Lecture 17 — k-means.

Alex Schwing and Matus Telgarsky

March 27, 2018

Announcements.

- Midterm is being graded right now.
  Grades posted tonight or tomorrow.
  Midterm handed back Thursday after lecture.
  Regrade requests have two weeks.
  (Sorry we are late with this!!!)
- **Homeworks pushed back** one week;
  no homework due this week;
  no TA office hours this week.

Schedule for today.

- Clustering basics; k-means objective.
- k-means algorithms.
- Applications.
- Ancillary topics.

Reading: Murphy book, chapter 11.

Clustering basics; unsupervised learning.

Clustering basics; unsupervised learning.

Currently we're doing **unsupervised learning**:

- Data comes **without supervision/labels**;
- Task is to **find structure in data**.

Clustering basics; unsupervised learning.

Currently we're doing **unsupervised learning**:

- Data comes **without supervision/labels**;
- Task is to **find structure in data**.

Schedule:

- Last lecture: **PCA**.

$$\underset{\substack{\text{subspaces } L \subseteq \mathbb{R}^d \\ \dim(L)=k}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \|x_i - \Pi_L x_i\|^2 .$$

- This lecture: k-**means clustering**.
- Future lectures: GMMs, HMMs, EM, GANs, ...

Clustering basics.

Clustering.

- *Partition* data $(x_i)_{i=1}^n$ into *clusters*.
- Similar data in same cluster;
  dissimilar data in different clusters.

Clustering basics.

Clustering.

- *Partition* data $(x_i)_{i=1}^n$ into *clusters*.
- Similar data in same cluster;
  dissimilar data in different clusters.
  (Good clustering depends on good similarity measure!)

Clustering basics.

Clustering.

- *Partition* data $(x_i)_{i=1}^n$ into *clusters*.
- Similar data in same cluster;
  dissimilar data in different clusters.
  (Good clustering depends on good similarity measure!)

Exemplar-based clustering.

- Associate each cluster with an *exemplar/center*.
- Many applications heavily use this center.
- Natural objective function:

$$\arg\min_{\mu_1,\ldots,\mu_k} \frac{1}{n}\sum_{i=1}^{n} \mathrm{sim}(x_i, \mu_j).$$

Remarks.

- $1/n$ often dropped;
  here it strengthens analogy to risks/losses.
- k-means uses $\mathrm{sim}(x_i, \mu_j) = \frac{1}{2}\|x_i - \mu_j\|_2^2$.

The k-means objective: basic properties.

Define the k-means objective as

$$\sum_{i=1}^{n} \min_j \|x_i - \mu_j\|_2^2 \,.$$

Remarks.

- $(\mu_1, \ldots, \mu_k)$ are the k means/exemplars/centers.
- Can treat $\min_j \|x - \mu_j\|_2^2$ as a (nonconvex!) loss.
- The k-means objective and the k-means *method* (presented shortly) are often conflated.
- This is an *exemplar*-based, *hard* clustering.
  There are many other types of clustering!

The k-means objective: gradients.

Let's take gradient wrt μ_1.

The k-means objective: gradients.

Let's take gradient wrt μ_1.
Define $\mu(x_i) \in (\mu_1, \ldots, \mu_k)$, closest center to x_i. (Ignore ties.)

$$\begin{aligned}
\nabla_{\mu_1} \sum_{i=1}^{n} \min_j \|x_i - \mu_j\|^2 &= \nabla_{\mu_1} \left(\sum_{\substack{i \in (1,\ldots,n) \\ \mu(x_i) = \mu_1}} \|x_i - \mu_1\|^2 + \sum_{\substack{i \in (1,\ldots,n) \\ \mu(x_i) \neq \mu_1}} \|x_i - \mu_1\|^2 \right) \\
&= \sum_{\substack{i \in (1,\ldots,n) \\ \mu(x_i) = \mu_1}} 2(x_i - \mu_1) + 0.
\end{aligned}$$

The k-means objective: gradients.

