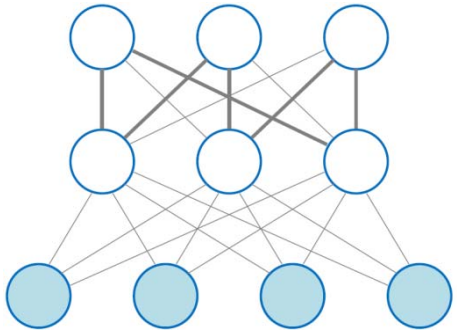
# 大数据时代的机器学习

- 大数据时代机器学习的特点
- 传统机器学习
- 几个核心问题
  - 深度学习
  - 在线学习
  - 哈希索引
  - 基于树的索引

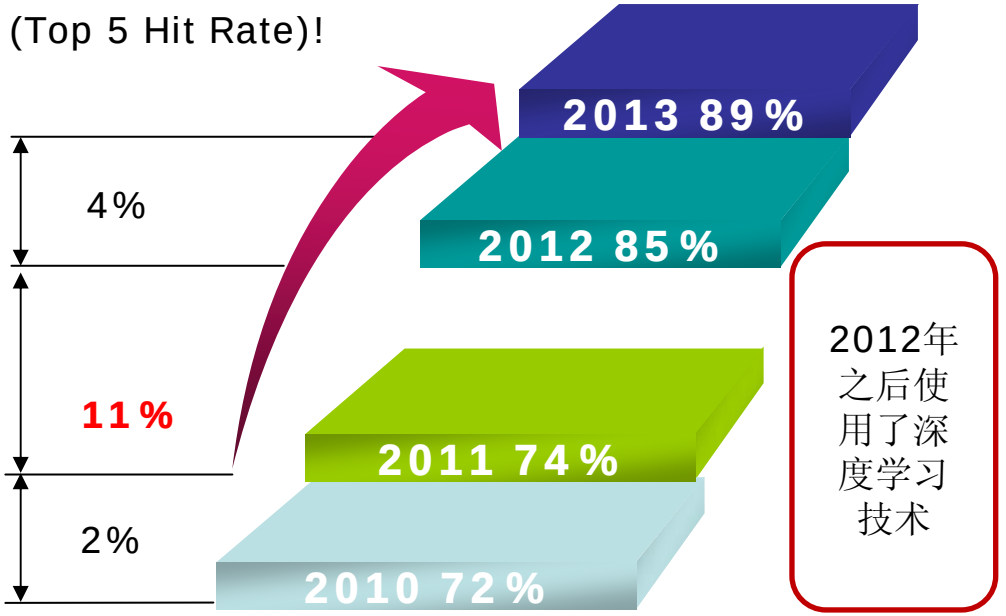
# 深度学习

- 从2006年开始重新兴起的一种学习方法，旨在建立类似人脑的神经网络，在学术界和工业界都产生了重大影响。



Race on ImageNet (Top 5 Hit Rate)!

# 深度学习

- **深度学习在大数据时代的重要性：**
  - 相比于以往的机器学习方法，深度学习能更有效利用海量数据。训练数据越大可以构建的模型越复杂，且越不容易over fitting

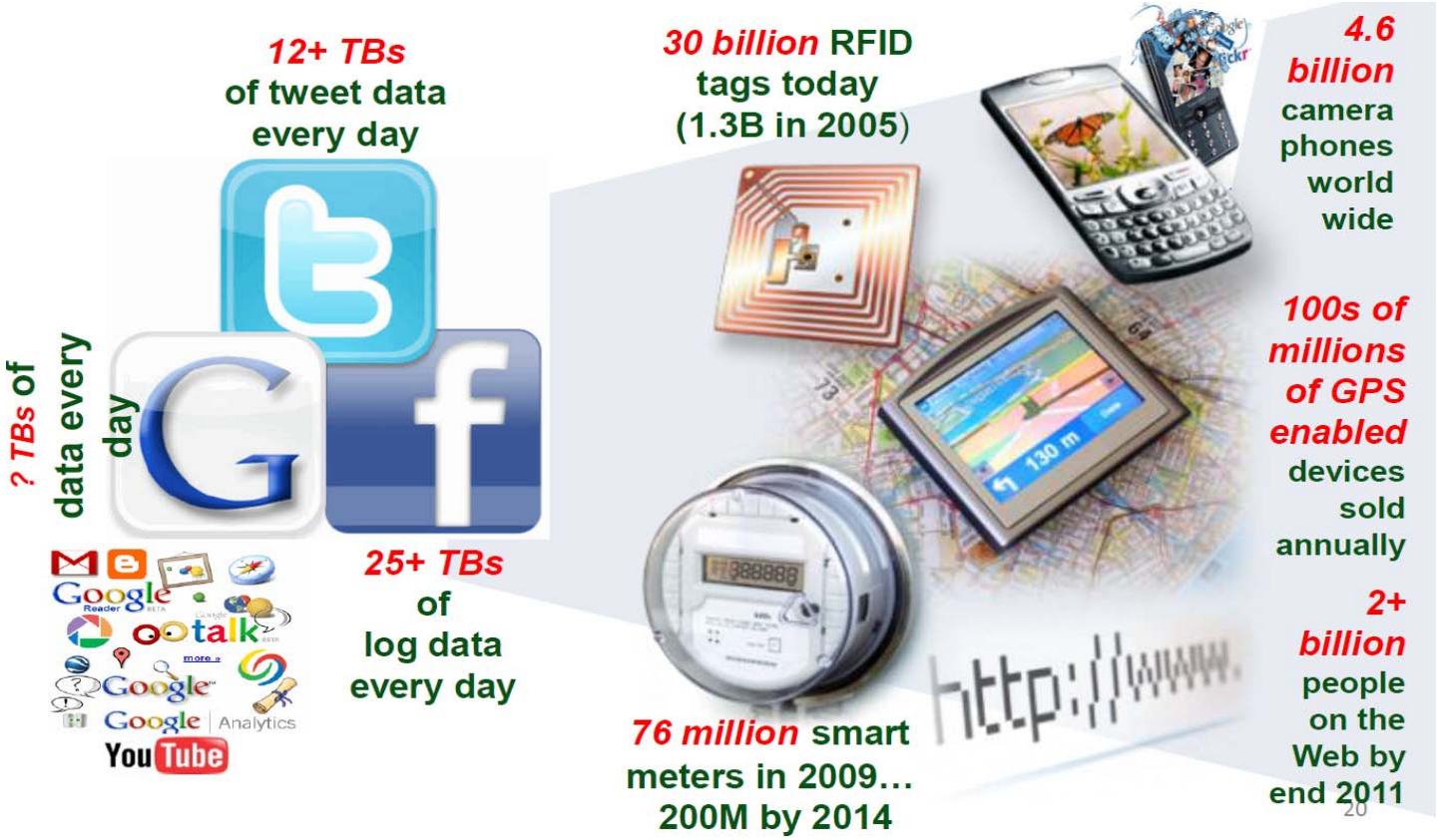
- **但是深度学习也存在一些问题：**
  - 算法上：训练耗时，调参复杂
  - 计算上：分布式优化困难
  - 硬件上：有些硬件还不能满足现在算法的需求，如 GPU稳定性差，显存小（运行时经常会出现GPU温度过高而导致程序崩溃的情形）

