discussion 10

Semi-Supervised Learning

K-means++

overview

- semi-supervised learning
  - Generative Models
  - semi-SVM
  - graph-based method --- convex!

  - co-training
- Unsupervised learning
  - clustering --- K-means; K-means++
  - PCA (not cover in this discussion)

semi-supervised learning

- inductive semi-supervised learning

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^{l} \overset{iid}{\sim} p(\mathbf{x}, y) \qquad \{\mathbf{x}_i\}_{i=l+1}^{l+u} \overset{iid}{\sim} p(\mathbf{x}),$$

$$\text{learns a predictor } f : \mathcal{X} \mapsto \mathcal{Y}$$

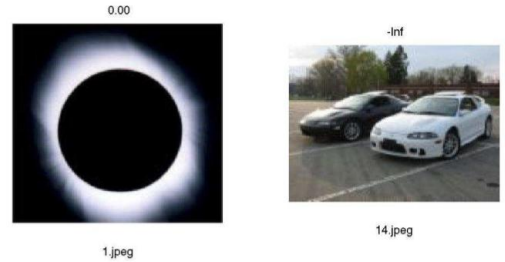
- transductive learning.
  - only interested in the predictions on the unlabeled training data

How semi-supervised learning help ?

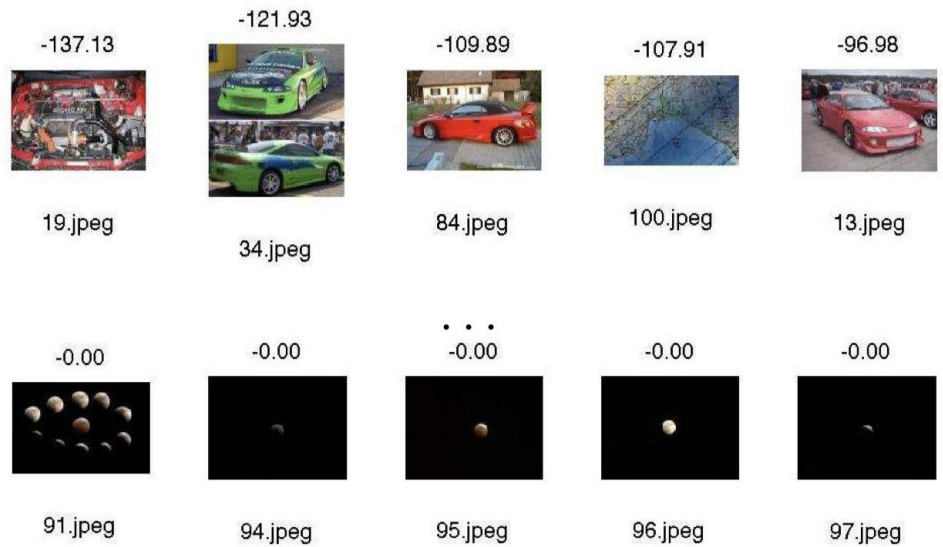
- What implicit ordering is induced by the unlabeled data?
- How to find a predictor near the top of this implicit ordering and fits the labeled data well ?

Self-training example: image categorization

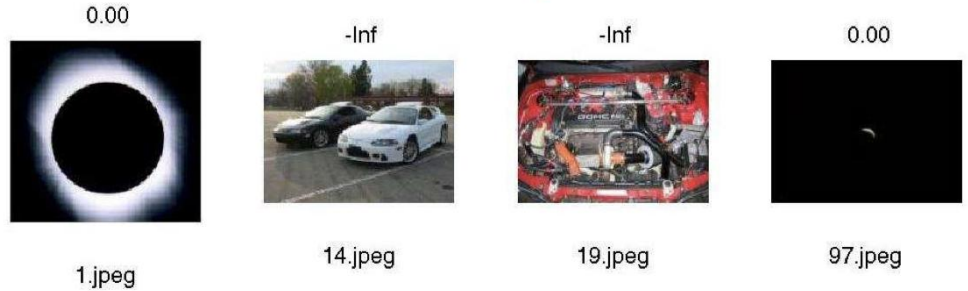
1. Train a naïve Bayes classifier on the two initial labeled images



