# Machine Learning (CSE 446): Geometry: Nearest Neighbors and $K$-means

(Ridiculously simple geom. approaches)

Sham M Kakade



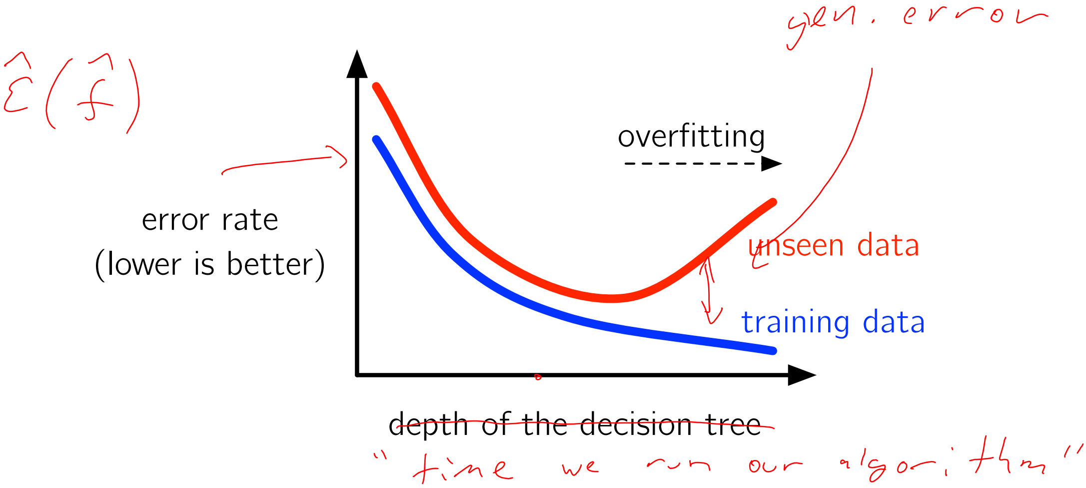
University of Washington
cse446-staff@cs.washington.edu

Remember:

HW1 due next week.

# Review

# Danger: Overfitting



# Estimating your Error and Some help in avoiding overfitting

- Test set: use these to estimate the performance of your learning algorithm. The cardinal rule of machine learning: **Don't touch your test data.**
- Dev set: use this during learning to help avoid overfitting. If we have **hyperparameters** (like depth, width), we can **tune** these on **development data**.

# One limit of learning: Inductive Bias

- Just as *you* had a tendency to focus on a certain type of function $f$, you want your algorithm to be "biased" towards the correct classifier (so that it can learn with a "small" number of examples).

- BUT remember there is "no free lunch": this "bias" means you must do worse on other (hopefully unrealistic) problems.

# Another Limit of Learning: The Bayes Optimal Classifier

Pr (x, y)

$$f^{(\mathsf{BO})}(x) = \operatorname*{argmax}_{y} \mathcal{D}(x, y)$$

$\in \{0, 1\}$

**Theorem:** The Bayes optimal classifier achieves minimal expected classification (e.g. zero/one) error ($\ell(y, \hat{y}) = [\![y \neq \hat{y}]\!]$) of any deterministic classifier.

See CIML and lecture notes for proof.