# 大数据时代的机器学习

- 大数据时代机器学习的特点
- 传统机器学习
- 几个核心问题
  - 深度学习
  - 在线学习
  - 哈希索引
  - 基于树的索引

# 在线学习

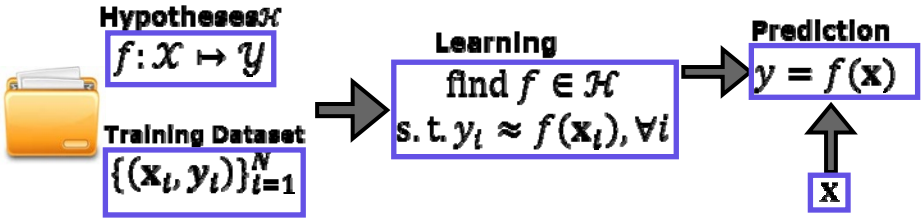
- 在大数据时代，新数据不断涌现，使得在线学习变得更加重要
  - 训练数据太大，离线学习方法训练时计算复杂度过高。
  - 需要不断更新模型，以适合新的数据，如果使用离线学习方法更新模型， 计算代价将无法忍受。



# 在线学习

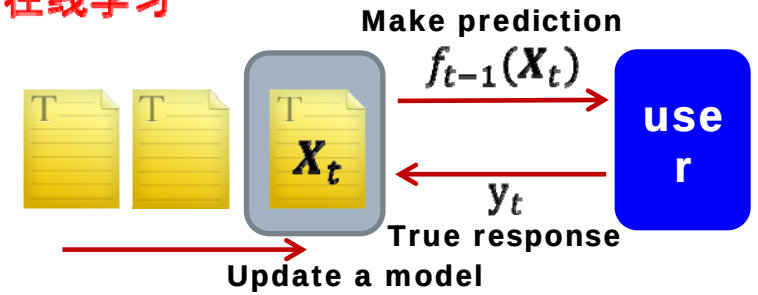
- 在线学习是什么?

- **批量学习/离线学习**




- 获得一个完整的训练数据集：
  $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$
- 从中学习出一个函数
  $f: \mathcal{X} \mapsto \mathcal{Y}$
- 用学到的函数$f$对新的样本点x进行预测

- **在线学习**




- 获得一列训练数据：
  $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_t, y_t)$
- 每获得一个新的样本点$\mathbf{x}_t$，更新函数
  $f: \mathcal{X} \mapsto \mathcal{Y}$
- 用新的函数$f$对数据序列进行预测

# 在线学习

- 当前在线学习方法的缺陷：
- 线性方法：对非线性数据效果很差
- 非线性方法（核方法）：需要保存历史数据，训练和测试复杂度高（随样本数线性增加），原因如下

**基于核方法的在线学习框架**

问题：$\min_f \Sigma_t L(y_t, f(x_t))$

subject to $|f|_{\mathcal{H}_\kappa} < R$

初始化：$f_1(x) = 0;$

**For** t = 1, …, T **do**

接收样本：$(x_t, y_t)$;

更新：$f_{t+1}(x) = f_t(x) - \eta \nabla L(y_t, f_t(x_t)) \kappa(x_t, x)$;

**End for**

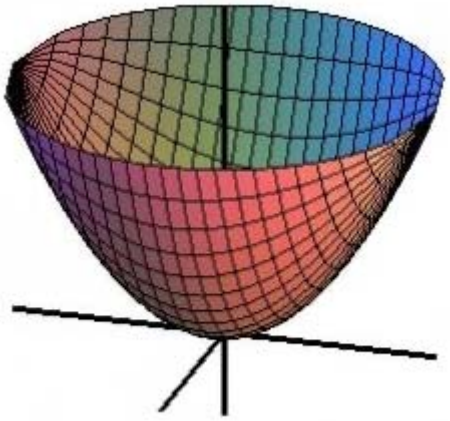
输出：$f(x) = \Sigma_t f_t(x)/T$

$L(u, v)$：损失函数

$\kappa(x, y)$：核函数

$f$：需要学习的函数

$\eta$：步长参数



$L(u, v)$示意图

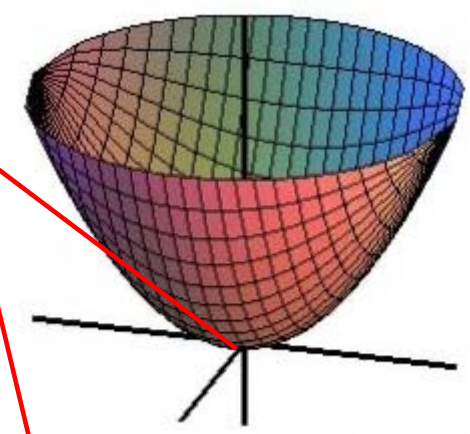
1. 每次都需要更新

2. 最后的$f(x)$与所有$x_t$相关，因此：
   - 需要保存所有历史数据
   - 计算$f(x)$复杂度高

# 在线学习

- 为解决上述非线性在线学习方法的缺陷，我们提出了具有上界的稀疏在线学习方法。

当$y_t$能被$f_t(x)$较为准确预测时，$\nabla L(y_t, f_t(x_t))$会比较小



**基于核方法的在线学习框架**

问题：$\min_f \Sigma_t L(y_t, f(x_t))$

subject to $|f|_{\mathcal{H}_\kappa} < R$

初始化：$f_1(x) = 0;$

**For** t = 1, …, T **do**

接收样本：$(x_t, y_t)$;

更新：$f_{t+1}(x) = f_t(x) - \eta \nabla L(y_t, f_t(x_t)) \kappa(x_t, x)$;