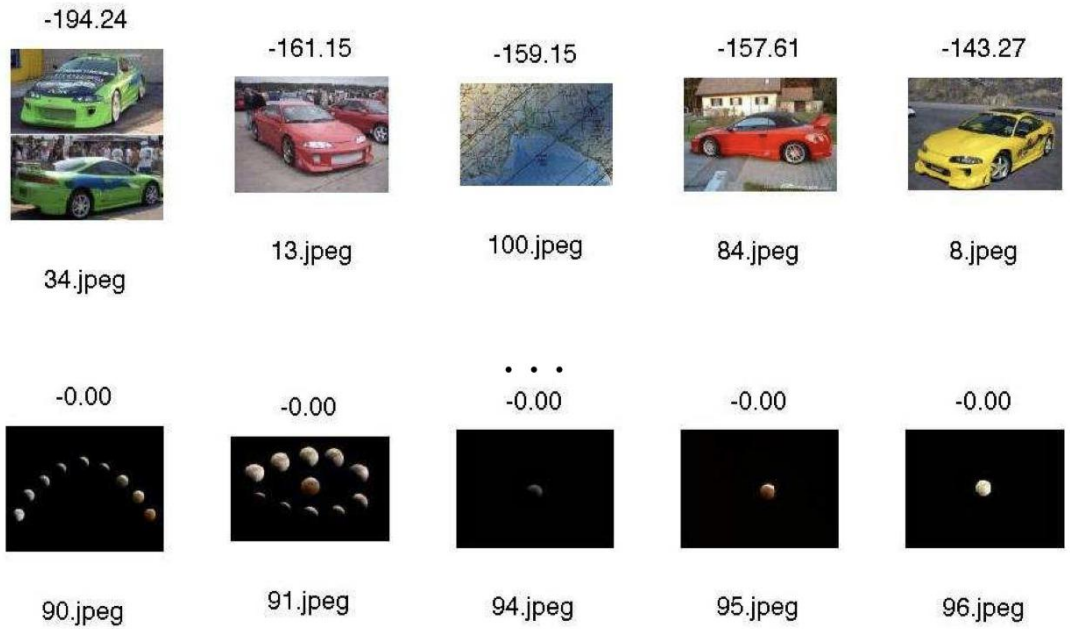
2. Classify unlabeled data, sort by confidence $\log p(y = \text{astronomy}|x)$



3. Add the most confident images and predicted labels to labeled data



4. Re-train the classifier and repeat



Generative Model

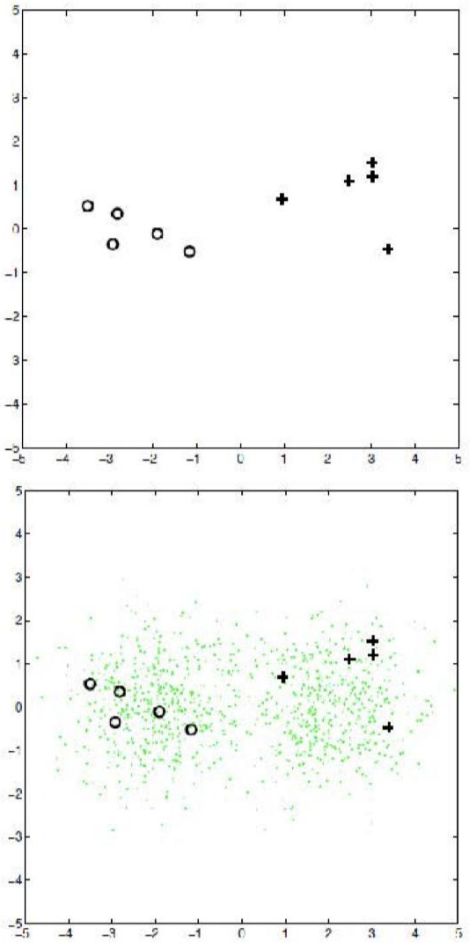
Gaussian mixture model (GMM)

- Model parameters:
  $\theta = \{\pi_i, \mu_i, \Sigma_i\}_{i=1}^{K}$, π_i: class priors, μ_i:Gaussian means, Σ_i:covariance matrices
- Joint distribution

$$\begin{aligned} p(\mathbf{x}, \mathbf{y}|\theta) &= p(\mathbf{y}|\theta)p(\mathbf{x}|\mathbf{y}, \theta) \\ &= \sum_{i=1}^{K} \pi_i \mathcal{N}(\mathbf{x}; \mu_i, \Sigma_i) \end{aligned}$$

- Classification:

$$p(\mathbf{y}|\mathbf{x}, \theta) = \frac{p(x, y|\theta)}{\sum_{i=1}^{K} p(\mathbf{x}, y_i|\theta)}$$

