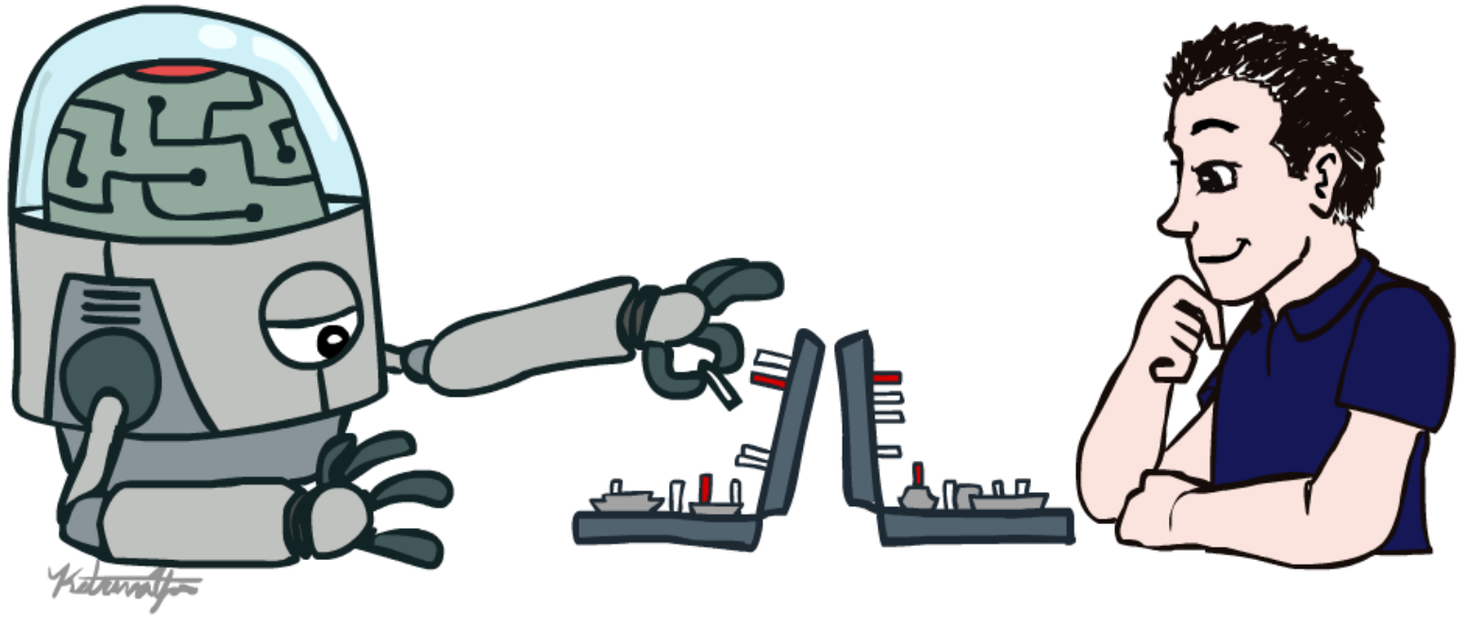
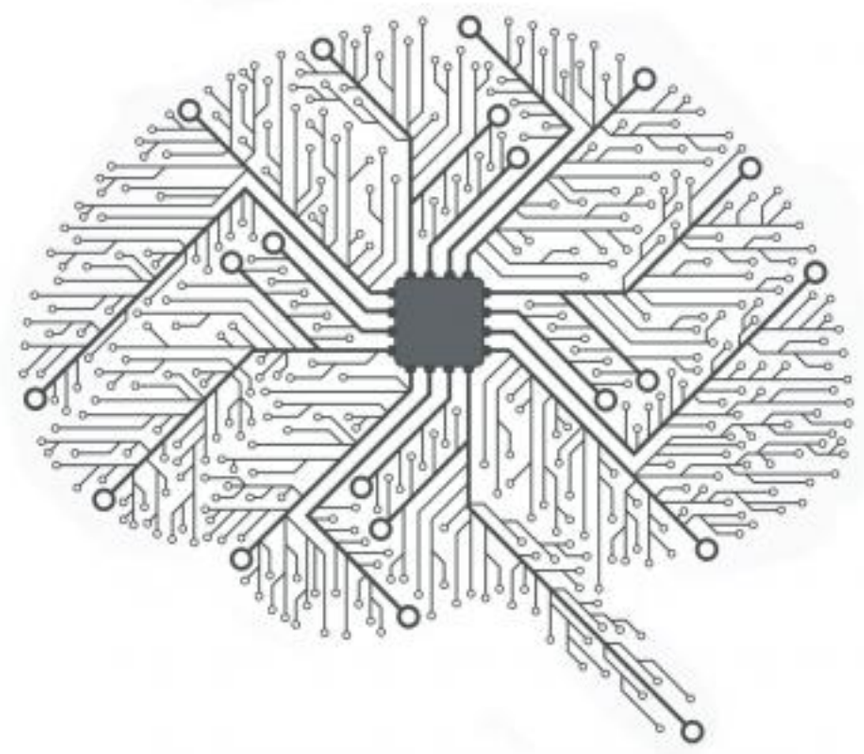
人工智能