**End for**

输出：$f(x) = \Sigma_t f_t(x)/T$

**主要思路：**

我们是否可以有选择性地更新？

如果可以，那何时更新，决定更新的标准是什么？

**我们认为：**

当$y_t$能被$f_t(x)$较为准确预测时，我们不需要更新。可以用$\nabla L(y_t, f_t(x_t))$来衡量 是否需要更新。

$\nabla L(y_t, f_t(x_t))$越大，表明$y_t$越不能被$f_t(x)$准确预测，越需要更新

# 在线学习

我们的算法

问题：$\min \Sigma_f L(y_t, f(x_t))$
$\text{subject to} |f|_{\mathcal{H}_K} < R$

初始化：$f_1(x) = 0;$
**For** $t = 1, \ldots, T$ **do**
(1)接收样本：$(x_t, y_t);$

(2)计算$\nabla L(y_t, f_t(x_t))$，采样一个随机样本$Z_t$
其中$\Pr(Z_t = 1) = \frac{1}{G}|\nabla L(y_t, f_t(x_t))|$

(3)更新：$f_{t+1}(x) = f_t(x)$
$-\eta \text{sign}(\nabla L(y_t, f_t(x_t)))Z_t G\kappa(x_t, x);$
**End for**

输出：$f(x) = \Sigma_t f_t(x)/T$

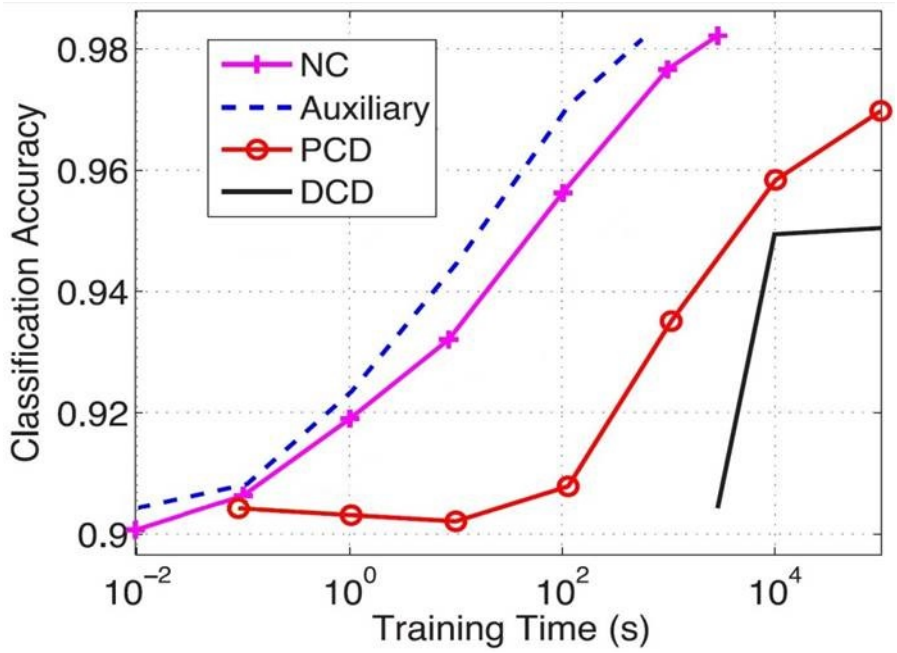
只有当$\frac{1}{G}|\nabla L(y_t, f_t(x_t))|$比较大时，选择更新的概率才比较大

# 在线学习

- 我们算法的更新次数具有如下上界：

$$\sum_{t=1}^{T} Z_t < \mathrm{Poly}(\ln T)$$

- 实验结果：
  - 我们在ijcnn1数据集上进行测试，评价标准为相同训练时间内所能达到的分类准确率




Auxilary:我们的算法
NC: 传统在线学习算法
PCD\DCD: 批量学习算法

# 大数据时代的机器学习

- 大数据时代机器学习的特点
- 传统机器学习
- 几个核心问题
  - 深度学习
  - 在线学习
  - 哈希索引
  - 基于树的索引

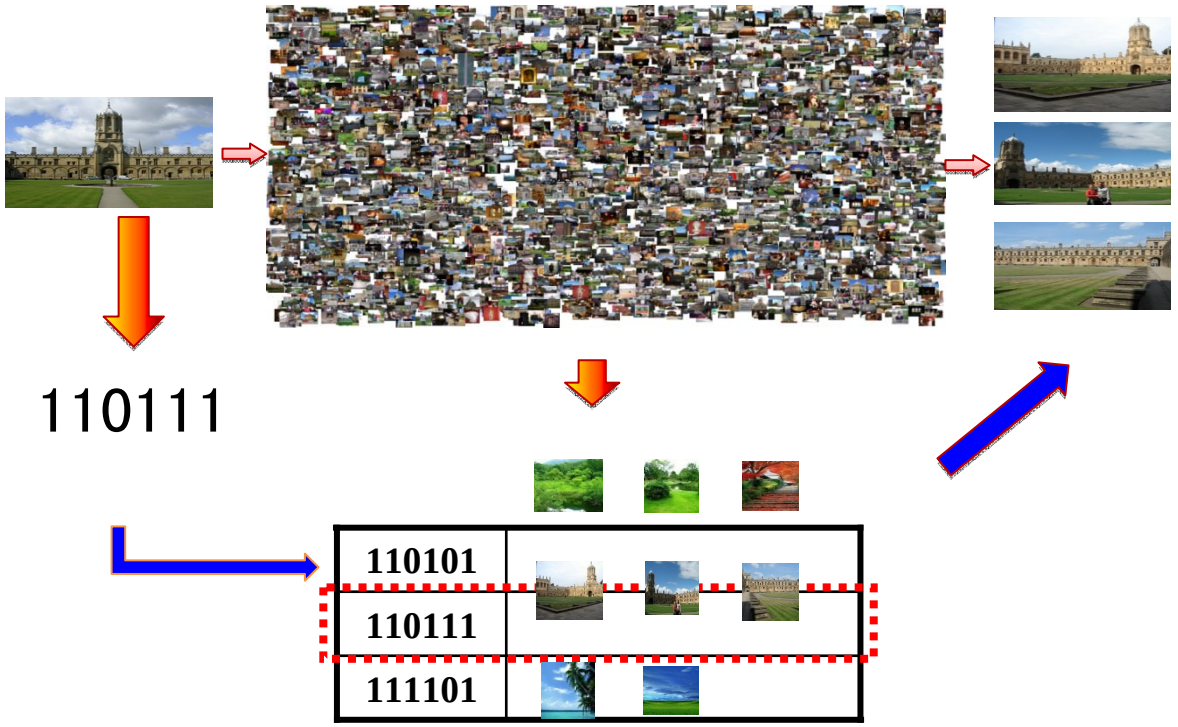
# 哈希索引

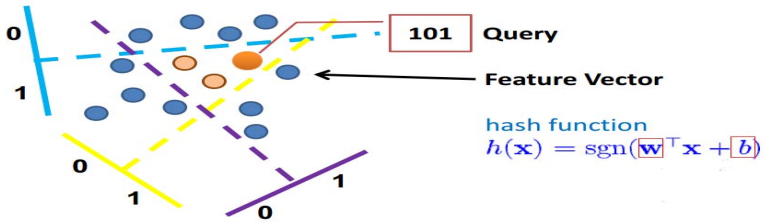
- 近似最近邻检索的重要性：
- 在大数据时代精确最近邻检索复杂度太高，所以常常采用近似最近邻检索，常用的方法有哈希索引和树结构检索。
- 近似近邻搜索的重要性：例如，数据规模：1亿，960维图片Gist特征，用普通台式机（i7, 3.4GHz）检索1次,1-NN。