Let's take gradient wrt μ_1.
Define $\mu(x_i) \in (\mu_1, \ldots, \mu_k)$, closest center to x_i. (Ignore ties.)

$$\begin{aligned}
\nabla_{\mu_1} \sum_{i=1}^{n} \min_j \|x_i - \mu_j\|^2 &= \nabla_{\mu_1} \left(\sum_{\substack{i \in (1,\ldots,n) \\ \mu(x_i) = \mu_1}} \|x_i - \mu_1\|^2 + \sum_{\substack{i \in (1,\ldots,n) \\ \mu(x_i) \neq \mu_1}} \|x_i - \mu_1\|^2 \right) \\
&= \sum_{\substack{i \in (1,\ldots,n) \\ \mu(x_i) = \mu_1}} 2(x_i - \mu_1) + 0.
\end{aligned}$$

Remarks.

- Setting to 0, get $\mu_j' := \text{mean}\left(\{x_i : \mu(x_i) = \mu_j\}\right)$.
- Can define multiple algs from here (will come back to this).

The k-means objective: alternate form via assignments.

Let's make the assignment of data to centers explicit:

$$\min_{\mu_1,\ldots,\mu_k} \sum_{i=1}^{n} \min_j \|x_i - \mu_j\|_2^2 = \min_{\mu_1,\ldots,\mu_k} \min_{\substack{A\in\{0,1\}^{n\times k} \\ A\mathbf{1}=\mathbf{1}}} \sum_{i=1}^{n} \sum_{j=1}^{k} A_{ij}\|x_i - \mu_j\|_2^2.$$

Remarks.

- $A \in \{0,1\}^{n\times k}$ assigns data points to centers.
  It is a **hard assignment**: each x_i gets exactly one μ_j.
- Natural to consider **soft clustering** $A \in [0,1]^{n\times k}$, $A\mathbf{1} = \mathbf{1}$.
  We'll return to this next lecture.

The k-medians objective!

What if we drop the square:

$$\min_{\mu_1,\ldots,\mu_k} \min_j \sum_{i=1}^{n} \|x_i - \mu_j\|_2.$$

- Setting derivative to 0,

$$\sum_{x_i \in C_j} \frac{x_i - \mu_j}{\|x_i - \mu_j\|} = 0.$$

  where $C_j := \{x_i : \mu(x_i) = \mu_j\}$.
- **Univariate case:** recovers median.
  Multivariate case: "geometric" medians.
- If the square seems weird,
  for now just treat it as giving means not medians.

k-means algorithms.

k-means algorithms.

k-means objective

$$\sum_{i=1}^{n} \min_{j} \|x_i - \mu_j\|_2^2.$$

k-means algorithms.

k-means objective

$$\sum_{i=1}^{n} \min_j \|x_i - \mu_j\|_2^2.$$

As before: applying ∇_{μ_l} and setting to zero gives

$$\mu_l' := \frac{1}{|C_l|} \sum_{x_i \in C_l} x_i,$$

where $C_l := \{x_i : \mu(x_i) = \mu_l\}$ are points with μ_l as closest center.

k-means algorithms.

k-means objective

$$\sum_{i=1}^{n} \min_j \|x_i - \mu_j\|_2^2.$$

As before: applying ∇_{μ_l} and setting to zero gives

$$\mu_l' := \frac{1}{|C_l|} \sum_{x_i \in C_l} x_i,$$

where $C_l := \{x_i : \mu(x_i) = \mu_l\}$ are points with μ_l as closest center.
Let's turn this into an algorithm.

LLoyd's method. ("k-means algorithm".)

1. Choose initial clusters $(C_1, \ldots, C_k)$.
2. Repeat until convergence:
   2.1 **(Recenter.)** Set $\mu_j := \text{mean}(C_j)$ for $j \in (1, \ldots, k)$.
   2.2 **(Reassign).** Update $C_j := \{x_i : \mu(x_i) = \mu_j\}$ for $j \in (1, \ldots, k)$ (break ties arbitrarily).

