Probability and Statistics in Vision, Gaussian Mixture Models and EM

Probability

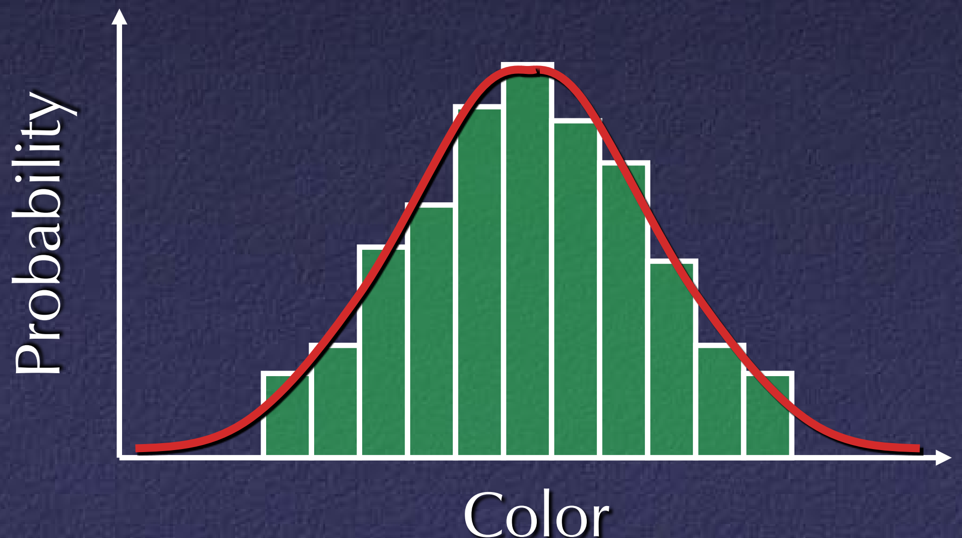
- Objects not all the same
  - Many possible shapes for people, cars, …
  - Skin has different colors
- Measurements not all the same
  - Noise
- But some are more probable than others
  - Green skin not likely

Probability and Statistics

- Approach: probability distribution of expected objects, expected observations
- Perform mid- to high-level vision tasks by finding most likely model consistent with actual observations
- Often don't know probability distributions – learn them from statistics of training data

Concrete Example – Skin Color

- Suppose you want to find pixels with the color of skin
- Step 1: learn likely distribution of skin colors from (possibly hand-labeled) training data



Conditional Probability

- This is the probability of observing a given color given that the pixel is skin
- Conditional probability p(color|skin)

Skin Color Identification

- Step 2: given a new image, want to find whether each pixel corresponds to skin
- Maximum likelihood estimation: pixel is skin iff $p(\text{skin} \mid \text{color}) > p(\text{not skin} \mid \text{color})$
- But this requires knowing $p(\text{skin} \mid \text{color})$ and we only have $p(\text{color} \mid \text{skin})$

Bayes's Rule

- "Inverting" a conditional probability:
  $$p(B|A) = p(A|B) \cdot p(B) \,/\, p(A)$$
- Therefore,
  $p(\text{skin}|\text{color}) = p(\text{color}|\text{skin}) \cdot p(\text{skin}) \,/\, p(\text{color})$
- p(skin) is the prior – knowledge of the domain
- p(skin|color) is the posterior – what we want
- p(color) is a normalization term

Priors

- p(skin) = prior
  - Estimate from training data
  - Tunes “sensitivity” of skin detector
  - Can incorporate even more information:
    e.g. are skin pixels more likely to be found in certain regions of the image?
- With more than 1 class, priors encode what classes are more likely

Skin Detection Results

Jones & Rehg

Skin Color-Based Face Tracking

Learning Probability Distributions

- Where do probability distributions come from?
- Learn them from observed data

Gaussian Model

- Simplest model for probability distribution: Gaussian

Symmetric:

$$p(\vec{x}) = e^{-\frac{(\vec{x}-\vec{\mu})^2}{2\sigma^2}}$$

Asymmetric:

$$p(\vec{x}) = e^{-\frac{(\vec{x}-\vec{\mu})^{\mathrm{T}}\Sigma^{-1}(\vec{x}-\vec{\mu})}{2}}$$