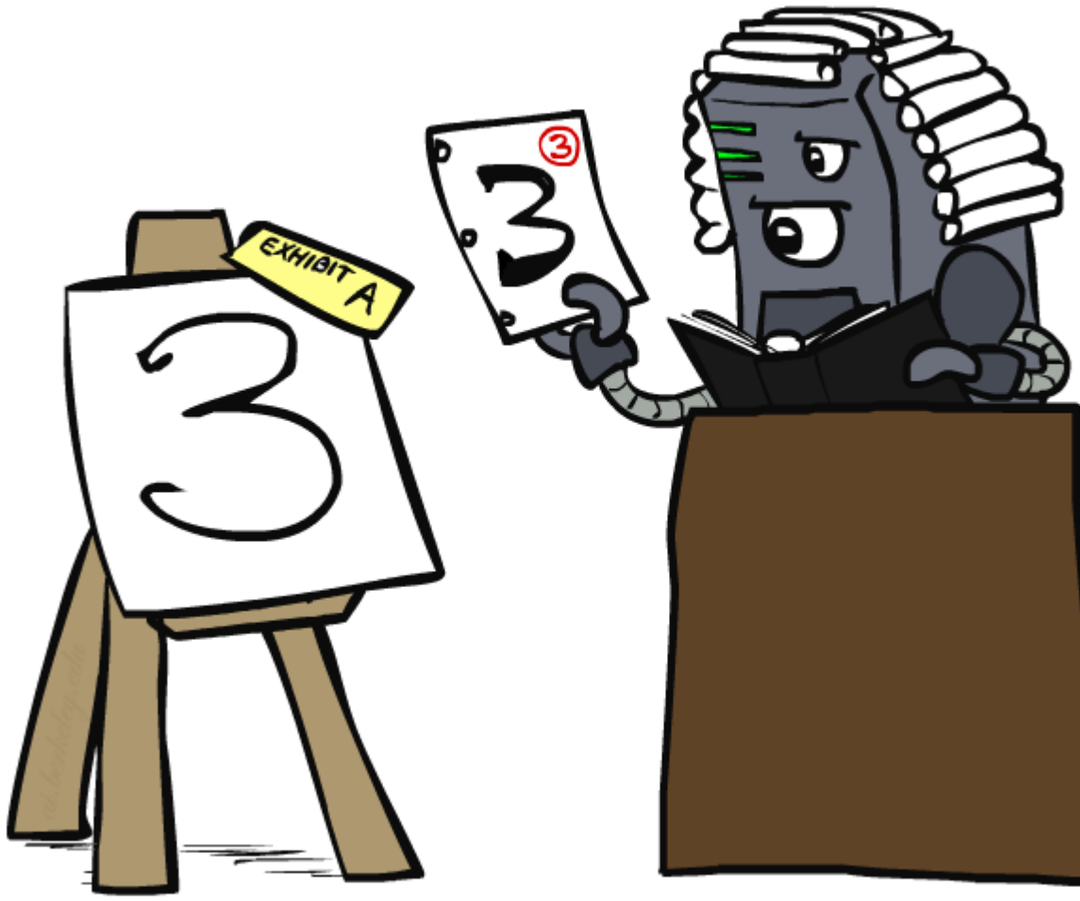


第十章·核方法和聚类

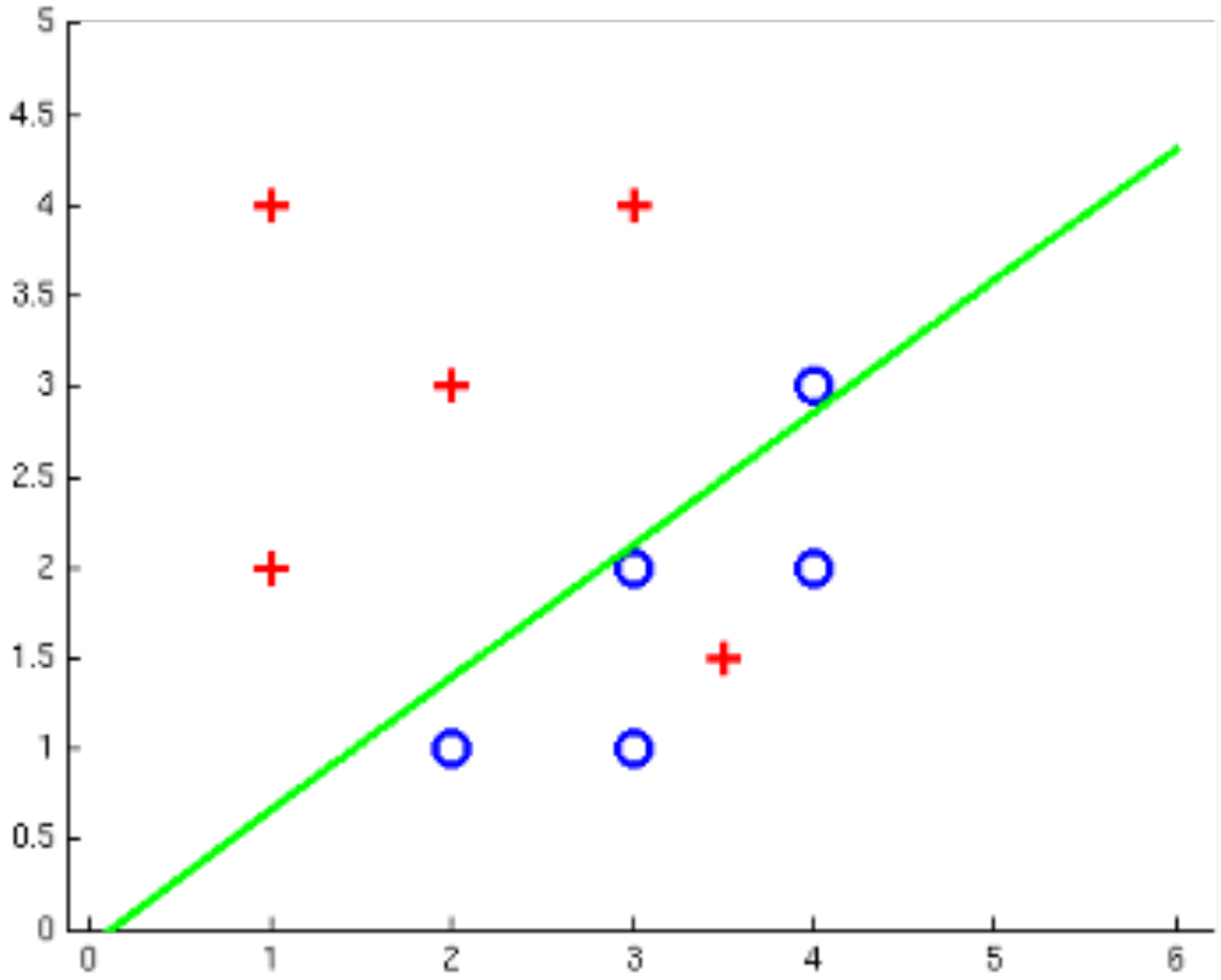
- 基于案例的学习
- 核方法
- 聚类学习



基于案例的学习

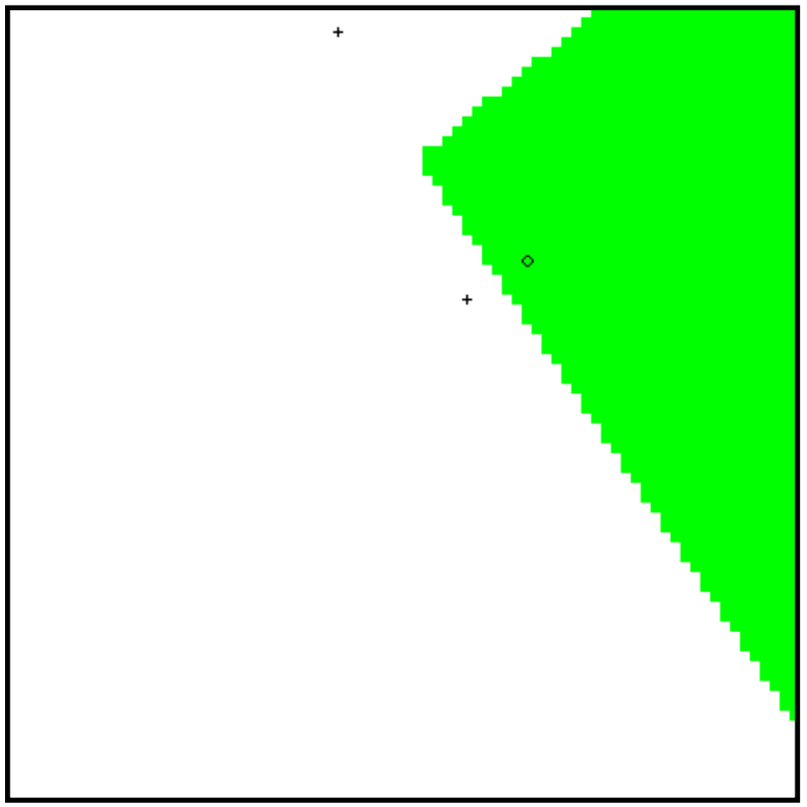


不可分离的数据



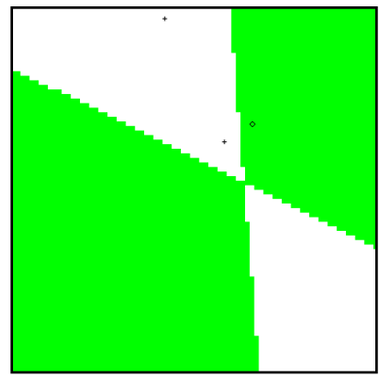
基于案例的推理

- 基于相似性进行分类
  - 基于案例的推理
  - 使用相似的实例预测实例的标签
- 最近邻(Nearest-neighbor)分类
  - 1-NN: 复制最邻近数据点的标签
  - K-NN: 基于 k个最邻近的样本进行投票(需要指定权重)
  - K值较小时邻居的相关性较强，k值较大时函数更平滑
  - 关键问题：如何定义相似性?

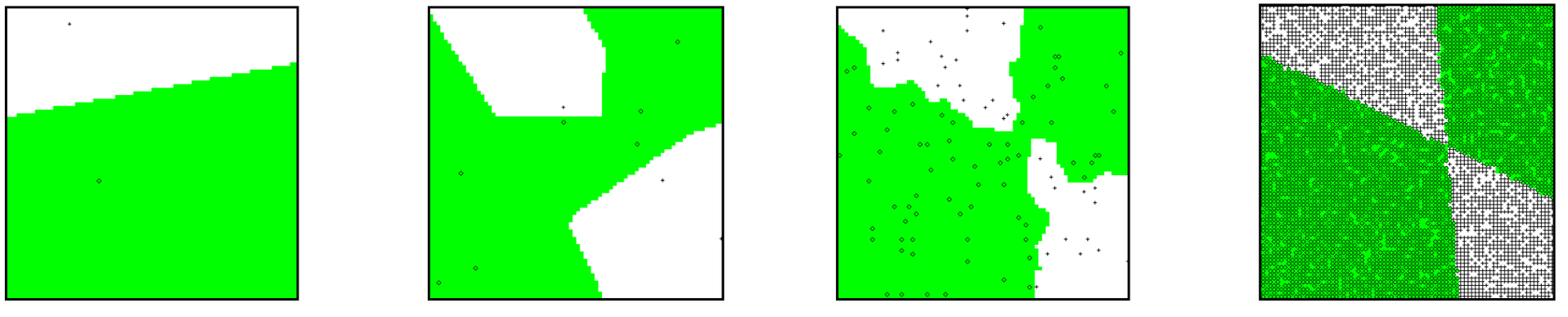


参数化/非参数化

- 参数化模型:
  - 有一组确定的参数集
  - 更多的数据意味着能够学习到更好的参数取值
- 非参数模型
  - 分类器的复杂度随着数据的增加而增加
  - 在数据量尽可能多时表现得很好，在小数据时往往较差
- (K)NN 是非参数模型