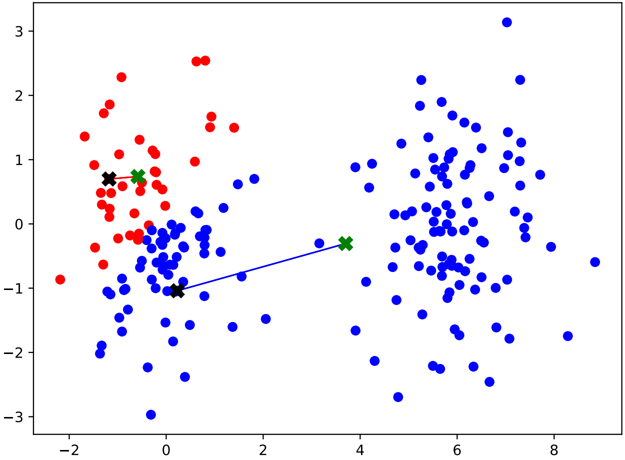
LLoyd's method. ("k-means algorithm".)

1. Choose initial clusters $(C_1, \ldots, C_k)$.
2. Repeat until convergence:
   2.1 **(Recenter.)** Set $\mu_j := \text{mean}(C_j)$ for $j \in (1, \ldots, k)$.
   2.2 **(Reassign).** Update $C_j := \{x_i : \mu(x_i) = \mu_j\}$ for $j \in (1, \ldots, k)$ (break ties arbitrarily).

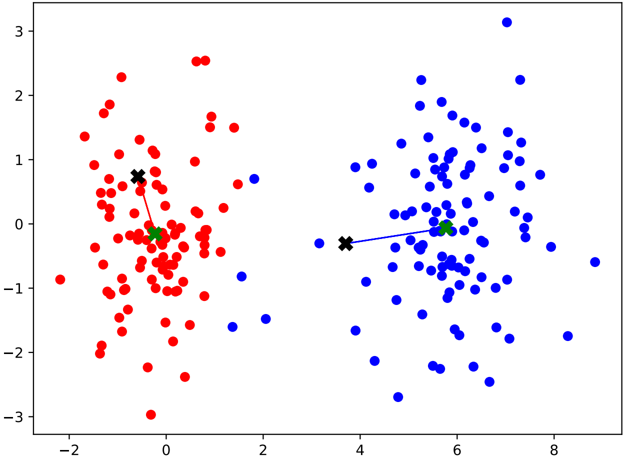
Remarks.

- $\mathcal{O}(nkd)$ per iteration.
- Initialization discussed shortly.
- This is **alternating minimization** on cluster assignments and cluster centers.
- Procedure terminates: each step can't increase cost, and there are finitely many partitions.

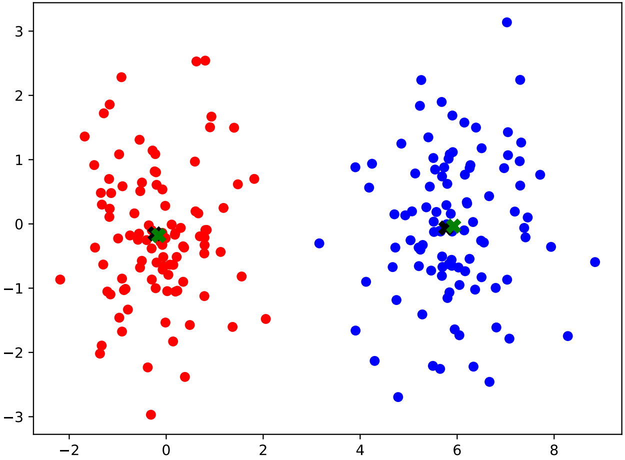
Static demo.



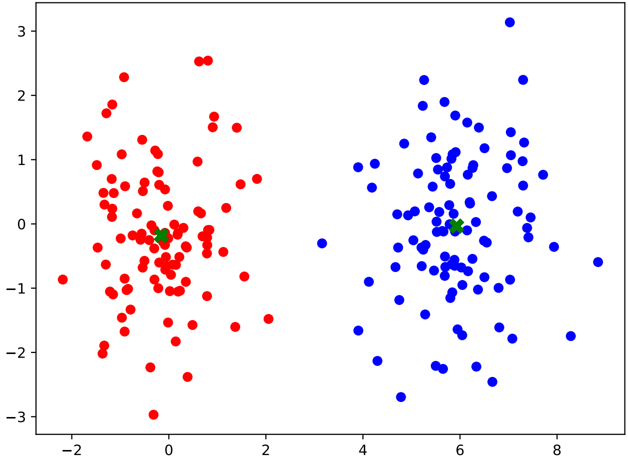
Static demo.