# Today

# Features

Data derived from `https://archive.ics.uci.edu/ml/datasets/Auto+MPG`

| mpg; | cylinders; | displacement; | horsepower; | weight; | acceleration; | year; | origin |
|---|---|---|---|---|---|---|---|
| 18.0 | 8 | 307.0 | 130.0 | 3504. | 12.0 | 70 | 1 |
| 15.0 | 8 | 350.0 | 165.0 | 3693. | 11.5 | 70 | 1 |
| 18.0 | 8 | 318.0 | 150.0 | 3436. | 11.0 | 70 | 1 |
| 16.0 | 8 | 304.0 | 150.0 | 3433. | 12.0 | 70 | 1 |
| 17.0 | 8 | 302.0 | 140.0 | 3449. | 10.5 | 70 | 1 |
| 15.0 | 8 | 429.0 | 198.0 | 4341. | 10.0 | 70 | 1 |
| 14.0 | 8 | 454.0 | 220.0 | 4354. | 9.0 | 70 | 1 |
| 14.0 | 8 | 440.0 | 215.0 | 4312. | 8.5 | 70 | 1 |
| 14.0 | 8 | 455.0 | 225.0 | 4425. | 10.0 | 70 | 1 |
| 15.0 | 8 | 390.0 | 190.0 | 3850. | 8.5 | 70 | 1 |
| 15.0 | 8 | 383.0 | 170.0 | 3563. | 10.0 | 70 | 1 |
| 14.0 | 8 | 340.0 | 160.0 | 3609. | 8.0 | 70 | 1 |
| 15.0 | 8 | 400.0 | 150.0 | 3761. | 9.5 | 70 | 1 |
| 14.0 | 8 | 455.0 | 225.0 | 3086. | 10.0 | 70 | 1 |
| 24.0 | 4 | 113.0 | 95.00 | 2372. | 15.0 | 70 | 3 |
| 22.0 | 6 | 198.0 | 95.00 | 2833. | 15.5 | 70 | 1 |
| 18.0 | 6 | 199.0 | 97.00 | 2774. | 15.5 | 70 | 1 |
| 21.0 | 6 | 200.0 | 85.00 | 2587. | 16.0 | 70 | 1 |
| 27.0 | 4 | 97.00 | 88.00 | 2130. | 14.5 | 70 | 3 |
| 26.0 | 4 | 97.00 | 46.00 | 1835. | 20.5 | 70 | 2 |
| 25.0 | 4 | 110.0 | 87.00 | 2672. | 17.5 | 70 | 2 |
| 24.0 | 4 | 107.0 | 90.00 | 2430. | 14.5 | 70 | 2 |

- All features are really represented as real values. (they are really "tuples")
- The "1–2–3" values suggest ordinality, which is misleading.
- Side note: can convert discrete origin feature into three binary features as follows:

$$1/\text{america} \to (1, 0, 0)$$
$$2/\text{europe} \to (0, 1, 0)$$
$$3/\text{asia} \to (0, 0, 1)$$

# Instance $x$ Becomes Vector $\mathbf{x}$

First example in the data, "Chevrolet Chevelle Malibu," becomes:

$$[8, 307.0, 130.0, 3504, 12.0, 70, 1, 0, 0]$$

"Buick Skylark 320" becomes:

$$[8, 350.0, 165.0, 3693, 11.5, 70, 1, 0, 0]$$

# Euclidean Distance

$x = (x[1], x[2], \ldots x[d])$
$x' = \ldots$

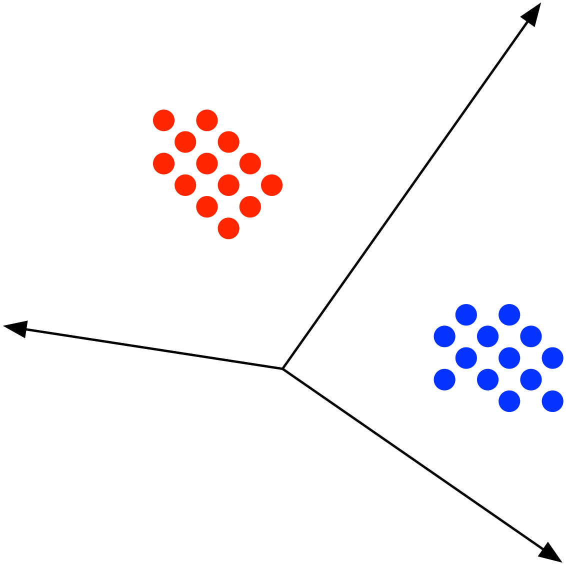
General formula for the Euclidean distance between two $d$-length vectors:

$$dist(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{j=1}^{d} (\mathbf{x}[j] - \mathbf{x}'[j])^2}$$
$$= \|\mathbf{x} - \mathbf{x}'\|_2$$