Truth



最近邻分类

- 数字图像的最近邻：
  - 拍摄新图像
  - 与所有训练图像进行比较
  - 根据最接近的例子分配标签

- 编码：图像是灰度的矢量：

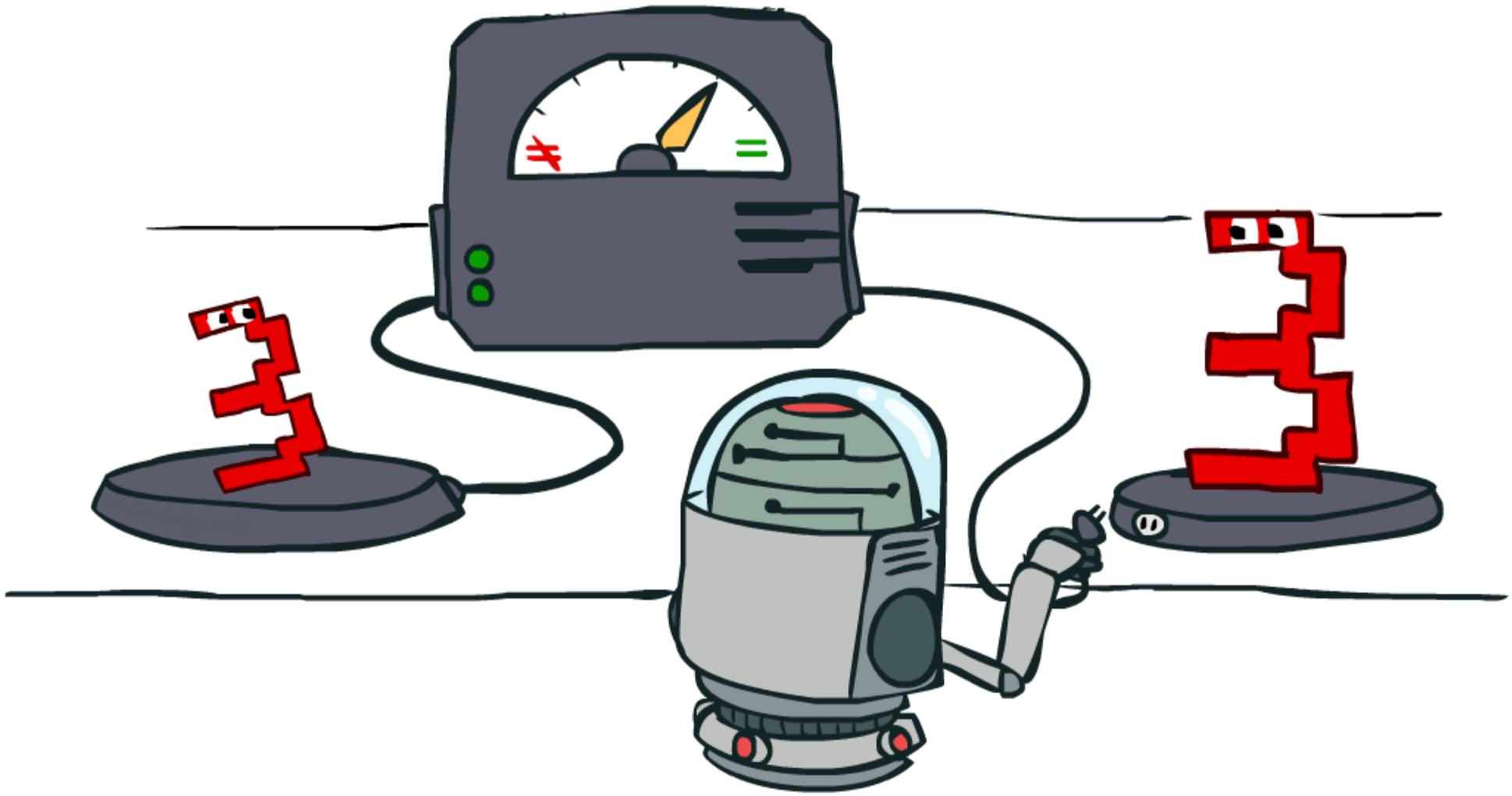
 $= \langle 0.0 \;\; 0.0 \;\; 0.3 \;\; 0.8 \;\; 0.7 \;\; 0.1 \ldots 0.0 \rangle$

- 如何定义相似函数？
  - 一种方式是利用两个图像向量的点积？

$$\text{sim}(x, x') = x \cdot x' = \sum_i x_i x'_i$$

  - 通常将向量规格化 ||x|| = 1
  - min = 0 , max = 1

相似函数



最基本的相似函数

- 最基本的相似函数是特征向量的点乘：

$$\text{sim}(x, x') = f(x) \cdot f(x') = \sum_i f_i(x) f_i(x')$$

- 如果特征只是像素：

$$\text{sim}(x, x') = x \cdot x' = \sum_i x_i x'_i$$

- 注：并非所有的相似函数都是这种形式

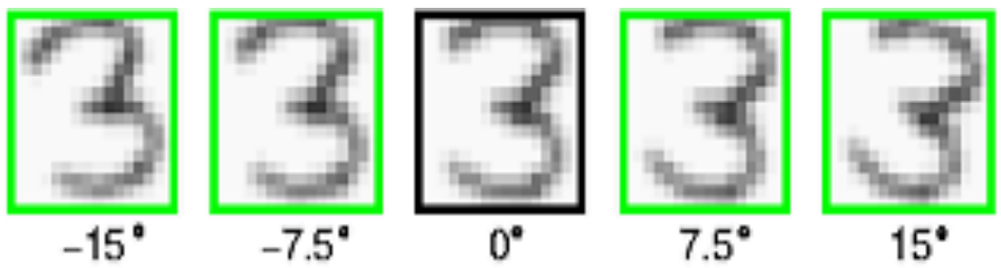
不变度量

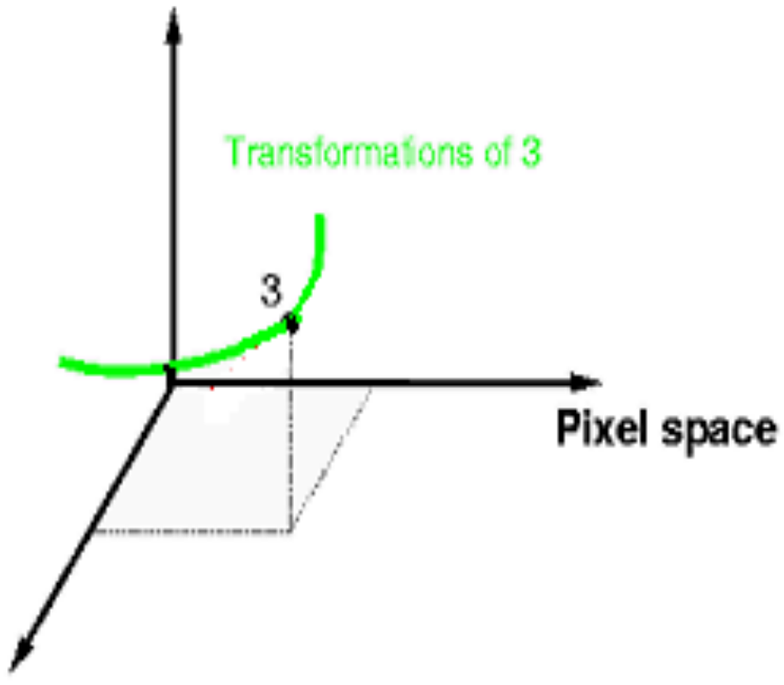
- 更好的相似性函数希望利用视觉知识
- 不变度量:
  - 相似性在某些变换下是不变的
  - 旋转、缩放、平移、笔画颜色深度...
  - 例如:

  - 16 x 16 = 256 像素; 一个像素点有 256级灰度
  - 如果直接计算像素空间的相似性，上述两幅图像的相似度会非常低
- 如何将这种不变性与相似性结合起来?

旋转不变度量

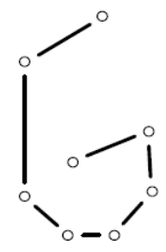
- 加入旋转不变函数r(s)，把每个样本映射到空间中的一个曲线
- 旋转不变相似性：
  - s’=max s(r()，r())
- s’存在的问题：
  - 难以计算
  - 最大允许多大的旋转?
    - 6 -> 9 ??!!



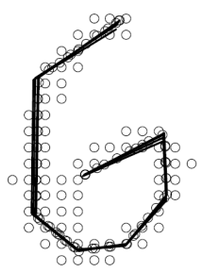
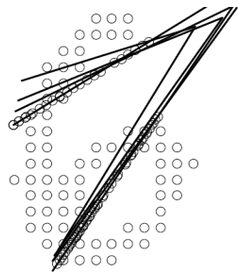
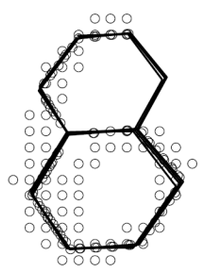
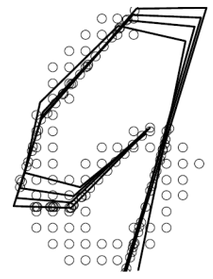