| 检索方法 | 线性检索 | 树结构检索(kd-tree) | 传统哈希检索(LSH) |
|---|---|---|---|
| 训练时间 | 0 | 6小时 | 3小时 |
| 检索时间 | 2分钟 | 20毫秒 | 2秒 |
| 存储空间 | 360GB+0GB | 360+8GB | 360+6GB |
| 精度 | 100% | 80% | 80% |

# 哈希索引

- 什么是哈希索引：



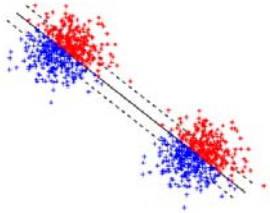




用分割面将数据点分割到一个个区域（哈希桶）中，用0-1向量代表原向量，提高检索效率。

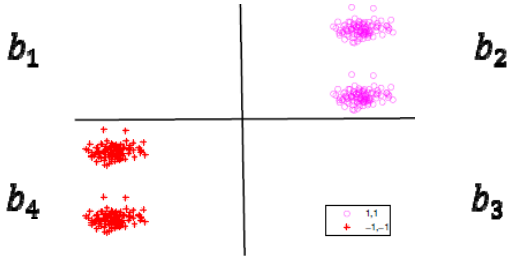
# 哈希索引

## 传统哈希索引存在的问题：

- 相邻的数据点被分到不同的哈希桶中
  - 解决办法：密度敏感哈希索引（Density Sensitive Hashing）



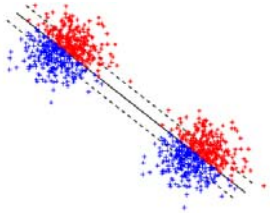
- 哈希桶$(b_1, \ldots, b_4)$中的数据点数量不均衡
  - 解决办法：互补投影哈希算法(Complementary Projection Hashing)



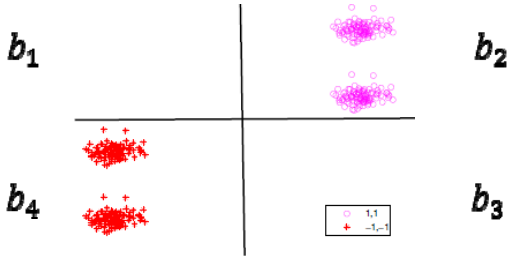

- 在编码较短时无法获取优秀的性能
  - 解决办法：压缩哈希（Compressed Hashing）

# 哈希索引

- 传统哈希索引存在的问题：
  - 相邻的数据点被分到不同的哈希桶中
    - 解决办法：密度敏感哈希索引（Density Sensitive Hashing）



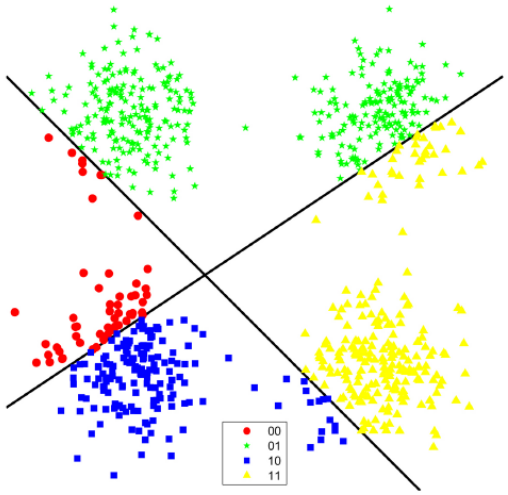
  - 哈希桶$(b_1, \ldots, b_4)$中的数据点数量不均衡
    - 解决办法：互补投影哈希算法(Complementary Projection Hashing)




  - 在编码较短时无法获取优秀的性能
    - 解决办法：压缩哈希（Compressed Hashing）

# 哈希索引

- 密度敏感哈希索引(Density Sensitive Hashing)
  - 该方法主要针对问题1：相邻的数据点被分到不同的哈希桶中
  - 传统LSH会出现如下情形



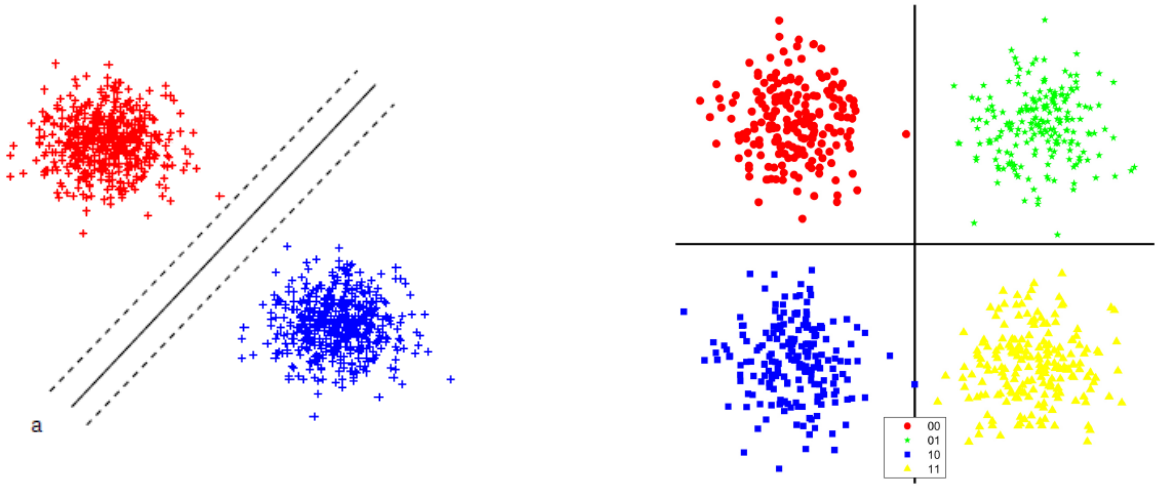

会返回许多的伪正例(false negative)，伪正例越多，之后计算原空间中实际距离所需要的时间就会越多

# 哈希索引

## 密度敏感哈希索引(Density Sensitive Hashing)

### 主要思想

一个合理的哈希函数应该如下图所示，分类面不能穿过数据密集分布的区域




### 具体形式

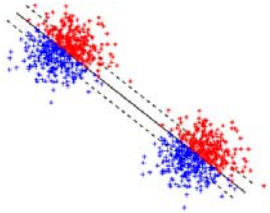
通过对分类面附近的数据点数进行惩罚，使得分类面穿过数据分布稀疏的区域

$\min \Sigma_{l=1}^{n}(H(\varepsilon - |W_k^T X_l - b_k|))$ ，其中$H(\varepsilon - |x|) = \frac{1}{2} + \frac{1}{2}\mathrm{sgn}(\varepsilon - x \cdot \mathrm{sgn}(x))$