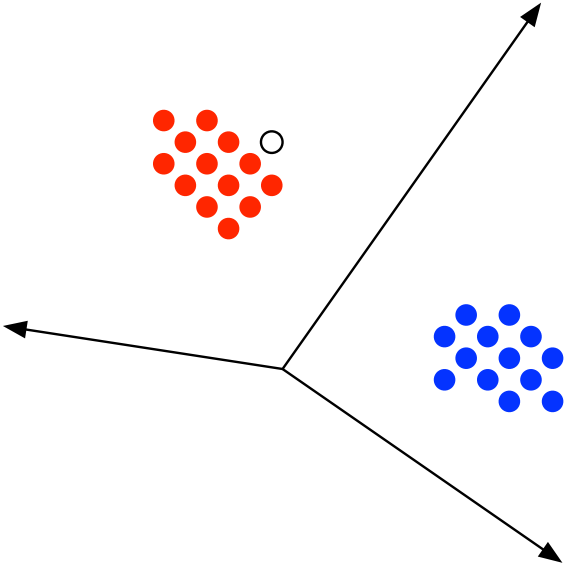
The distance between the Chevrolet Chevelle Malibu and the Buick Skylark 320:

$$\sqrt{\begin{array}{c}(8-8)^2 + (307-350)^2 + (130-165)^2 + (3504-3693)^2 \\ +(12-11.5)^2 + (70-70)^2 + (1-1)^2 + (0-0)^2 + (0-0)^2\end{array}}$$
$$= \sqrt{1849 + 1225 + 35721 + 0.25}$$
$$\approx 196.965$$