模板变形

- 可变形模板:
  - 为每个类别给定一个“理想”版本
  - 使用最小“变化”来计算图像的最佳拟合



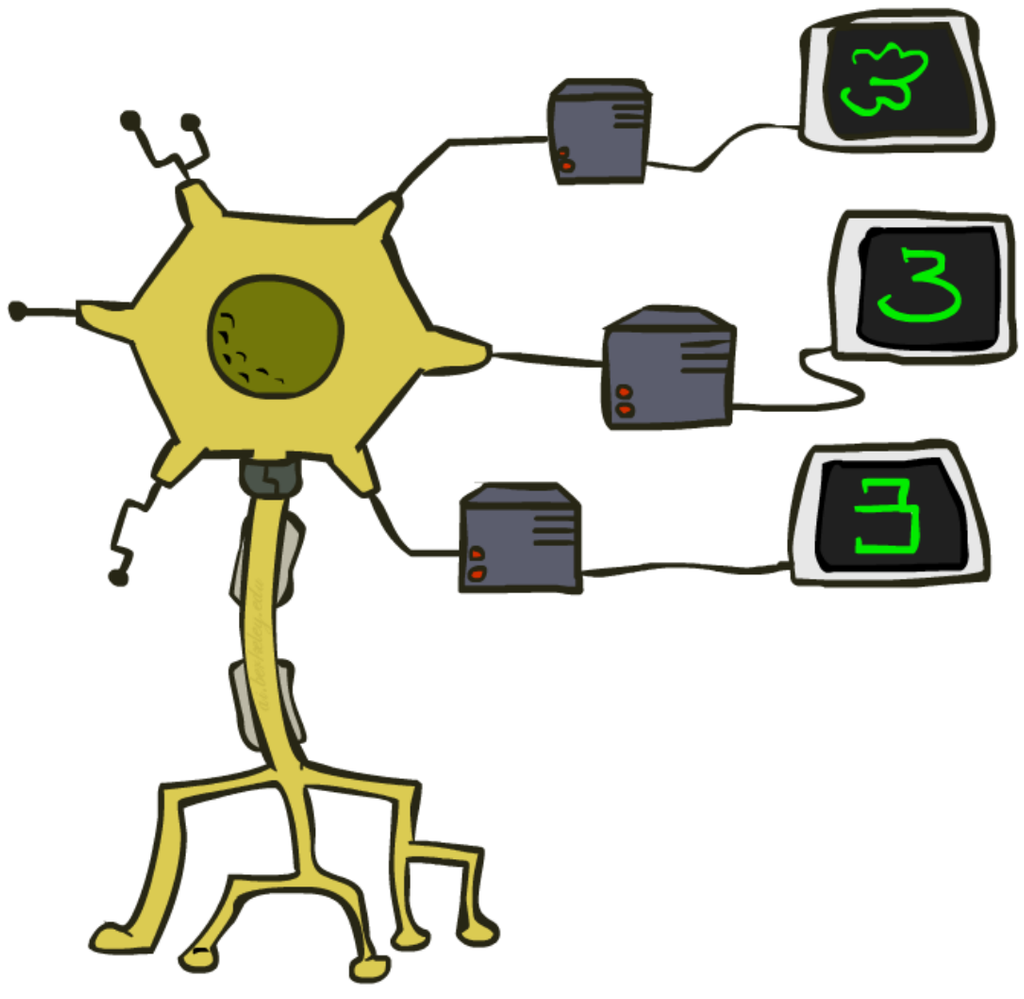
- 在许多商业数字识别器中使用

两种方式的比较

- 基于邻近样本的方法
  - 可以使用相似函数
  - 实际上不需要进行显式学习
- 类感知机方法
  - 显式训练以减少经验误差
  - 不能使用相似性，只能使用特征
  - 或者可以吗？让我们来看看！

核化



感知机权重

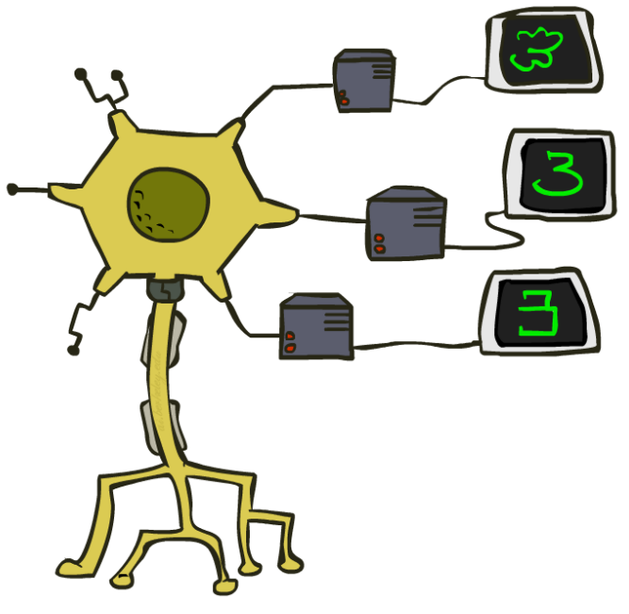
- 感知机的权重w_y的最终值是多少?
  - 权重w_y能是任何实向量吗?
  - 不能！它是通过将输入相加而建立起来的。

$$w_y = \mathbf{0} + f(x_1) - f(x_5) + \ldots$$

$$w_y = \sum_i \alpha_{i,y} f(x_i)$$

- 可以根据更新计数重建权重向量

$$\alpha_y = \langle \alpha_{1,y} \;\; \alpha_{2,y} \;\; \ldots \;\; \alpha_{n,y} \rangle$$



对偶感知机

- 如何分类一个新的样本x?

$$\begin{aligned}
\text{score}(y, x) &= w_y \cdot f(x) \\
&= \left(\sum_i \alpha_{i,y} f(x_i) \right) \cdot f(x) \\
&= \sum_i \alpha_{i,y} \left(f(x_i) \cdot f(x) \right) \\
&= \sum_i \alpha_{i,y} \; K(x_i, x)
\end{aligned}$$

- 如果有人告诉我们每对样本的K值，就不需要构建权重向量（或特征向量）！

对偶感知机

- 从零计数（alpha）开始
- 逐个挑选训练样本
- 尝试对x_n进行分类，$y = \arg\max_y \sum_i \alpha_{i,y} K(x_i, x_n)$
- 如果正确，不做改变!
- 如果错误：减少错误类的计数，增加正确类的计数

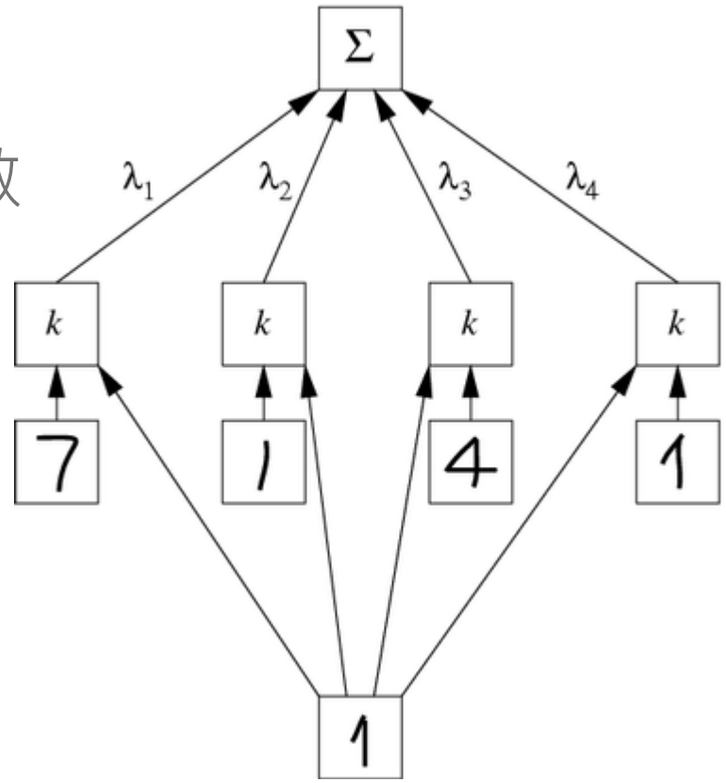
$$\alpha_{y,n} = \alpha_{y,n} - 1 \qquad w_y = w_y - f(x_n)$$

$$\alpha_{y^*,n} = \alpha_{y^*,n} + 1 \qquad w_{y^*} = w_{y^*} + f(x_n)$$

核化感知机

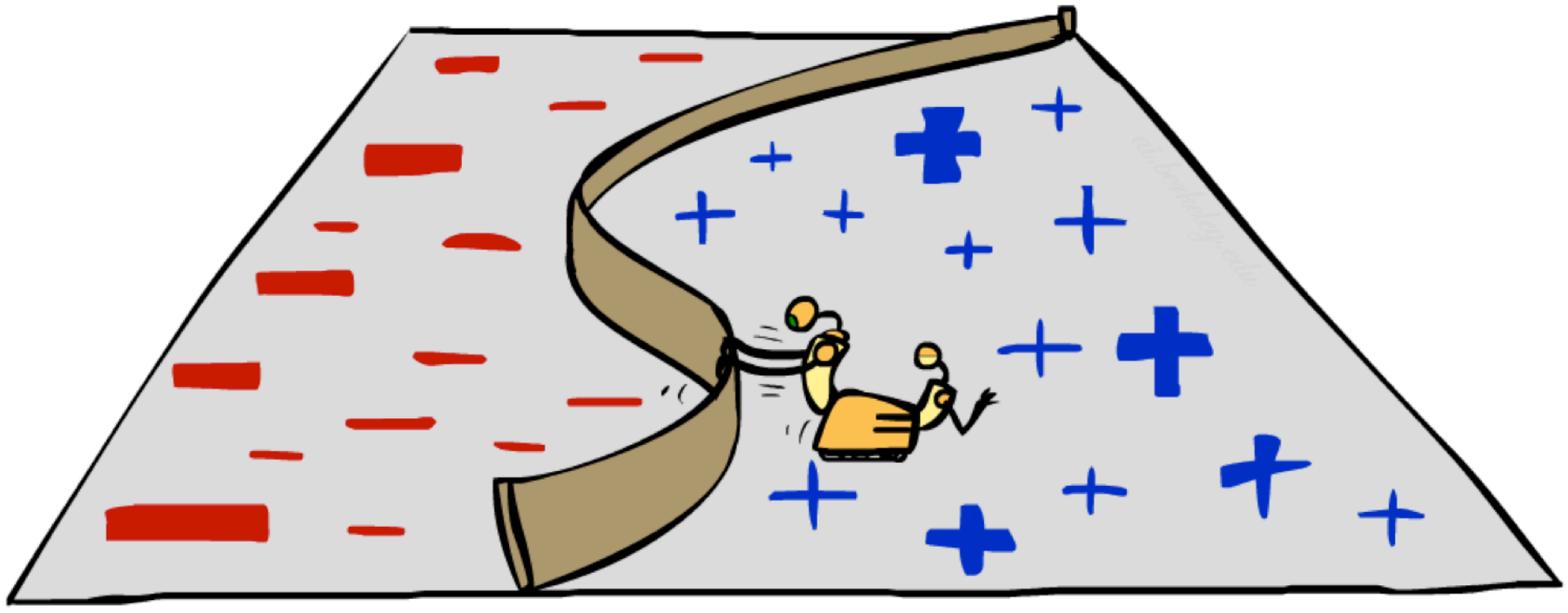
- 如果我们有一个黑匣子（核）K告诉我们两个例子x和x'的点积：
  - 可以用对偶表示法
  - 不需要使用点乘，而是使用任意的相似性度量函数