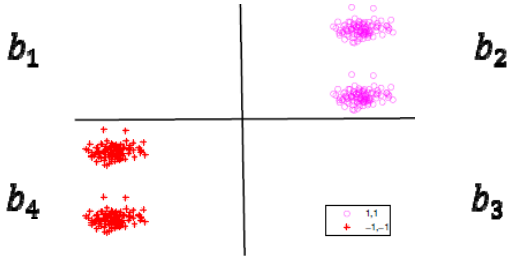
# 哈希索引

## 传统哈希索引存在的问题：

- 相邻的数据点被分到不同的哈希桶中
  - 解决办法：密度敏感哈希索引（Density Sensitive Hashing）



- 哈希桶 $(b_1, \ldots, b_4)$ 中的数据点数量不均衡
  - 解决办法：互补投影哈希算法(Complementary Projection Hashing)

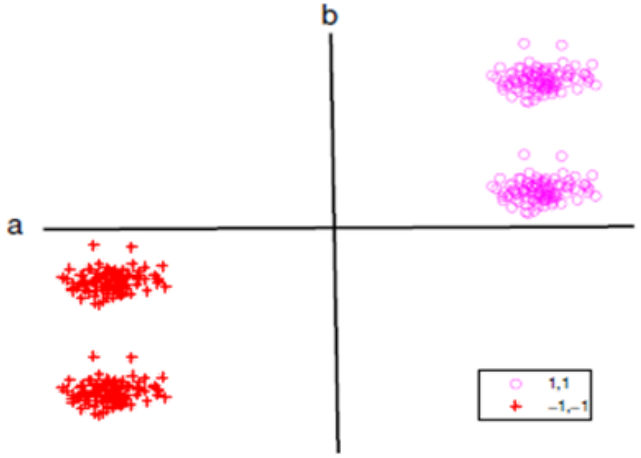



- 在编码较短时无法获取优秀的性能
  - 解决办法：压缩哈希（Compressed Hashing）

# 哈希索引

- 互补投影哈希算法(Complementary Projection Hashing)
  - 该方法主要针对问题2：哈希桶中的数据点数量不均衡。
  - 主要思想：

    在前一工作中，我们解决了分类面通过数据密集区域的问题。但是我们发现哈希桶中的数据点数量不均衡。



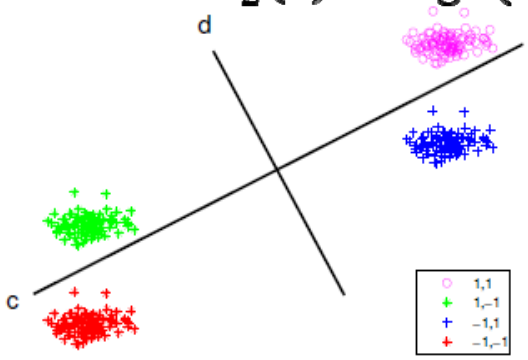

(LSH)

- 这会导致哈希检索时多次访问到空的哈希桶，从而不得不扩大搜索范围，最终降低检索效率
- 所以我们希望不出现空桶，各个哈希桶间的数据量尽量均衡

# 哈希索引

- 互补投影哈希算法(Complementary Projection Hashing)
  - 我们希望最终的哈希桶间的数据量尽量均衡，应用数学定理，将此要求转化为惩罚函数

$$h_2(\mathbf{x}) = \text{sign}(w_2^T \mathbf{x}_i - b_2)$$




$$h_1(\mathbf{x}) = \text{sign}(w_1^T \mathbf{x}_i - b_1)$$

  - 数据被均匀分隔的条件等价于

$$\begin{cases} \Sigma_i h_1(x_i) = 0 \\ \Sigma_i h_2(x_i) = 0 \\ \Sigma_i h_1(x_i) h_2(x_i) = 0 \end{cases}$$

# 哈希索引

- 互补投影哈希算法(Complementary Projection Hashing)
  - 实验结果对比：

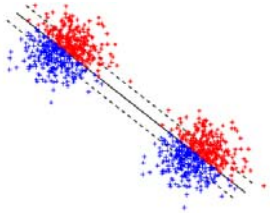
我们在100万大小的数据集上，在不同的哈希长度下，分别测试各种哈希算法和我们的上述CPH方法，将和测试点最相近的1000个点作为近邻。用平均精度MAP（Mean Average Precision）作为评价指标，结果如下：

| 哈希长度 | 16 | 24 | 32 | 48 | 64 |
|---|---|---|---|---|---|
| LSH | 0.02 | 0.035 | 0.045 | 0.085 | 0.135 |
| CPH | 0.05 | 0.085 | 0.12 | 0.18 | 0.22 |

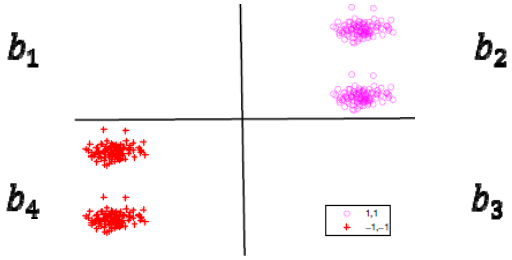
# 哈希索引

## 传统哈希索引存在的问题：

- 相邻的数据点被分到不同的哈希桶中
  - 解决办法：密度敏感哈希索引（Density Sensitive Hashing）



- 哈希桶$(b_1, \ldots, b_4)$中的数据点数量不均衡
  - 解决办法：互补投影哈希算法(Complementary Projection Hashing)