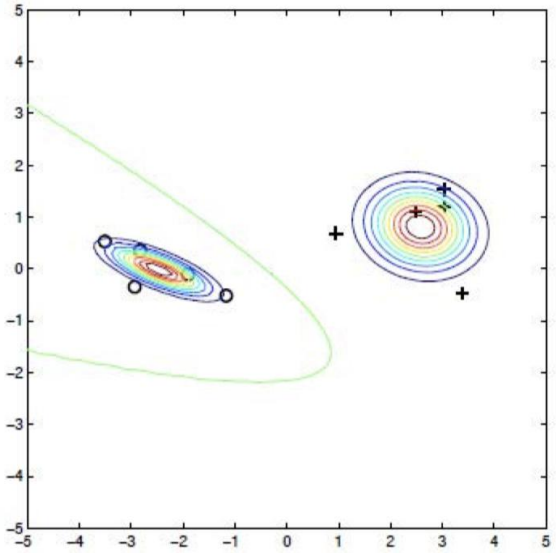


Generative Model

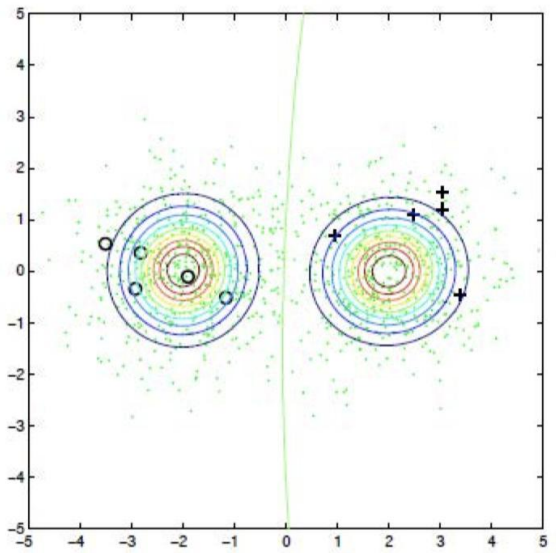
Effect of unlabeled data in GMM

The difference comes from maximizing different quantities

$p(X_l, Y_l|\theta)$



$p(X_l, Y_l, X_u|\theta)$



Generative Model

implicit ordering: large to small ordering of log likelihood of θ

$$\log p(\{\mathbf{x}_i\}_{i=l+1}^{l+u} \mid \theta) = \sum_{i=l+1}^{l+u} \log \left(\sum_{y \in \mathcal{Y}} p(\mathbf{x}_i, y \mid \theta) \right)$$