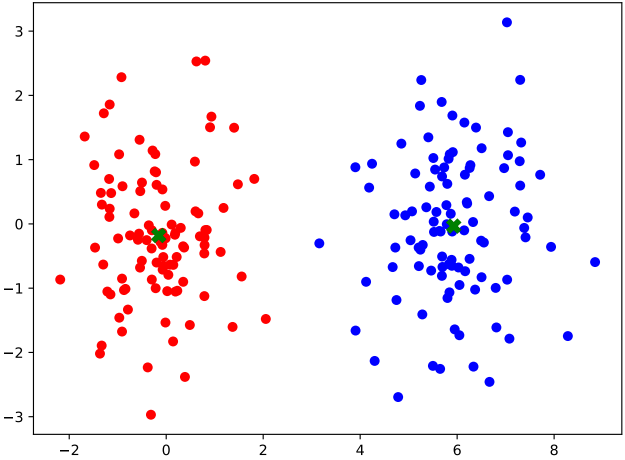
Static demo.



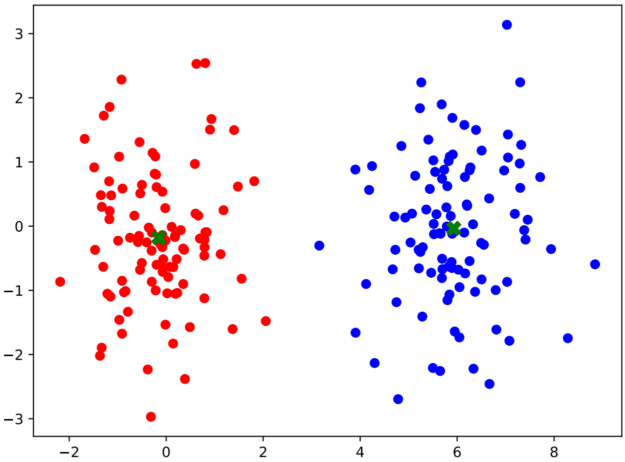
Static demo.



Static demo.



Static demo.



Lloyd's method revisited.

1. Choose initial clusters $(C_1, \ldots, C_k)$.
2. Repeat until convergence:
    - 2.1 **(Recenter.)** Set $\mu_j := \text{mean}(C_j)$ for $j \in (1, \ldots, k)$.
    - 2.2 **(Reassign).** Update $C_j := \{x_i : \mu(x_i) = \mu_j\}$ for $j \in (1, \ldots, k)$ (break ties arbitrarily).

Lloyd's method revisited.

1. Choose initial clusters $(C_1, \ldots, C_k)$.
2. Repeat until convergence:
   2.1 **(Recenter.)** Set $\mu_j := \text{mean}(C_j)$ for $j \in (1, \ldots, k)$.
   2.2 **(Reassign).** Update $C_j := \{x_i : \mu(x_i) = \mu_j\}$ for $j \in (1, \ldots, k)$ (break ties arbitrarily).

Let's understand this geometrically.

- Centers define a **Voronoi diagram/partition**: for each μ_j, define cell $V_j := \{x \in \mathbb{R}^d : \mu(x) = \mu_j\}$ (break ties arbitrarily).
- Reassignment leaves assignment consistent with Voronoi cells.
- Recentering might shift data outside Voronoi cells!

Interactive demo.

Go to `http://mjt.cs.illinois.edu/htv/`.

(Shown in class.)

(This should make the Voronoi cells clear!)

Does the algorithm optimize well?

k-means objective is NP-hard when $d \geq 2$.

- In practice, Lloyd's method seems to optimize well;
  In theory, output can have **unboundedly poor cost**.
  (Example given in class: 4 corners of a rectangle.)
- In practice, method takes few iterations;
  in theory: can take $2^{\Omega(\sqrt{n})}$ iterations!
  (Examples of this are painful.)