# Training Data in $\mathbb{R}^d$



# Classifying a New Example in $\mathbb{R}^d$



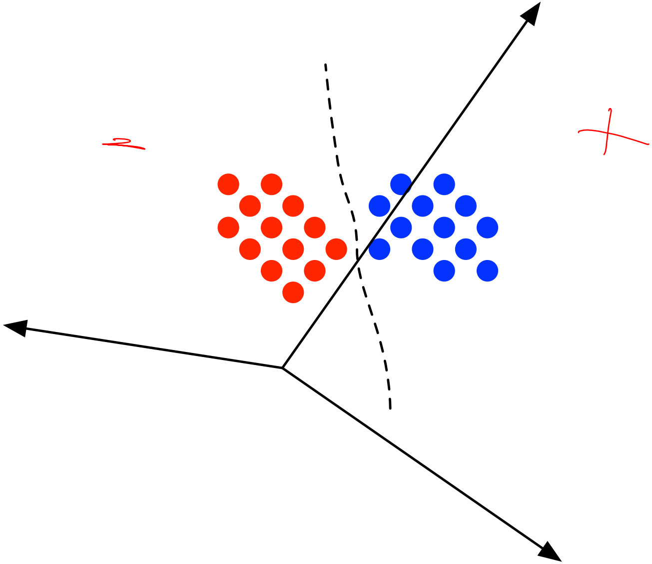
# Nearest Neighbor Classifier

label x with the same label as its nearest training input

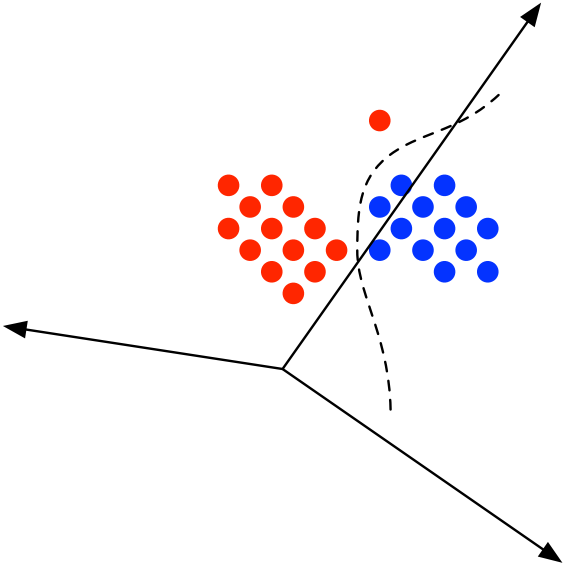
**Data**: training data $D = \langle (\mathbf{x}_n, y_n) \rangle_{n=1}^{N}$, input $\mathbf{x}$
**Result**: predicted class
let $n^* = \underset{n \in \{1,\ldots,N\}}{\operatorname{argmin}} \; \mathit{dist}\,(\mathbf{x}_n, \mathbf{x})$;
return $y_{n^*}$;

**Algorithm 1:** NNTest

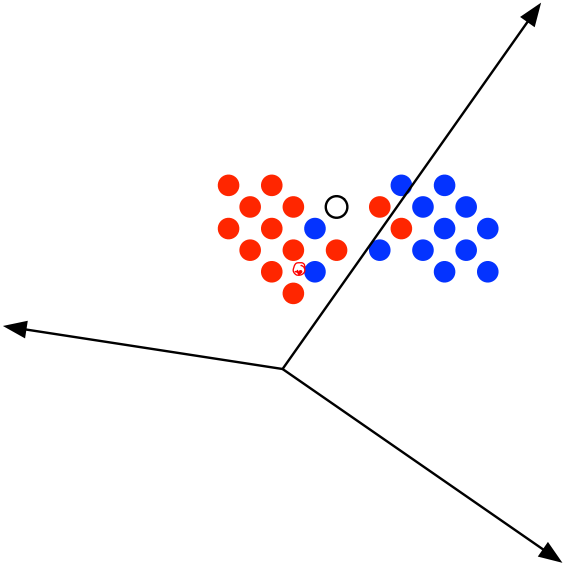
# Concept: Decision Boundary



# Concept: Decision Boundary



# Classifying a New Example in $\mathbb{R}^d$



# $K$-Nearest Neighbors Classifier

**Data**: training data $D = \langle(\mathbf{x}_n, y_n)\rangle_{n=1}^{N}$, input $\mathbf{x}$
**Result**: predicted class
$S = \emptyset$;
**for** $n \in \{1, \ldots, N\}$ **do**
  $S = S \cup \{(dist(\mathbf{x}_n, \mathbf{x}), y_n)\}$;
**end**
# sort on distances
$L = \textsc{sort}(S)$;
return $\textsc{majorityClass}(L[1], \ldots, L[K])$;

**Algorithm 2:** $\textsc{KNNTest}$

classify x based on the majority vote of x's nearest neighbors

# $K$-Nearest Neighbors: Comments

- Inductive Bias:
  - Neighbors have the same label; classes align to contiguous "regions" in feature space.
  - All features are equally important.
- What is the training error of 1-NN? 0
- Should $K$ be even or odd?
- What about high dimensions? ← read CIML

# Detour: Unsupervised Learning

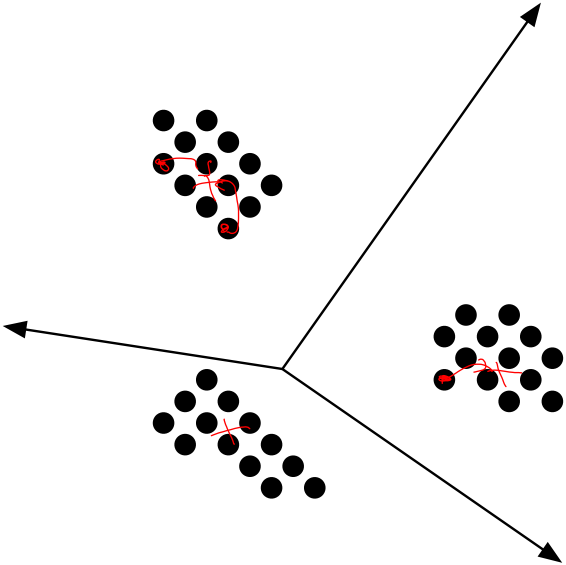
# Unsupervised Learning

The training dataset consists only of $\langle \mathbf{x}_n \rangle_{n=1}^{N}$.