Maximum Likelihood

- Given observations $x_1 \ldots x_n$, want to find model m that maximizes likelihood

$$p(x_1 \ldots x_n \mid m) = \prod_{i=1}^{n} p(x_i \mid m)$$

- Can rewrite as

$$-\log L(m) = \sum_{i=1}^{n} -\log p(x_i \mid m)$$

Maximum Likelihood

- If *m* is a Gaussian, this turns into

$$-\log L(m) = \sum_{i=1}^{n} (x_i - \mu)^{\mathrm{T}} \Sigma^{-1} (x_i - \mu)$$

and minimizing it (hence maximizing likelihood) can be done in closed form

$$\mu = \tfrac{1}{n} \sum_{i=1}^{n} x_i$$

$$\Sigma = \tfrac{1}{n} \sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^{\mathrm{T}}$$

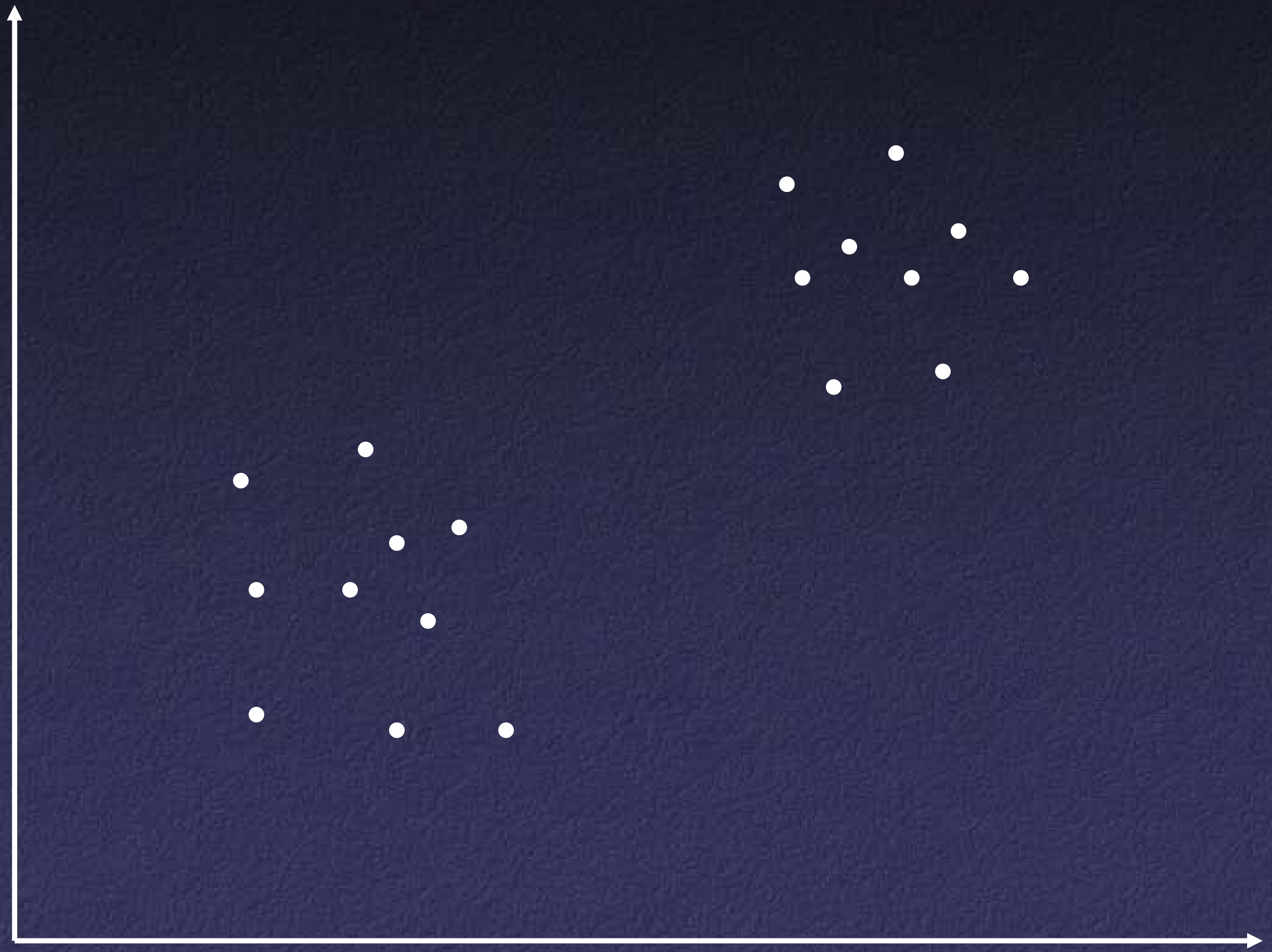
Mixture Models

- Although single-class models are useful, the real fun is in multiple-class models
- p(observation) = $\Sigma\ \pi_{\text{class}}\ p_{\text{class}}$(observation)
- Interpretation: the object has some probability π_{class} of belonging to each class
- Probability of a measurement is a linear combination of models for different classes