Initialization matters!

- **Easy choices:**
  - k random points from dataset.
  - Random partition.
- **Standard choice** (theory and practice)**:**
  "D^2-sampling"/`kmeans++`
  1. Choose μ_1 uniformly at random from data.
  2. for $j \in (2, \ldots, k)$:
     2.1 Choose $x_i \propto \min_{l<j} \|x_i - \mu_l\|_2^2$.
- `kmeans++` is *randomized furthest-first traversal*; regular furthest-first fails with outliers.
- Scikits-learn and Matlab both default to `kmeans++`.

Applications

Applications

- The **clusters** found by k-means are useful to *data analysis*: finding groupings that were hard to see.
- The **exemplars/centers** are also extremely useful!

Application: vector quantization.

Vector quantization with k-means.

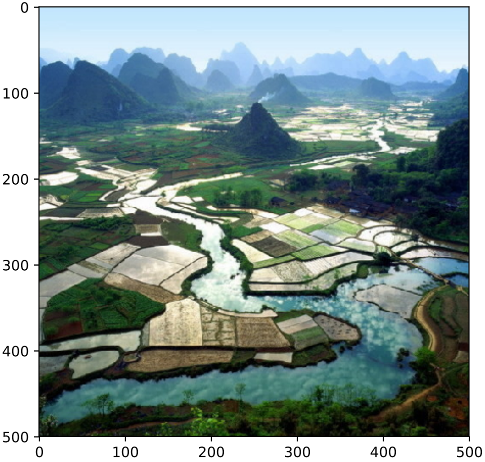
- Let $(x_i)_{i=1}^n$ be given.
- run k-means to obtain $(\mu_1, \dots, \mu_k)$.
- Replace each $(x_i)_{i=1}^n$ with $(\mu(x_i))_{i=1}^n$.

Encoding size reduces from $\mathcal{O}(nd)$ to $\mathcal{O}(kd + n \ln(k))$.
Examples.

- Audio compression.
- Image compression.

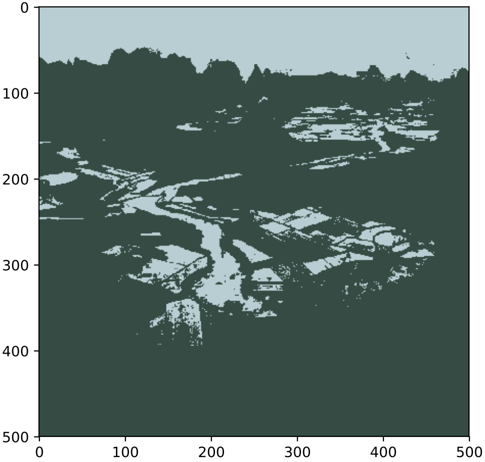
Application: pixel-level quantization.

- Run k-means on **pixels**.
- Obtain k exemplars, replace pixels with closest exemplar.



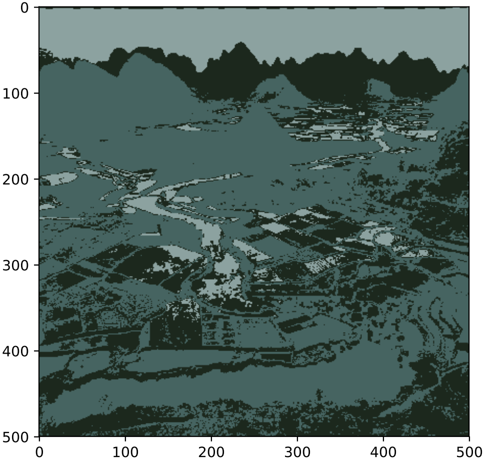
Application: pixel-level quantization.

- Run k-means on **pixels**.
- Obtain k exemplars, replace pixels with closest exemplar.