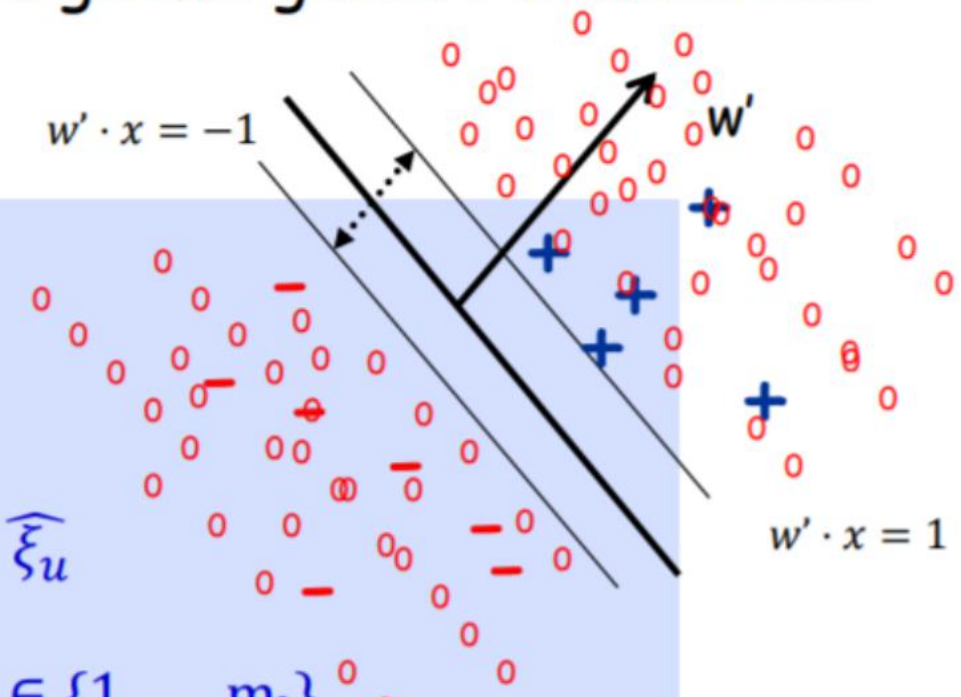
Generative Model

- objective function
  - with only labeled data
    - $\log p(X_l, Y_l|\theta) = \sum_{i=1}^{l} \log p(y_i|\theta)p(x_i|y_i, \theta)$
    - MLE for θ trivial (sample mean and covariance)
  - with both labeled and unlabeled data

$$\log p(X_l, Y_l, X_u|\theta) = \sum_{i=1}^{l} \log p(y_i|\theta)p(x_i|y_i, \theta) + \sum_{i=l+1}^{l+u} \log \left(\sum_{y=1}^{2} p(y|\theta)p(x_i|y, \theta) \right)$$

semi - SVM

Optimize for the separator with large margin wrt labeled and unlabeled data. [Joachims '99]



Input: $S_l=\{(x_1, y_1), \ldots,(x_{m_l}, y_{m_l})\}$

$S_u=\{x_1, \ldots,x_{m_u}\}$

$\text{argmin}_w \, ||w||^2 + C \sum_i \xi_i + C \sum_u \widehat{\xi_u}$

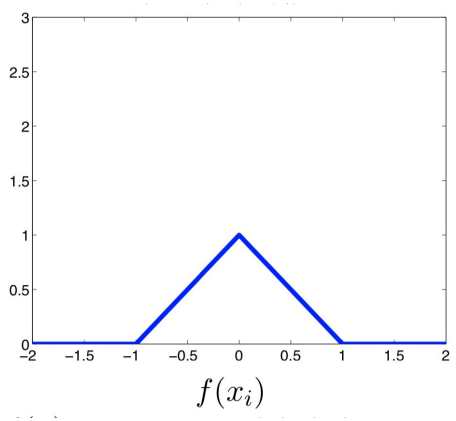
- $y_i \, w \cdot x_i \geq 1-\xi_i$, for all $i \in \{1, \ldots, m_l\}$
- $\widehat{y_u} w \cdot x_u \geq 1 - \widehat{\xi_u}$, for all $u \in \{1, \ldots, m_u\}$
- $\widehat{y_u} \in \{-1, 1\}$ for all $u \in \{1, \ldots, m_u\}$

semi-SVM

objective function

$$\underset{f}{\operatorname{argmin}} \frac{1}{l}\sum_{i=1}^{l}\max(1-y_i f(\mathbf{x}_i),0)+\lambda_1\|f\|^2+\lambda_2\frac{1}{u}\sum_{i=l+1}^{l+u}\max(1-|f(\mathbf{x})|,0)$$

hat loss: （non-convex） $\max(1-|f(\mathbf{x})|,0)$



Heuristic idea:

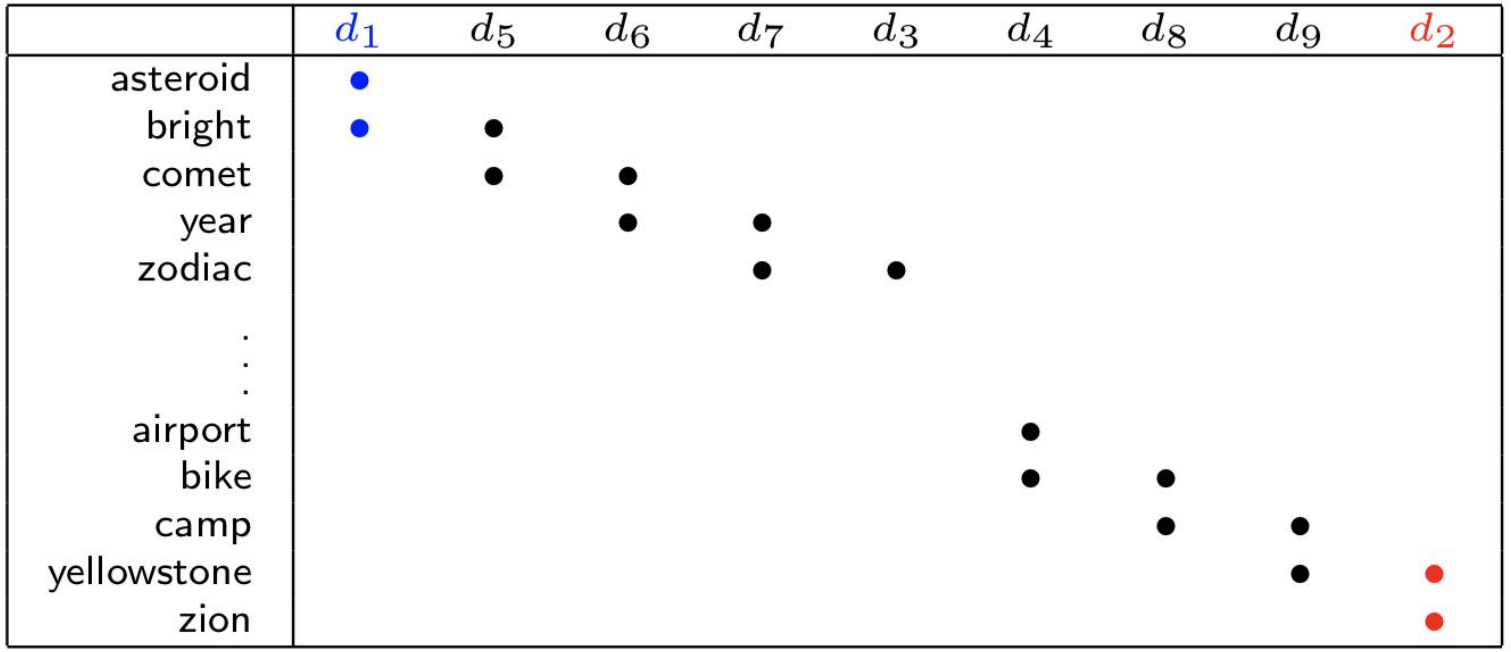
- Maximize margin over the labeled points
- Get initial labels to unlabeled points based on this separator.
- Try flipping labels of unlabeled points to see if doing so can increase margin.

- {-1, +1} => {+1, -1} any other class combinations ?

- local optimal

Graph-Based Models

Labels "propagate" via similar unlabeled articles.

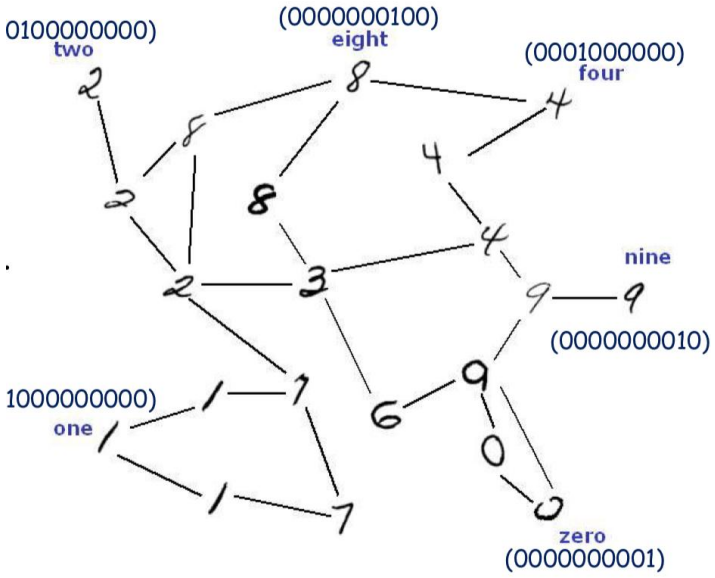


Graph-Based Models

How to create the graph:

$$G =< V, E >= \begin{cases} V : datapoints\ (S_l \cup S_u) \\ E : Similarity/Weights \end{cases}$$

1. Adjacency graph:
   - K-NN
   - $\sum$-NN, where $\sum$ is the distance of two data points
2. Graph weights
   - Simple formulation: {0,1}
   - Gaussian kernel function



Graph-Based Models

Large w_{ij} implies a preference for the predictions $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ to be the same. This can be formalized by the graph energy of a function f:

$$\sum_{i,j=1}^{l+u} w_{ij}(f(\mathbf{x}_i) - f(\mathbf{x}_j))^2.$$

The graph energy induces an implicit ordering of $f \in \mathcal{F}$ from small to large.