There might, or might not, be a test set with correct classes $y$.

Simplest kind of unsupervised learning: cluster into $K$ groups.

# How should we cluster the points?

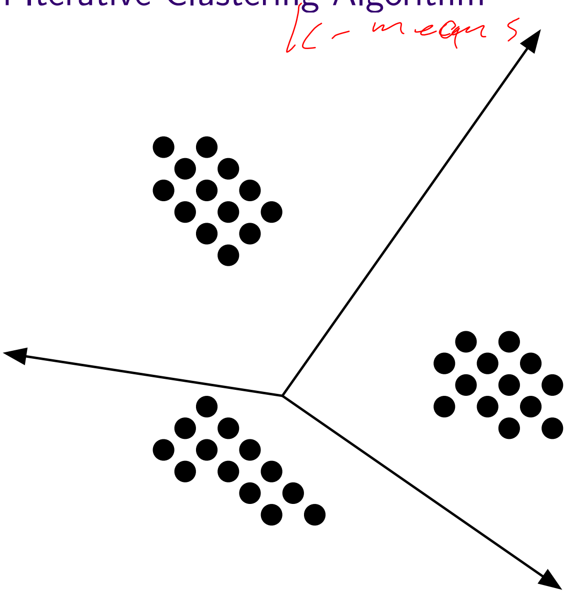


"chicken-egg" problem:

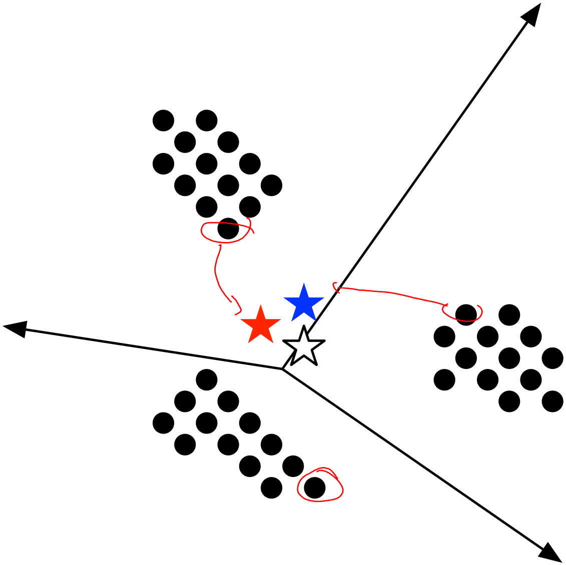
- If we know which points are grouped together, then easy to figure out the mean of any cluster.
- If we know the means of the clusters, then easy to group the points together.

How do we cluster our points?