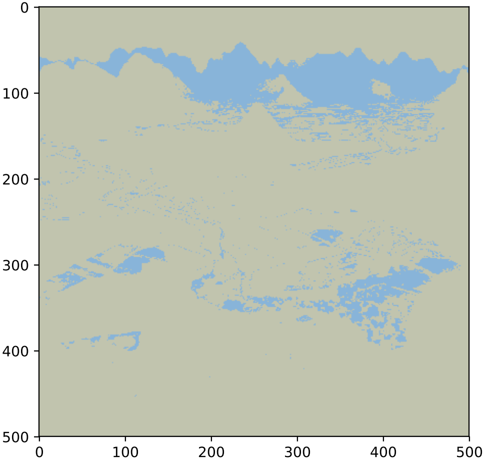
Application: pixel-level quantization.

- Run k-means on **pixels**.
- Obtain k exemplars, replace pixels with closest exemplar.

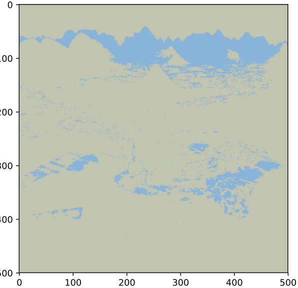


Application: pixel-level quantization.

- Run k-means on **pixels**.
- Obtain k exemplars, replace pixels with closest exemplar.



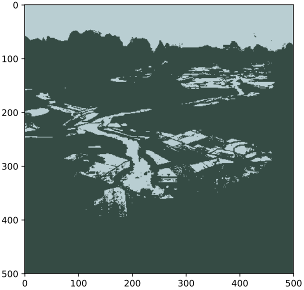
Application: pixel-level quantization.



Looks kindof bad!

Quick fix: use a different *color space*.

Application: pixel-level quantization.



Separating objects in an image is called **segmentation**.

With some tweaks (color space, different clustering method), can cheaply get a reasonable segmentation.

Application: patch-level quantization.

1. Now $(x_i)_{i=1}^n$ denotes *patches* of *many* images.
2. Obtain exemplars $(\mu_1, \ldots, \mu_k)$ via k-means.
3. Replace image patches with closest exemplar.

Application: feature learning.

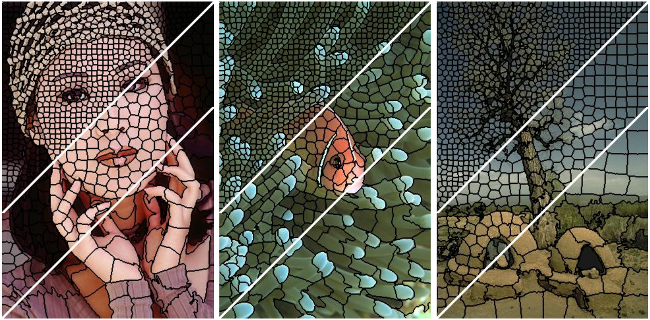
1. Start with $(x_i)_{i=1}^n$, where $x_i \in \mathbb{R}^d$.
2. Run k-means, obtain $(\mu_1, \ldots, \mu_k)$, where $\mu_j \in \mathbb{R}^d$.
3. Replace x_i with $\tilde{x}_i \in \mathbb{R}^k$ where $(\tilde{x}_i)_j := \exp(-\|x_i - \mu_j\|^2)$ (or some other similarity measure).
4. Run whatever ML algorithm (e.g., least squares) on $(\tilde{x}_i)_{i=1}^n$. (Example in class: the "xor" example we keep mentioning...)

Application: quantizing into "superpixels".

Earlier quantizations ignore spatial structure.
Cheap fix: add image coordinates to RGB data at each pixel, and tune the distance metric!

Application: quantizing into "superpixels".

Earlier quantizations ignore spatial structure.
Cheap fix: add image coordinates to RGB data at each pixel, and tune the distance metric!



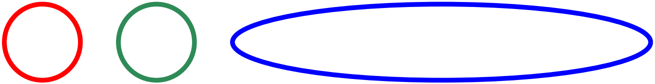
Other applications.

Especially after tuning feature encoding and distance metric,
can apply k-means much more broadly.
(E.g., to text.)

Ancillary topics.