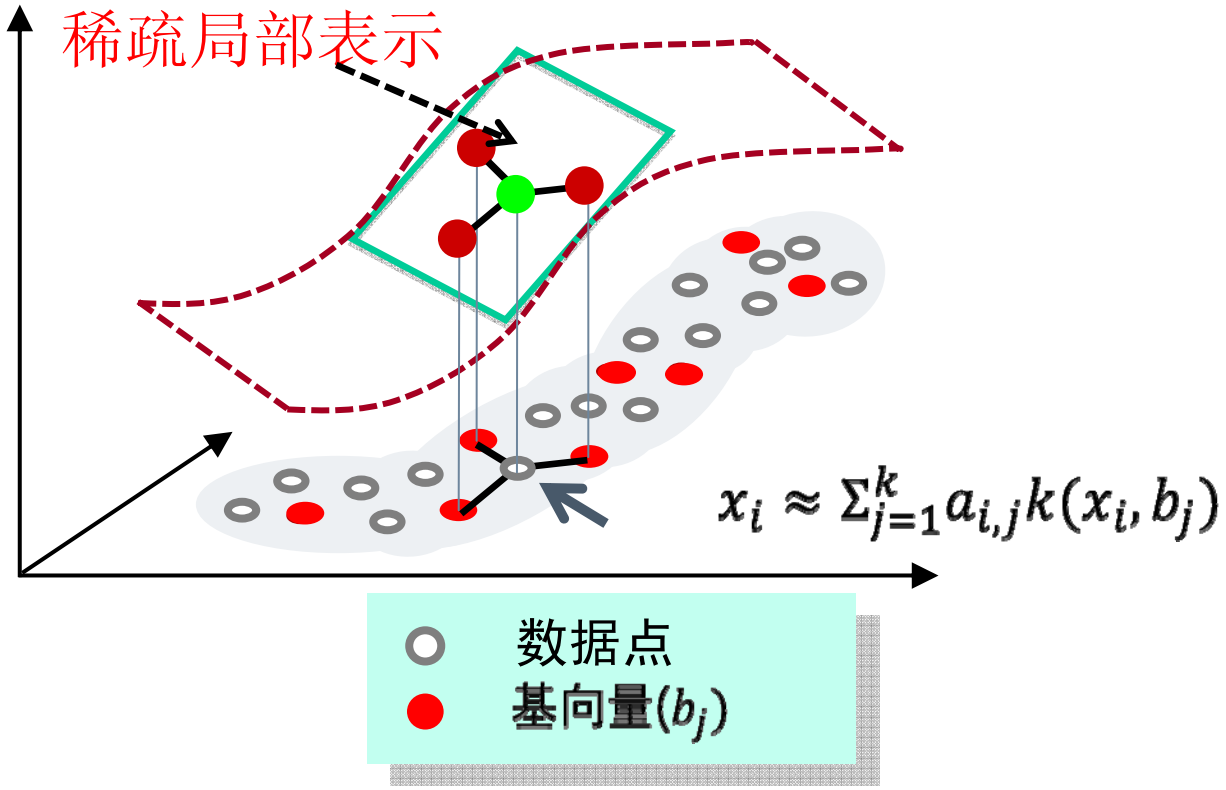

- 在编码较短时无法获取优秀的性能
  - 解决办法：压缩哈希（Compressed Hashing）

# 哈希索引

- 压缩哈希（Compressed Hashing）
  - 该方法主要用来解决问题3：在编码较短时无法获取优秀的性能
  - Restricted Isometry Property：假设d < m，令$\phi$是一个根据符合均值为0和方差是1的高斯分布而随机产生的矩阵，矩阵大小为$m \times d$。如果$s/m$足够小，令$d = C_s s \log\left(\frac{m}{s}\right)$,其中$C_s$是一个和$s$相关的参数，则以下情况会以非常大的概率成立：存在一个正常量$\delta_s$，使得下面的不等式对于任意的向量$z \in R^m$成立，其中$z$最多有$s$个维度非零：$(1-\delta_s)|z|_2 \leq \left|\phi^{\mathrm{T}} z\right| \leq (1+\delta_s)|z|_2$

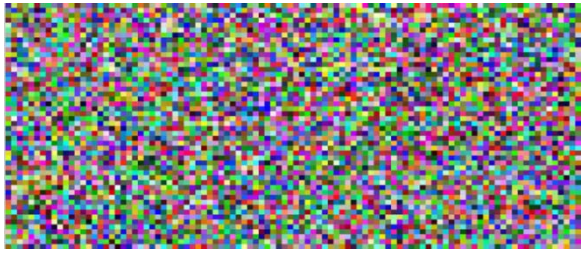
# 哈希索引

- 压缩哈希（Compressed Hashing）
  - 主要思想及解决方法
    - 充分考虑数据几何结构
    - 用稀疏编码和压缩传感理论对数据进行更好的表达

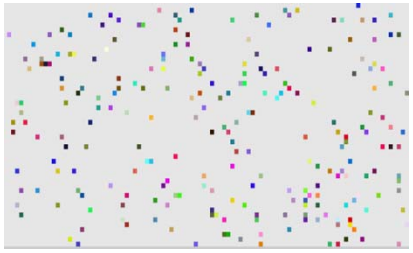


# 哈希索引

- 压缩哈希（Compressed Hashing）
  - 方法步骤
    - 对数据进行稀疏表示（用m个代表点的组合系数来表示每个数据点）：把数据从　维转为　维

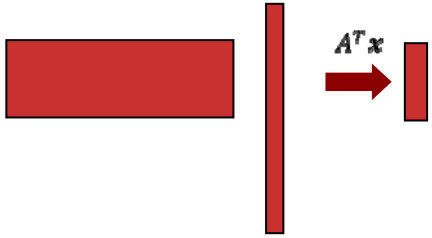


列为原始维度
（每一行为一个数据点）



列为基向量个数
（每一行为数据的稀疏表示）

    - 用压缩传感理论(Restricted Isometry Property)对数据进行低维表述(投影)，去除原始坐标中大量的0值，加速后续计算




    - 对最终的向量进行二值表示（大于平均值为1，否则为0）

# 哈希索引

- 压缩哈希（Compressed Hashing）
  - 实验结果对比
    - 我们在100万大小的数据集上，在不同的哈希长度下，分别测试各种哈希算法和我们的CH方法，并将与测试点最近的2%的点作为近邻点，用平均精度MAP（Mean Average Precision）作为评价指标，结果如下：