# An Iterative Clustering Algorithm

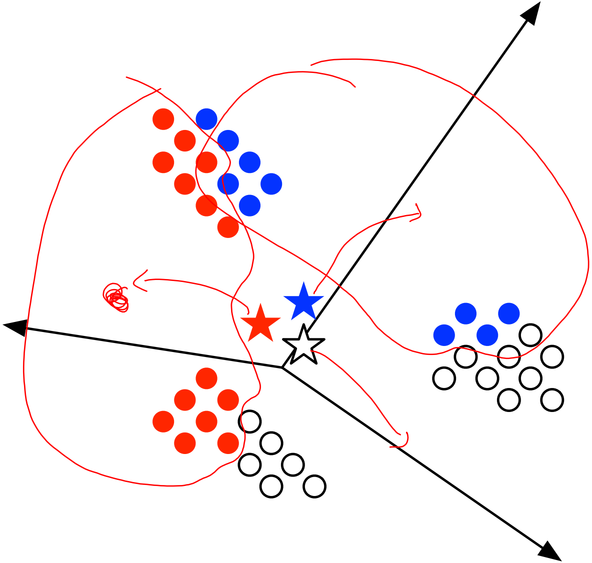


# An Iterative Clustering Algorithm



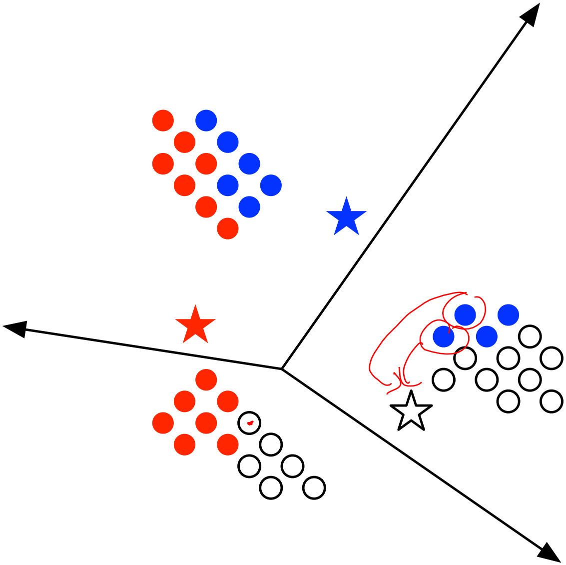
The stars are **cluster centers**, randomly initialized.

# An Iterative Clustering Algorithm



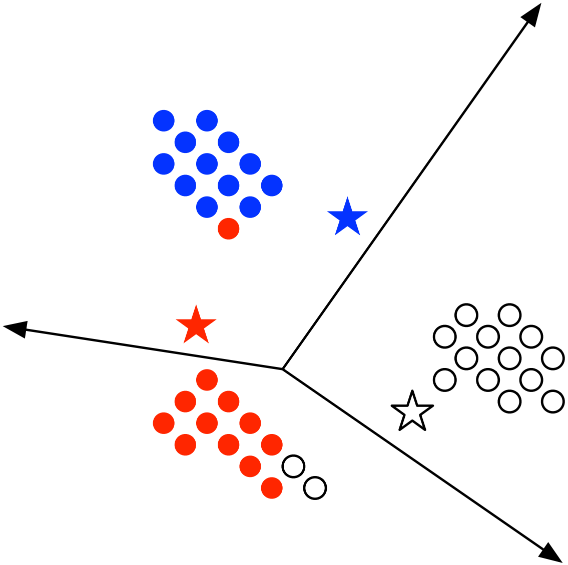
Assign each example to its nearest cluster center.

# An Iterative Clustering Algorithm



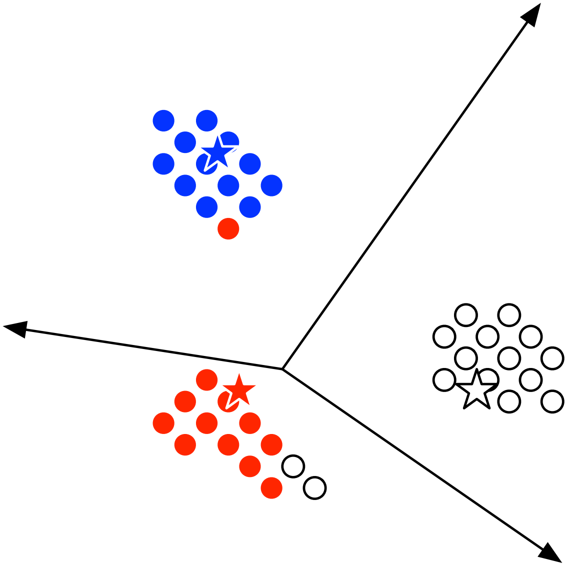
Recalculate cluster centers to reflect their respective examples.