$$\begin{aligned}\text{score}(y, x) &= w_y \cdot f(x) \\ &= \sum_i \alpha_{i,y} \; K(x_i, x)\end{aligned}$$



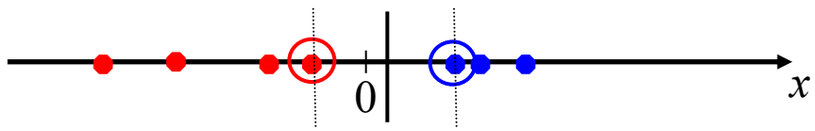
- 像最邻近算法一样---使用（黑盒）相似性进行学习
- 缺点：如果很多例子的alpha值为非零，则速度较慢

非线性

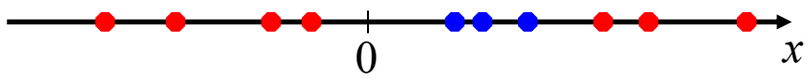


非线性分离器

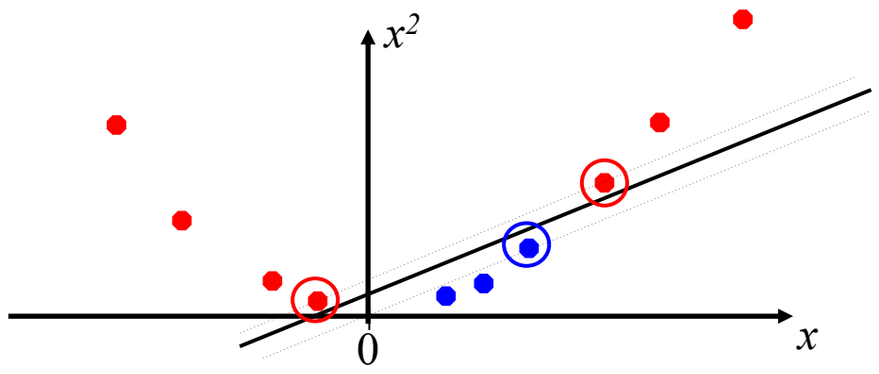
- 线性决策规则对线性可分离的数据非常有效：



- 但是有些数据很难进行线性分离：

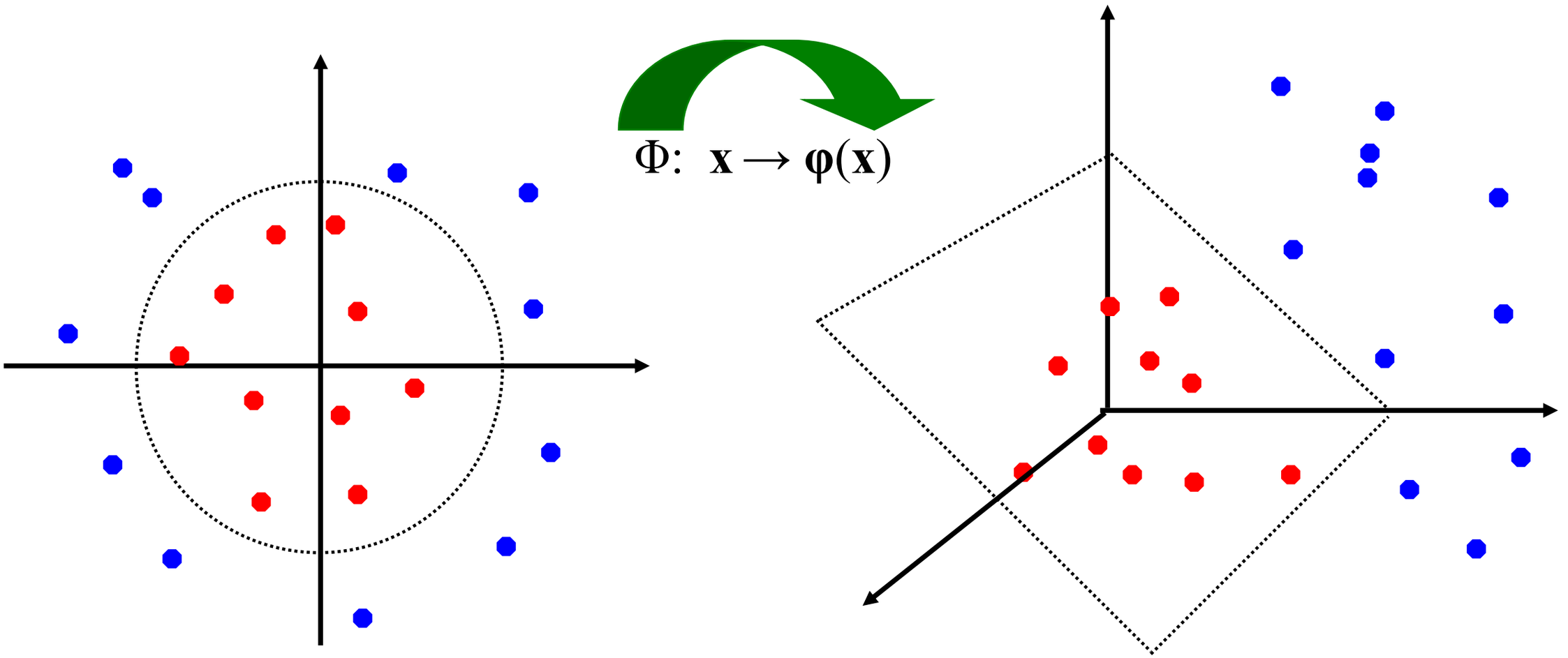


- 是否可以将数据映射到更高维空间?



非线性分离器

- 总体思路：原始特征空间可以映射到训练集可分离的高维特征空间：



一些核

- 核隐式地将原始向量映射到高维空间，在那里取点积，然后将结果返回
- 线性核：$K(x, x') = x \cdot x' = \sum_i x_i \, x'_i$
- 二次核：$K(x, x') = (x \cdot x' + 1)^2$

$$= \sum_{i,j} x_i x_j \, x'_i x'_j + 2 \sum_i x_i \, x'_i + 1$$

- 高斯核: 无限维表示

$$K(x, x') = \exp(-||x - x'||^2)$$

- 离散核：例如字符串核

多项式核

- 多项式核的定义为：

$$K(x, y) = (x^\mathsf{T} y + c)^d$$

- 作为核，K对应于基于某种映射φ的特征空间中的内积：

$$K(x, y) = \langle \varphi(x), \varphi(y) \rangle$$

- 设想一种简单的情况，d=2，对其展开：

$$K(x, y) = \left(\sum_{i=1}^{n} x_i y_i + c \right)^2 = \sum_{i=1}^{n} \left(x_i^2 \right) \left(y_i^2 \right) + \sum_{i=2}^{n} \sum_{j=1}^{i-1} \left(\sqrt{2} x_i x_j \right) \left(\sqrt{2} y_i y_j \right) + \sum_{i=1}^{n} \left(\sqrt{2c} x_i \right) \left(\sqrt{2c} y_i \right) + c^2$$