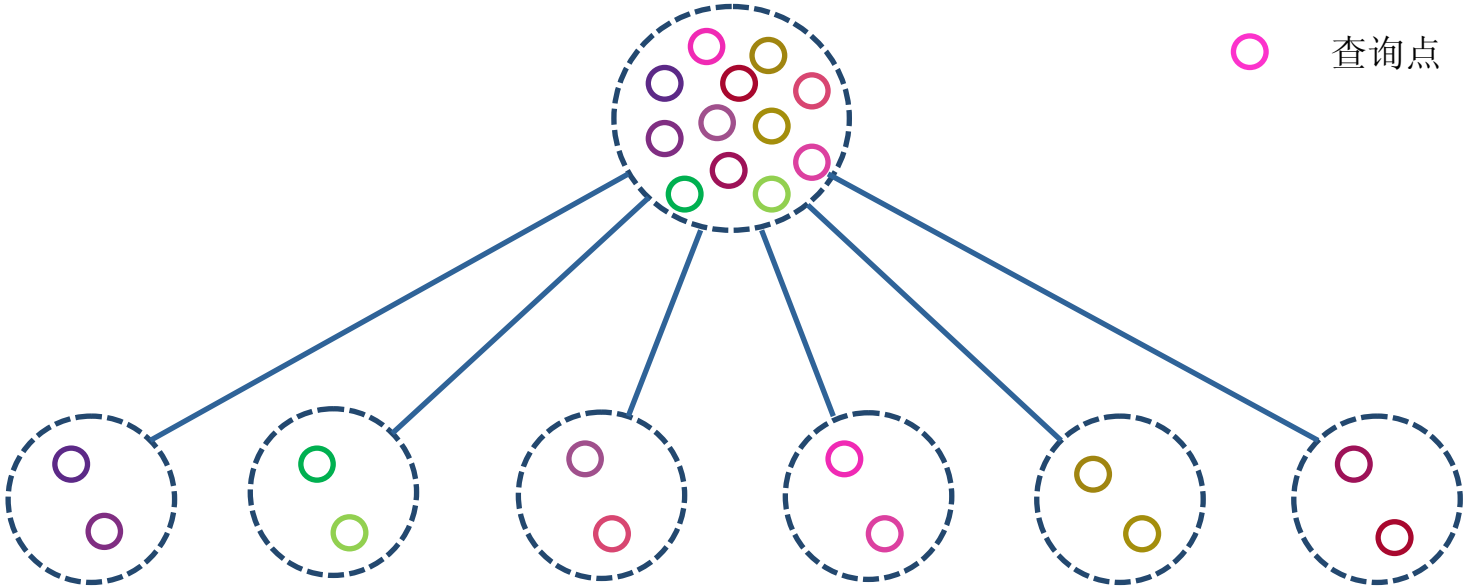
| 哈希长度 | 16 | 32 | 48 | 64 | 80 | 96 |
|---|---|---|---|---|---|---|
| LSH | 0.15 | 0.24 | 0.29 | 0.33 | 0.37 | 0.41 |
| CH | 0.23 | 0.33 | 0.38 | 0.41 | 0.42 | 0.45 |