# An Iterative Clustering Algorithm



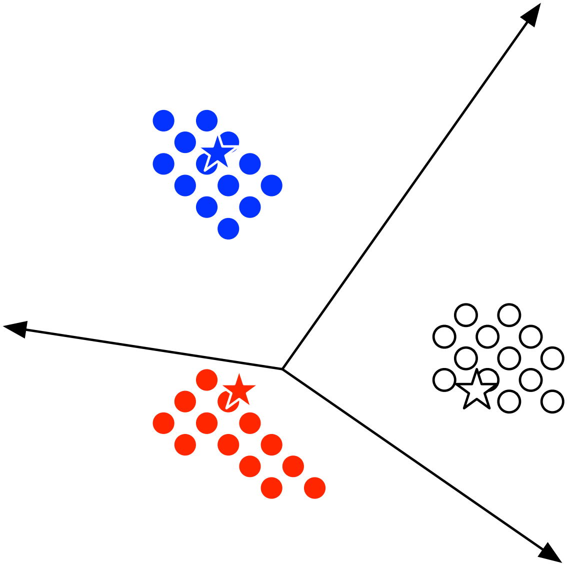
Assign each example to its nearest cluster center.

# An Iterative Clustering Algorithm



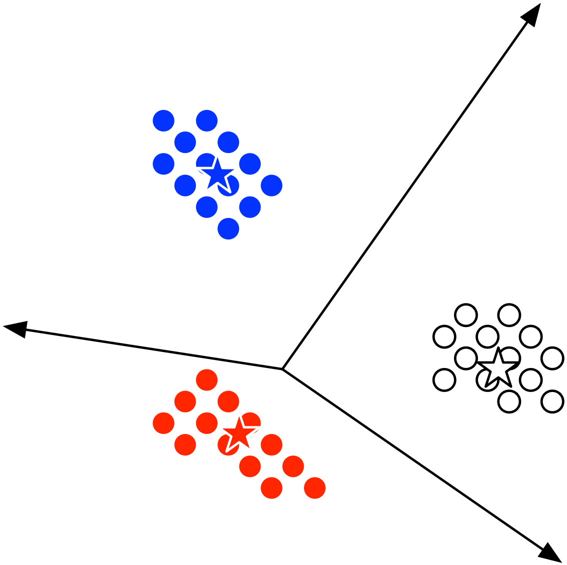
Recalculate cluster centers to reflect their respective examples.

# An Iterative Clustering Algorithm



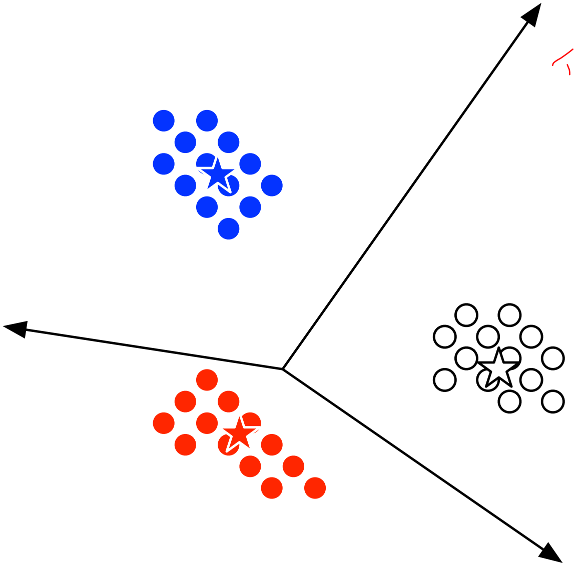
Assign each example to its nearest cluster center.

# An Iterative Clustering Algorithm



Recalculate cluster centers to reflect their respective examples.

# An Iterative Clustering Algorithm



there is a "cost"

At this point, nothing will change; we have converged.