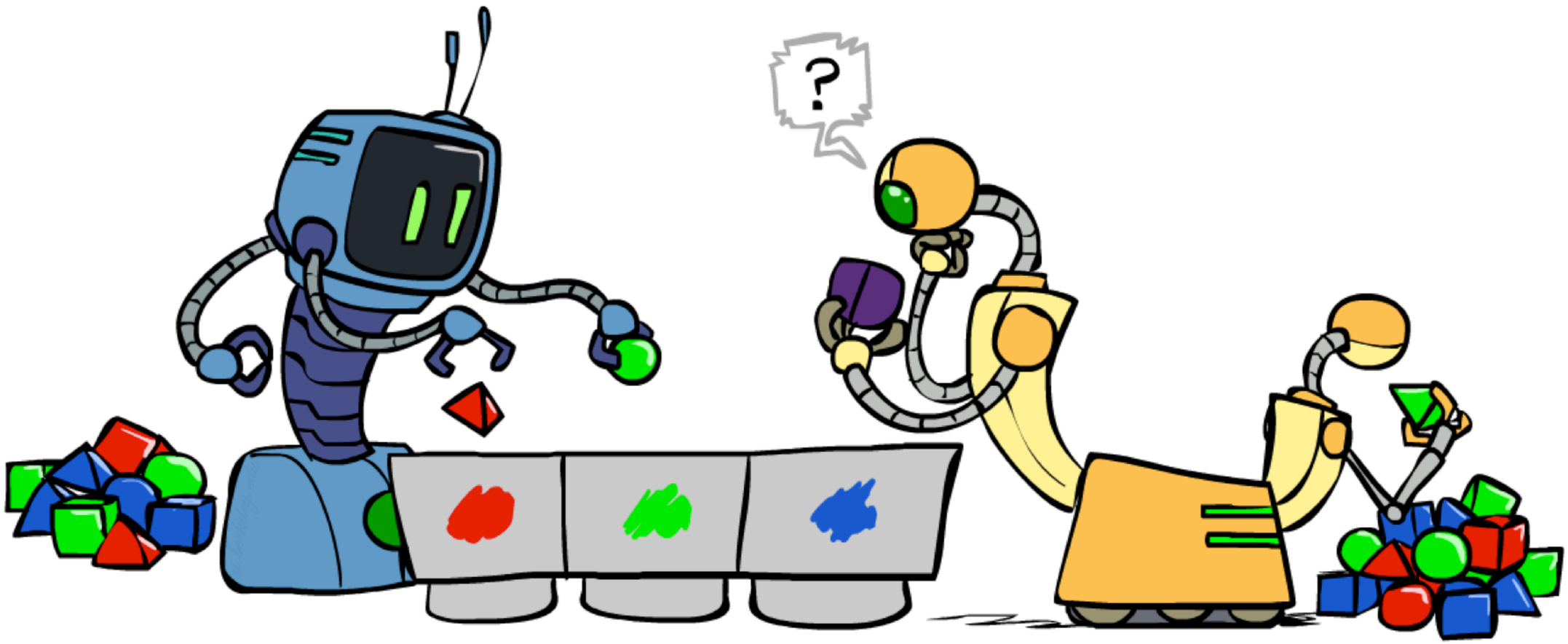
- 由此可知，其对应的映射φ如下所示：

$$\varphi(x) = \langle x_n^2, \ldots, x_1^2, \sqrt{2} x_n x_{n-1}, \ldots, \sqrt{2} x_n x_1, \sqrt{2} x_{n-1} x_{n-2}, \ldots, \sqrt{2} x_{n-1} x_1, \ldots, \sqrt{2} x_2 x_1, \sqrt{2c} x_n, \ldots, \sqrt{2c} x_1, c \rangle$$

为什么使用核

- 不能自己添加这些特征吗（例如，添加所有成对的特征，而不是使用二次核函数）？
  - 是的，原则上可以
  - 无需修改任何算法
  - 但是，特征的数量可能会变大（或无限）
- 核让我们用这些特性进行隐式计算
  - 例如：二次核中的隐式点积在每个点积上占用的空间和时间要少得多
  - 使用纯对偶算法是有代价的：需要计算每个训练数据的相似性

聚类



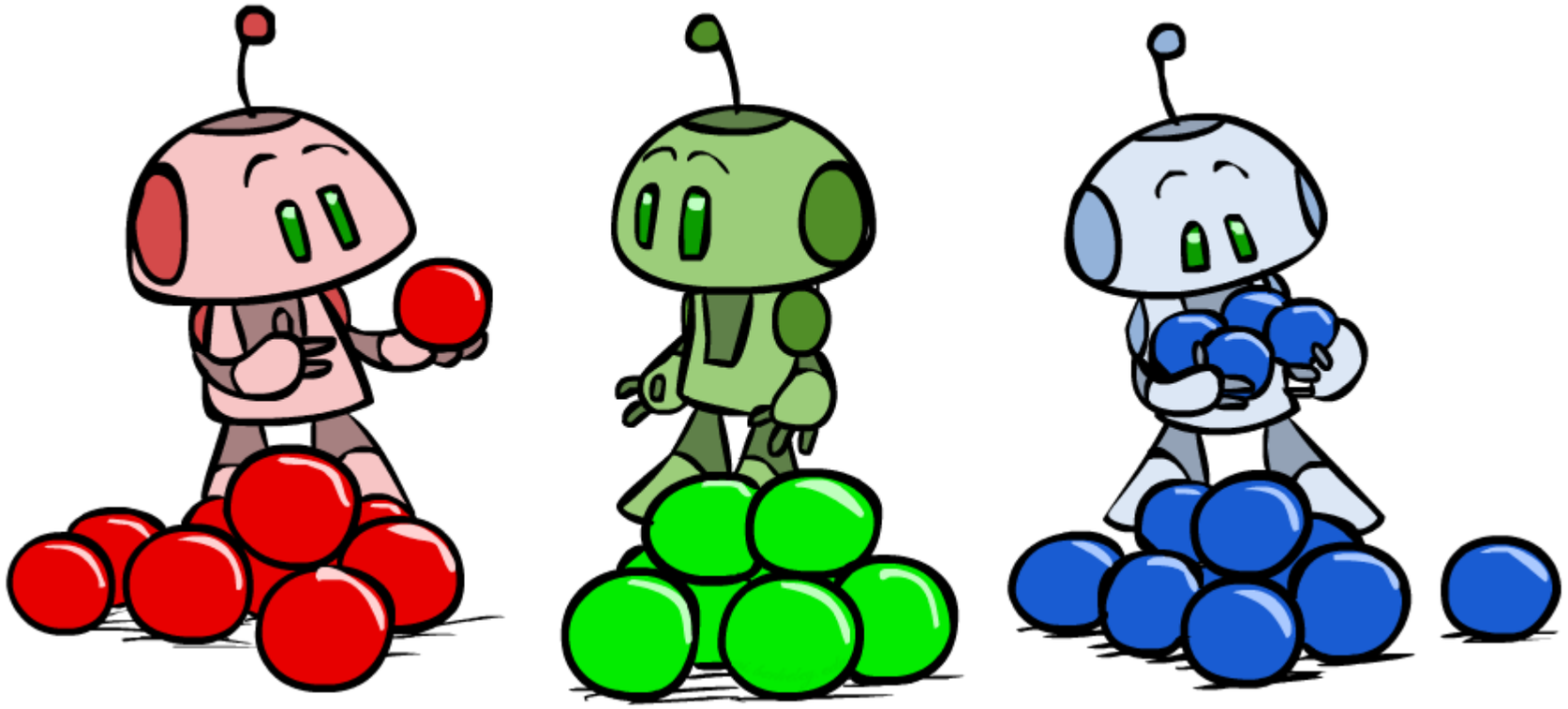
聚类

- 基本思想：将相似的实例组合在一起
- 示例：二维点模式

- “相似”是什么意思?
  - 一种度量方式：欧几里德距离

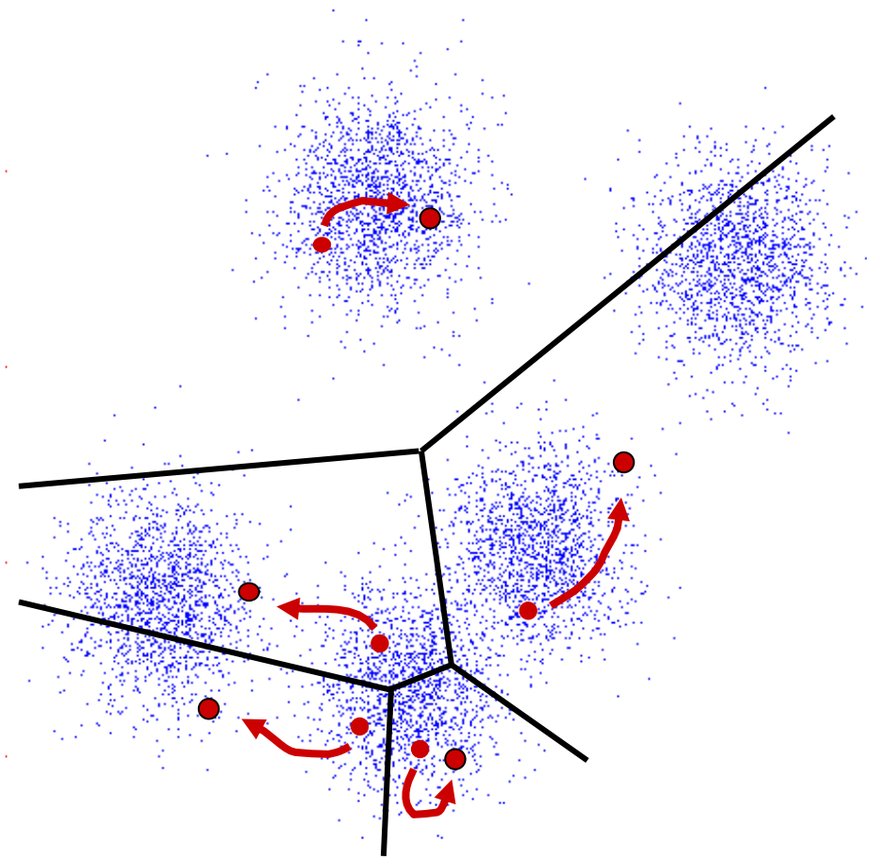
$$\text{dist}(x, y) = (x - y)^{\mathsf{T}}(x - y) = \sum_i (x_i - y_i)^2$$