# 大数据时代的机器学习

- 大数据时代机器学习的特点
- 传统机器学习
- 几个核心问题
  - 深度学习
  - 在线学习
  - 哈希索引
  - 基于树的索引

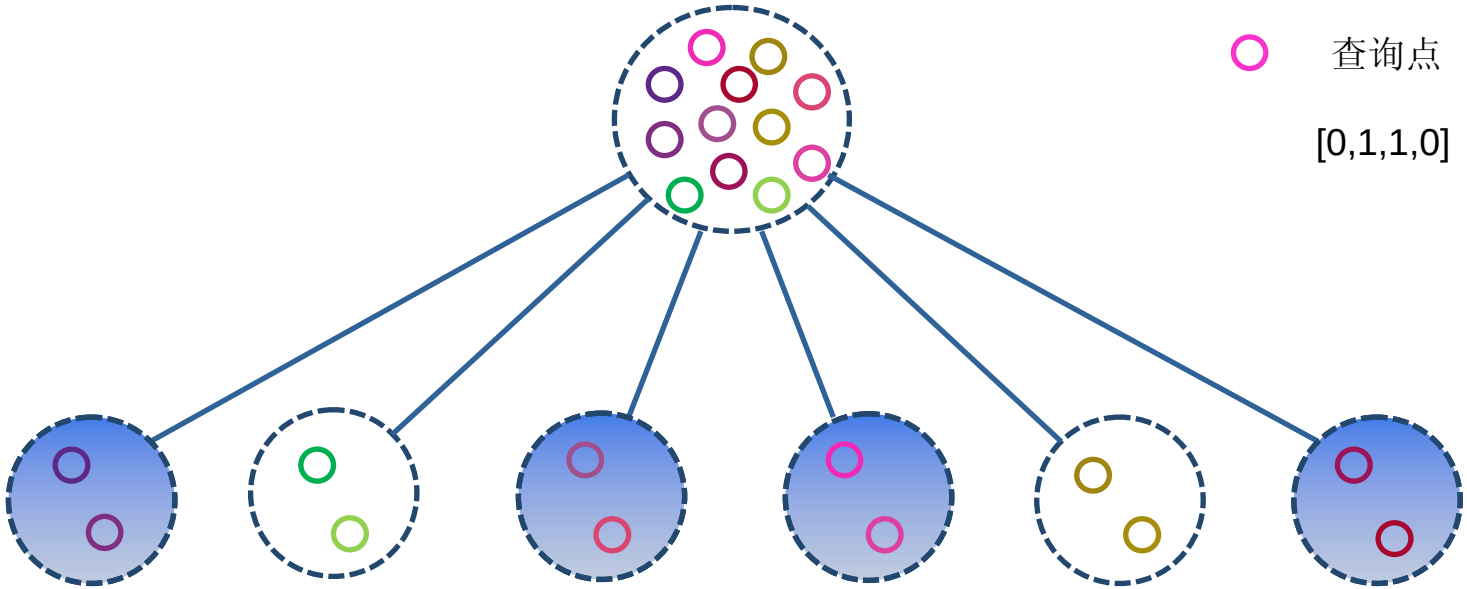
# 基于树的索引

## 统一的近似最近邻检索



# 基于树的索引

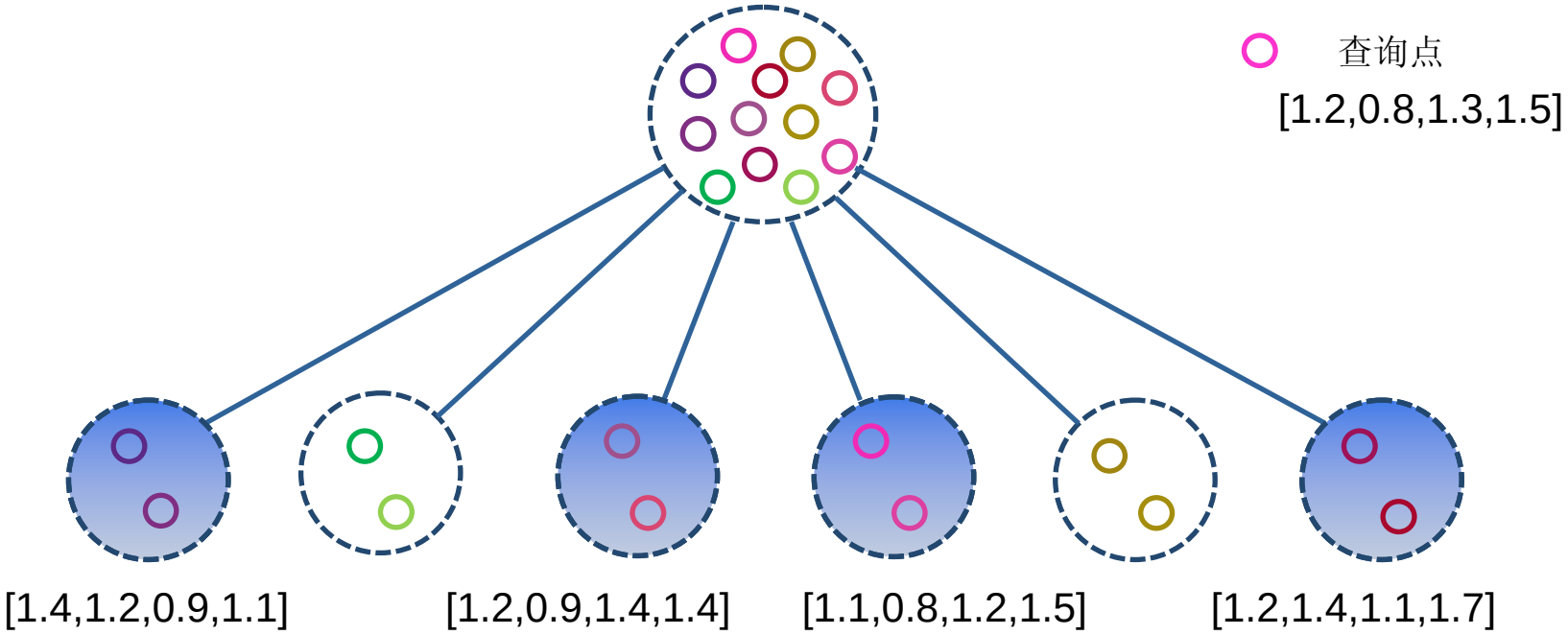
## 统一的近似最近邻检索




- 计算查询点和每个节点的Hashing码之间的Hamming距离，对节点做近似筛选

# 基于树的索引

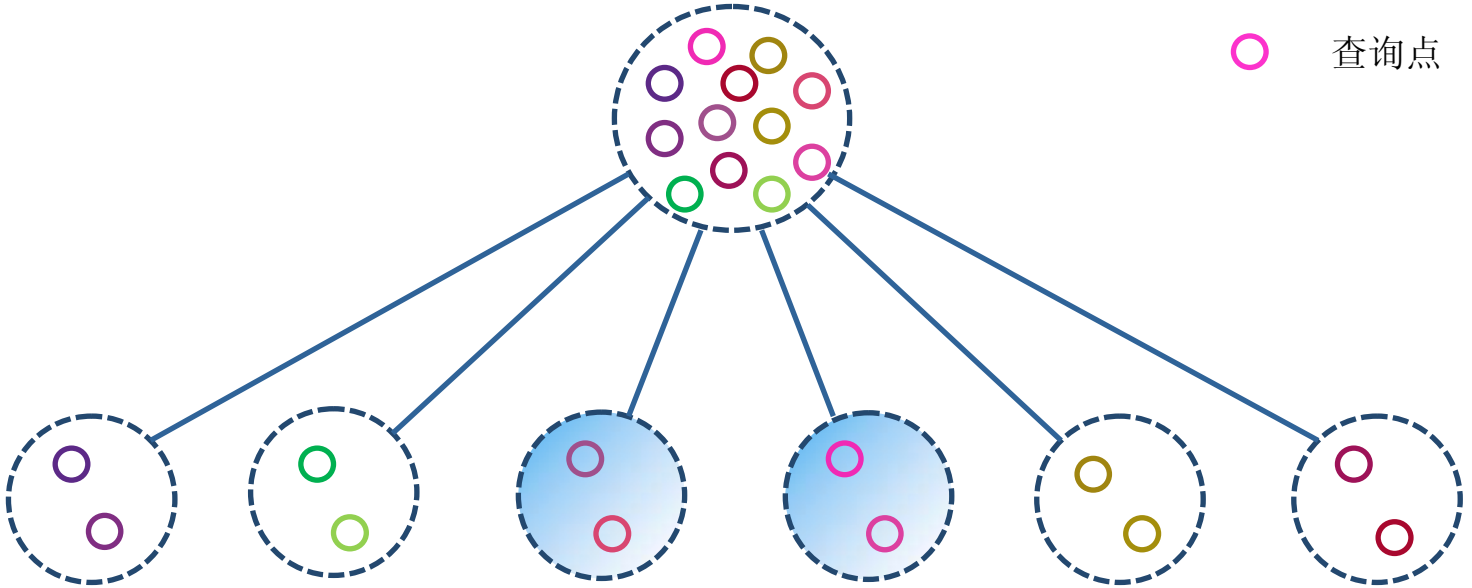
## 统一的近似最近邻检索




- 在近似检索的基础上，使用欧式距离作精确检索

# 基于树的索引

## 统一的近似最近邻检索




- 选出最近的两个节点，进入下一层

# 基于树的索引

- 统一的近似最近邻检索：
- 实验效果
  - 数据集：SIFT-1M，GIST-1M

| DataSet | Mean Preicision | Method | Time (ms) |
|---|---|---|---|
| SIFT-1M | 80% | HKM+KLSH | 0.18 |
| | | HKM | 0.51 |
| | | kd-trees | 0.92 |
| | 90% | HKM+KLSH | 0.42 |
| | | HKM | 0.78 |
| | | kd-trees | 2.2 |
| GIST-1M | 80% | HKM+KLSH | 1.8 |
| | | HKM | 6.7 |
| | | kd-trees | 4.2 |
| | 90% | HKM+KLSH | 4.2 |
| | | HKM | 12.2 |
| | | kd-trees | 9.3 |

1NN的情形

| DataSet | Mean Preicision | Method | Time (ms) |
|---|---|---|---|
| SIFT-1M | 80% | HKM+KLSH | 0.59 |
| | | HKM | 4.0 |
| | | kd-trees | 1.0 |
| | 90% | HKM+KLSH | 1.3 |
| | | HKM | 9.5 |
| | | kd-trees | 1.9 |
| GIST-1M | 70% | HKM+KLSH | 1.0 |
| | | HKM | 7.5 |
| | | kd-trees | 7.5 |
| | 80% | HKM+KLSH | 1.8 |
| | | HKM | 12 |
| | | kd-trees | 13 |

50NN的情形

  - 比较对象：kd-树，分层k-means树（HKM）

谢谢！