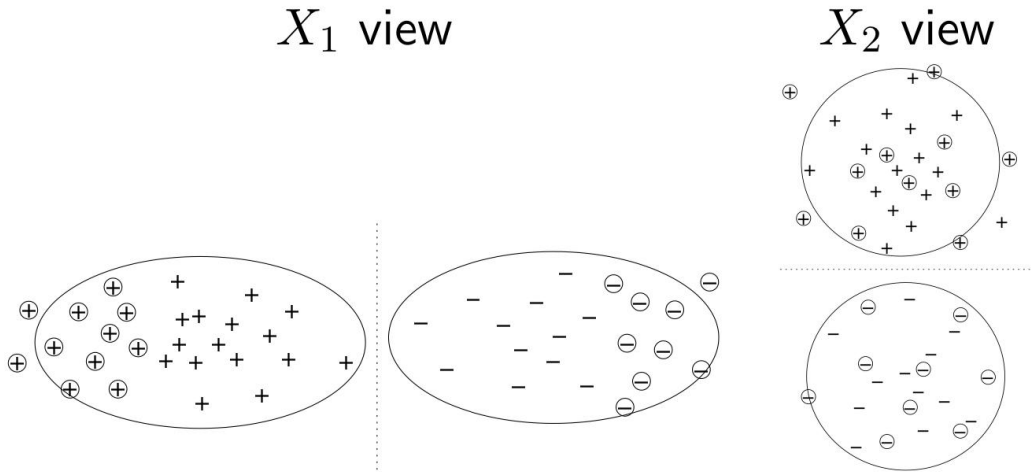
graph Laplacian matrix.

Graph-Based Models

- objective function

$$\underset{f}{\operatorname{argmin}} \frac{1}{l} \sum_{i=1}^{l} c(f(\mathbf{x}_i), y_i) + \lambda_1 \|f\|^2 + \lambda_2 \sum_{i,j=1}^{l+u} w_{ij} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2,$$

co-training



Co-training algorithm

1. Train two classifiers: $f^{(1)}$ from $(X_l^{(1)}, Y_l)$, $f^{(2)}$ from $(X_l^{(2)}, Y_l)$.
2. Classify X_u with $f^{(1)}$ and $f^{(2)}$ separately.
3. Add $f^{(1)}$'s k-most-confident $(x, f^{(1)}(x))$ to $f^{(2)}$'s labeled data.
4. Add $f^{(2)}$'s k-most-confident $(x, f^{(2)}(x))$ to $f^{(1)}$'s labeled data.
5. Repeat.

co-training

- objective function

individual loss function $c^{(u)}$ and regularizer $\Omega^{(u)}$

$$\underset{\langle f^{(1)},\ldots f^{(m)}\rangle}{\operatorname{argmin}} \sum_{u=1}^{m}\left(\frac{1}{l}\sum_{i=1}^{l} c^{(u)}(f^{(u)}(\mathbf{x}_i), y_i) + \lambda_1 \Omega^{(u)}(f^{(u)})\right)$$

$$+\lambda_2 \sum_{i=l+1}^{l+u}\sum_{u,v=1}^{m} c(f^{(u)}(\mathbf{x}_i), f^{(v)}(\mathbf{x}_i)).$$

disaggreement on the unlabeled data

unsupervised learning

- Kmeans ; kmeans++
- PCA

K-Means Algorithm: **Details**

```
centers ← pick k initial Centers

while (centers are changing) {
   // Compute the assignments (E-Step)
   asg ← [(x, nearest(centers, x)) for x in data]

   // Compute the new centers (M-Step)
   for i in range(k):
      centers[i] =
         mean([x for (x, c) in asg if c == i])
}
```

Guaranteed to converge!

… to what?