K-Means 算法

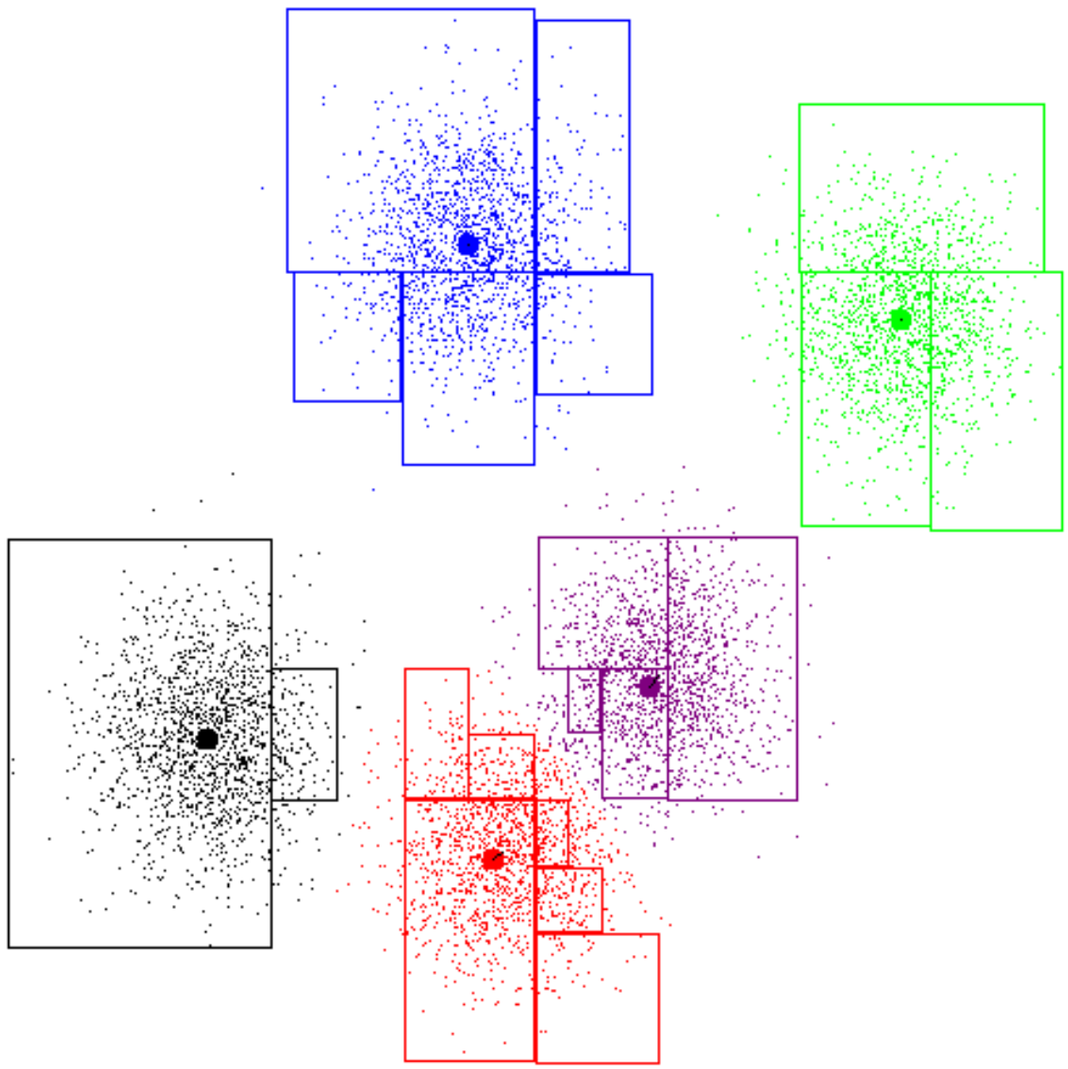


K-Means 算法

- 一种迭代聚类算法
  - 选取K个随机点作为聚类中心（means）
  - 交替执行：
    - 将数据样本分配给最接近的平均值
    - 将每个平均值更新为给其分配的数据样本的平均值
- 当没有数据样本的分配改变时停止



K-Means 示例

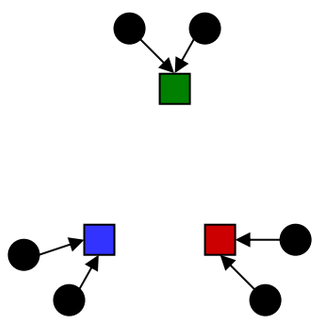


从函数优化的角度看K-Means

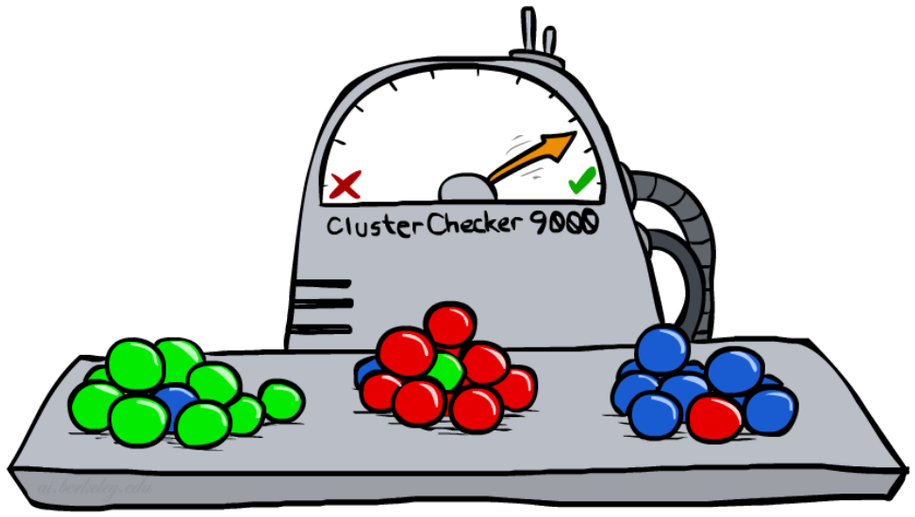
- 考虑到平均值的总距离：

$$\phi(\{x_i\},\{a_i\},\{c_k\}) = \sum_i \text{dist}(x_i, c_{a_i})$$

points　assignments　means



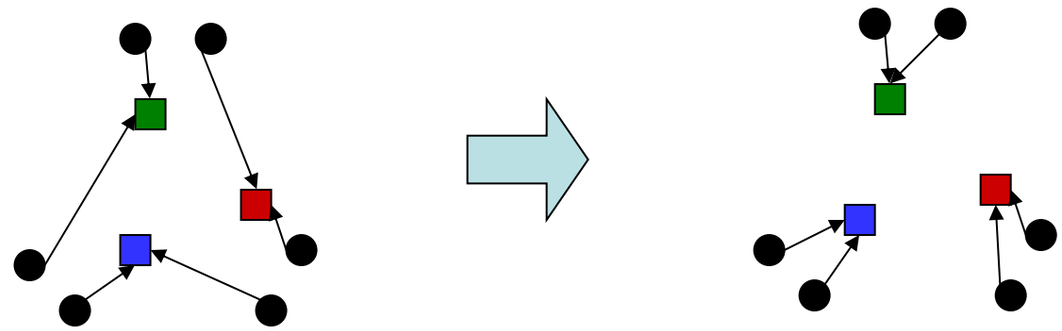
- 每次迭代都会降低ϕ
- 每个迭代有两个阶段：
  - 更新分配：固定中心 c，改变分配 a
  - 更新中心：固定分配 a，改变中心 c



第一阶段：更新分配

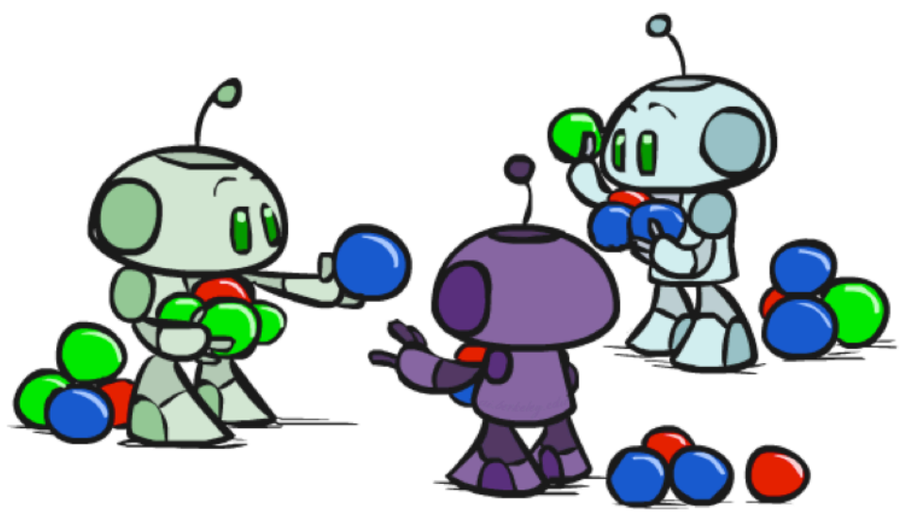
- 对于每个点，重新分配到最接近的平均值：

$$a_i = \underset{k}{\text{argmin}}\, \text{dist}(x_i, c_k)$$



- 只能降低总距离ϕ!