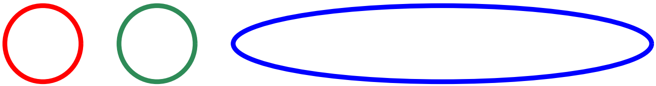
Ancillary topics: spherical clusters.

What will k-means do with $k = 3$?



Ancillary topics: spherical clusters.

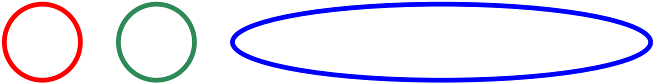
What will k-means do with $k = 3$?



k-means prefers **spherical clusters**.

Ancillary topics: spherical clusters.

What will k-means do with $k = 3$?

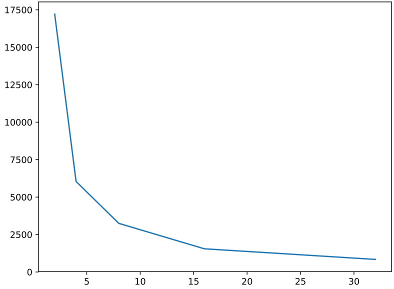


k-means prefers **spherical clusters**.

Changing similarity measure means all clusters still same shape.
Next lecture gives another option.

How to choose k?

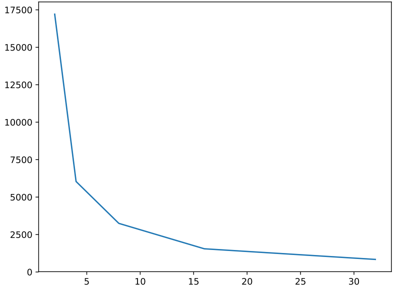
Costs when quantizing pixels of *Guilin*.



Reasonable to choose k at "elbow";
trades off accuracy and complexity.

How to choose k?

Costs when quantizing pixels of *Guilin*.



Reasonable to choose k at “elbow”;
trades off accuracy and complexity.

There are other, more complicated ways. (E.g., “bayes information criterion”, “Akaike information criterion”, ...)

Choice of k; sometimes no good choice.

Which of $k \in \{2, 3\}$ better on following data?

Choice of k; sometimes no good choice.

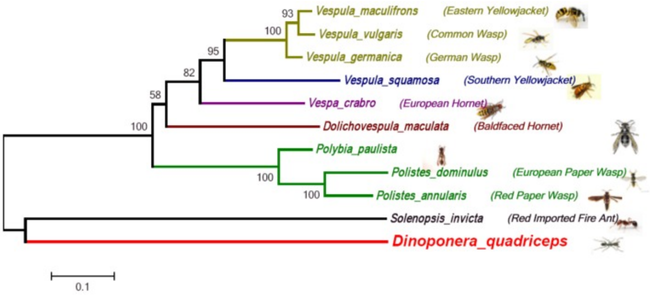
Which of $k \in \{2, 3\}$ better on following data?

Perhaps **neither**; want 6. (“Elbow method” works here.)

Choice of k; hierarchical clustering.

Sometimes *multiple* choices of k make sense.

E.g., Phylogenetic trees have multiple notions of scales.



(Image credit: `https://www.researchgate.net/figure/Phylogenetic-tree-based-on-neighbor-joining-analyses-of-a-fig8_260108945`.)

The k-means objective: alternate form without exemplars!

Let C_j be the points assigned to μ_j.

$$\sum_{i=1}^{n} \min_j \|x_i - \mu_j\|_2^2 = \sum_{j=1}^{k} \frac{1}{2|C_j|} \sum_{a,b \in C_j} \|a - b\|_2^2.$$

This gives a **non-exemplar** way to reason about k-means.

k-means: key points.

k-means: key points.

- k-means is a (hard) clustering method.
- The objective function is $\min_{\mu_1,\ldots,\mu_k} \sum_{i=1}^{n} \min_j \|x_i - \mu_j\|_2^2$.
- Remember the standard heuristic ("Lloyd's method"), it is **alternating minimization** between **assignments** and **centers**.
- k-means finds means/exemplars/centers; these are useful in many applications, e.g., **vector quantization**.

Next time: **G**aussian **M**ixture **M**odels!