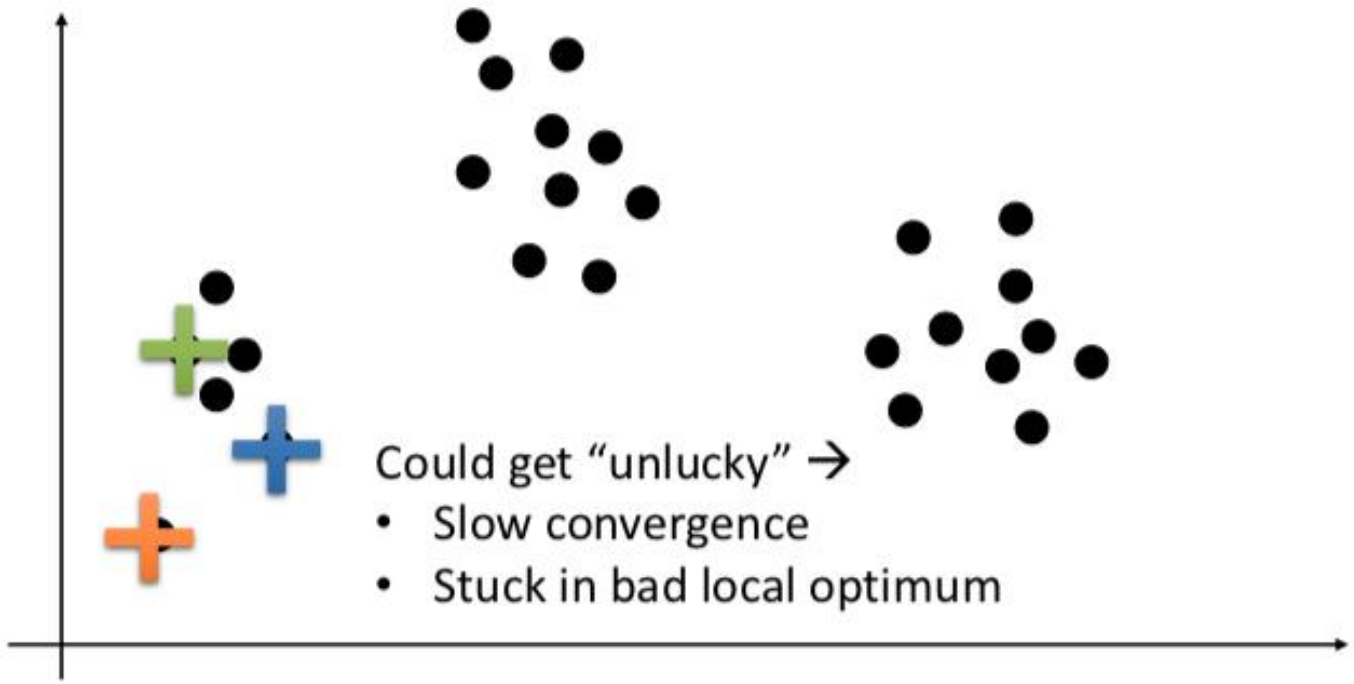
To a local optimum. ☹

Depends on Initial Centers

Kmeans++

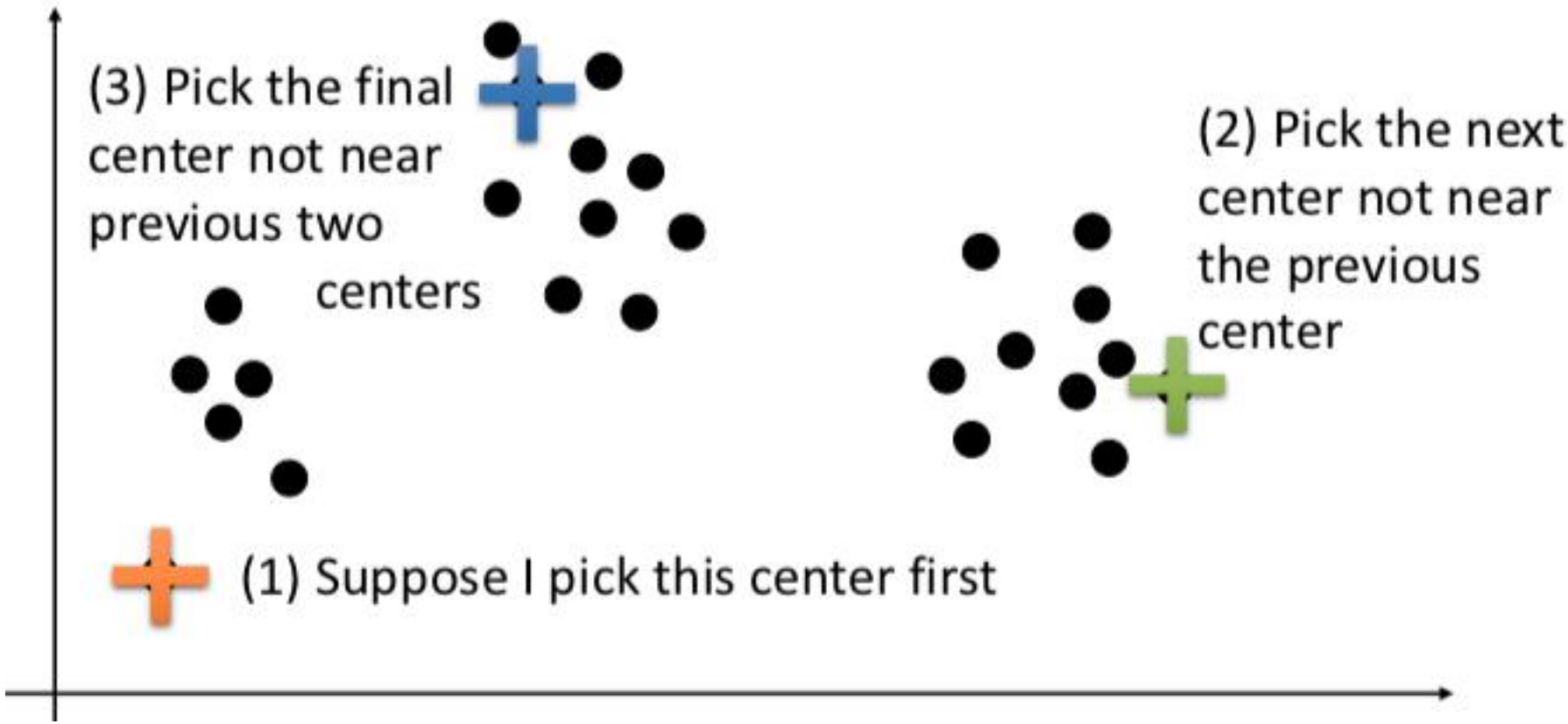
Picking the Initial Centers

- **Simple Strategy:** select *k* points at random
  - What could go wrong?



➢ **Better Strategy:** kmeans++

- Randomized approx. algorithm
- Intuition select points that are not near existing centers



K-Means++ Algorithm

```
centers ← set(randomly select a single point)


while len(centers) < k:
  # Compute the distance of each point
  # to its nearest center dSq = d^2
  dSq ← [(x, dist_to_nearest(centers, x)^2) for x in data]
  # Sample a new point with probability
  # proportional to dSq
  c ← sample_one(data, prob = dSq / sum(dSq))
  # Update the clusters
  centers.add(c)
```

Res-A: weighted reservoir sampling

- **Goal:** *Sample **k** records from a stream where record **i** is included in the sample with probability proportional to w_i*
- **Algorithm:**
  - For each record ***i*** draw a uniform random number:

$$u_i \sim \mathbf{Unif}(0, 1)$$

  - Select the top-k records ordered by: u_i^{1/w_i}

- **Common ML Pattern?**