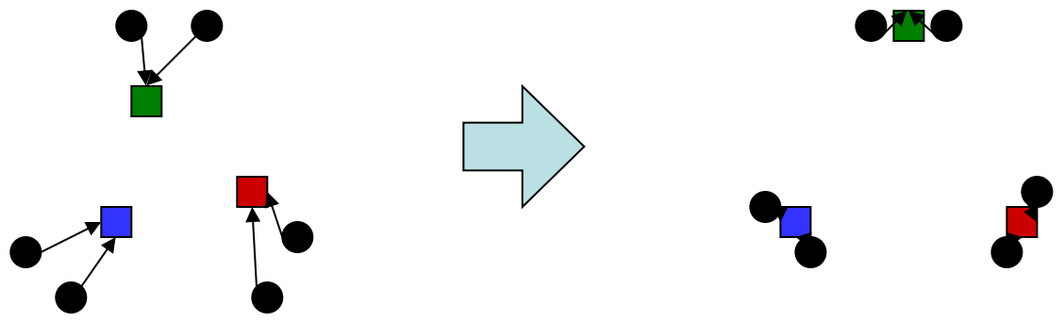
$$\phi(\{x_i\}, \{a_i\}, \{c_k\}) = \sum_i \text{dist}(x_i, c_{a_i})$$



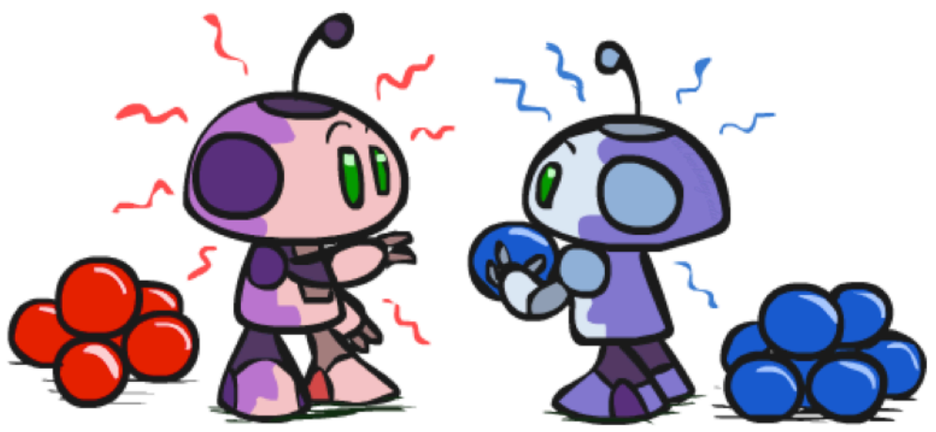
第二阶段：更新中心

- 将每个中心移动到其分配点的平均值：

$$c_k = \frac{1}{|\{i : a_i = k\}|} \sum_{i:a_i=k} x_i$$



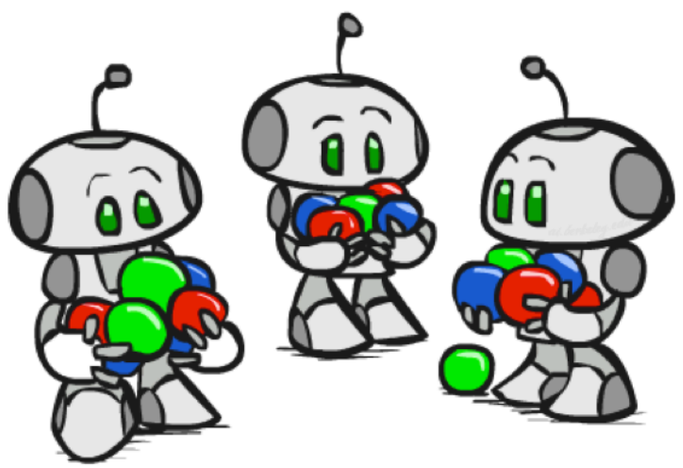
- 也只能减少总距离…（为什么？）
  - 与一组点{x}有最小平方欧几里德距离的点y是它们的平均值



初始化

- K-means是不确定的
  - 需要初始中心
  - 中心的初始化对于结果很重要!

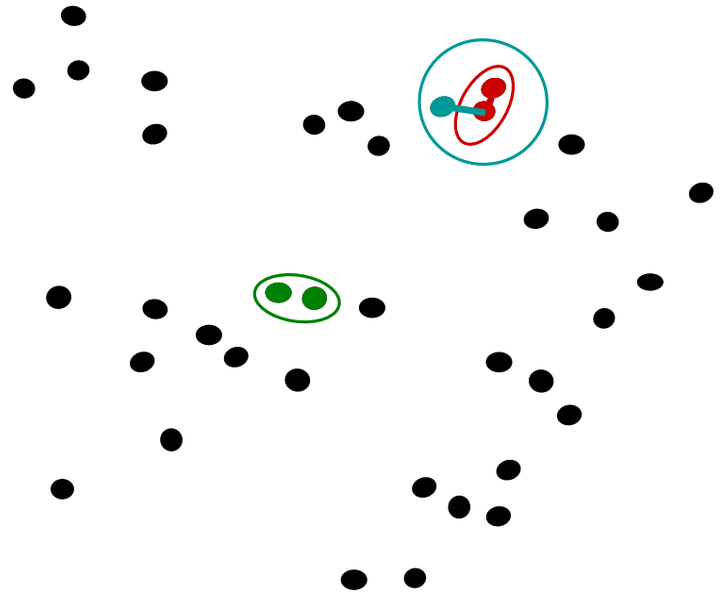
- 改进方法:
  - 基于方差的分割/合并
  - 启发式初始化方法

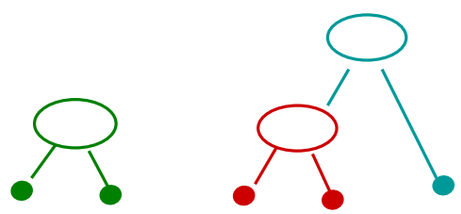


示例：K-means陷入局部最优

自底向上的聚类

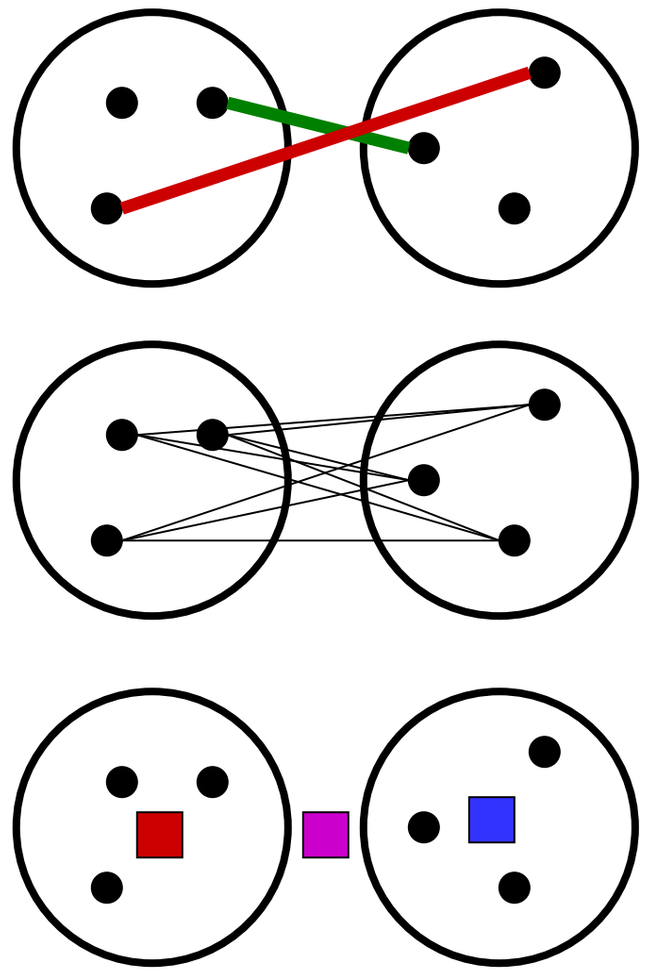
- 自底向上的聚类
  - 首先合并非常相似的实例
  - 从较小的集群增量构建更大的集群
- 算法：
  - 维护一组簇
  - 最初，每个实例都在自己的簇中
  - 重复：
    - 选择两个最近的簇
    - 将它们合并到一个新的簇中
    - 当只剩下一个簇时停止
- 产生的不是一个聚类，而是一个由树状图表示的聚类族





自底向上的聚类

- 如何为具有多个元素的簇定义“最近”?
- 多种可行的度量方法
  - 最近对（单链路群集）
  - 最远对（完整链路群集）
  - 所有对的平均值
  - 沃德法（最小方差，如k均值）
- 不同的选择会产生不同的聚类行为



示例：Google新闻

新闻类别：分类算法

新闻分组：聚类算法