  - **Query Function:** `[pow(rand(), 1 / record.w), record]`
  - **Agg. Function:** *top-k heap*

Basic Analysis Behind **Res-A**

- Define the random variable: $X_i = u_i^{1/w_i}$
- Then:

$$\mathbf{P}(X_i < \alpha) = \mathbf{P}\left(u_i^{1/w_i} < \alpha\right) = \mathbf{P}(u_i < \alpha^{w_i}) = \alpha^{w_i}$$

$$\mathbf{p}(X_i = \alpha) = w_i \alpha^{w_i - 1}$$

Derivative of CDF → PDF

- Suppose we want to pick just one element (k=1)
  - Probability of selecting X_i is:

$$\int_0^1 \mathbf{p}(X_i = \alpha) \prod_{j \neq i} \mathbf{P}(X_j < \alpha)\, d\alpha = \int_0^1 (w_i \alpha^{w_i - 1}) \prod_{j \neq i} \alpha^{w_j}\, d\alpha$$

$$= \frac{w_i}{\sum_j w_j}$$

We won't test you on this derivation

$$\int_0^1 \mathbf{p}\left(X_i = \alpha\right) \prod_{j \neq i} \mathbf{P}\left(X_j < \alpha\right) d\alpha = \int_0^1 \left(w_i \alpha^{w_i - 1}\right) \prod_{j \neq i} \alpha^{w_j} d\alpha$$

$$= w_i \int_0^1 \left(\alpha^{w_i - 1}\right) \alpha^{\sum_{j \neq i} w_j} d\alpha$$

$$= w_i \int_0^1 \alpha^{-1 + \sum_j w_j} d\alpha$$

$$= \frac{w_i}{\sum_j w_j} \alpha^{\sum_j w_j} \Bigg|_{\alpha=0}^{\alpha=1}$$

$$= \frac{w_i}{\sum_j w_j}$$