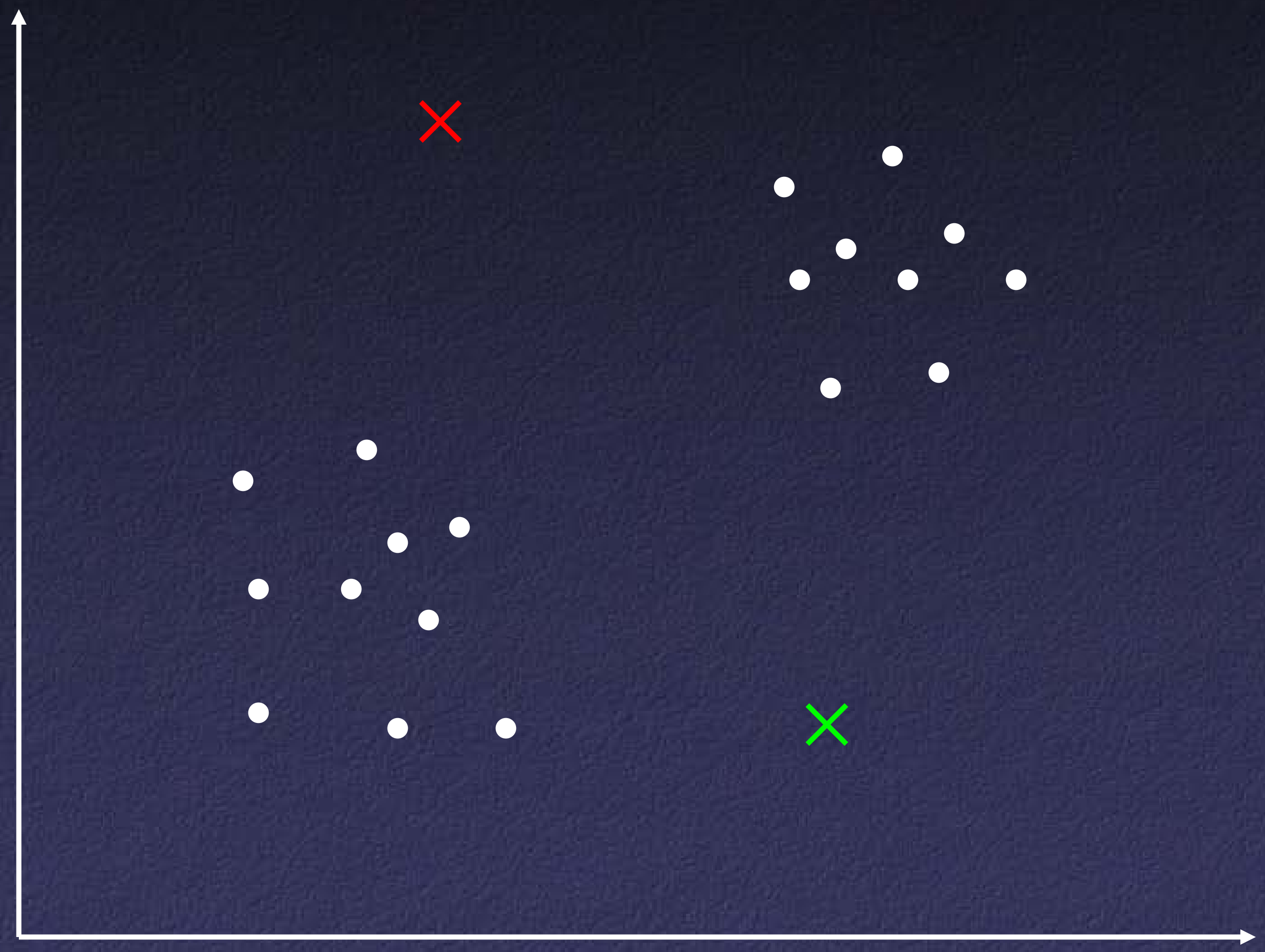
Learning Mixture Models

- No closed form solution
- *k*-means: Iterative approach
  - Start with *k* models in mixture
  - Assign each observation to closest model
  - Recompute maximum likelihood parameters for each model

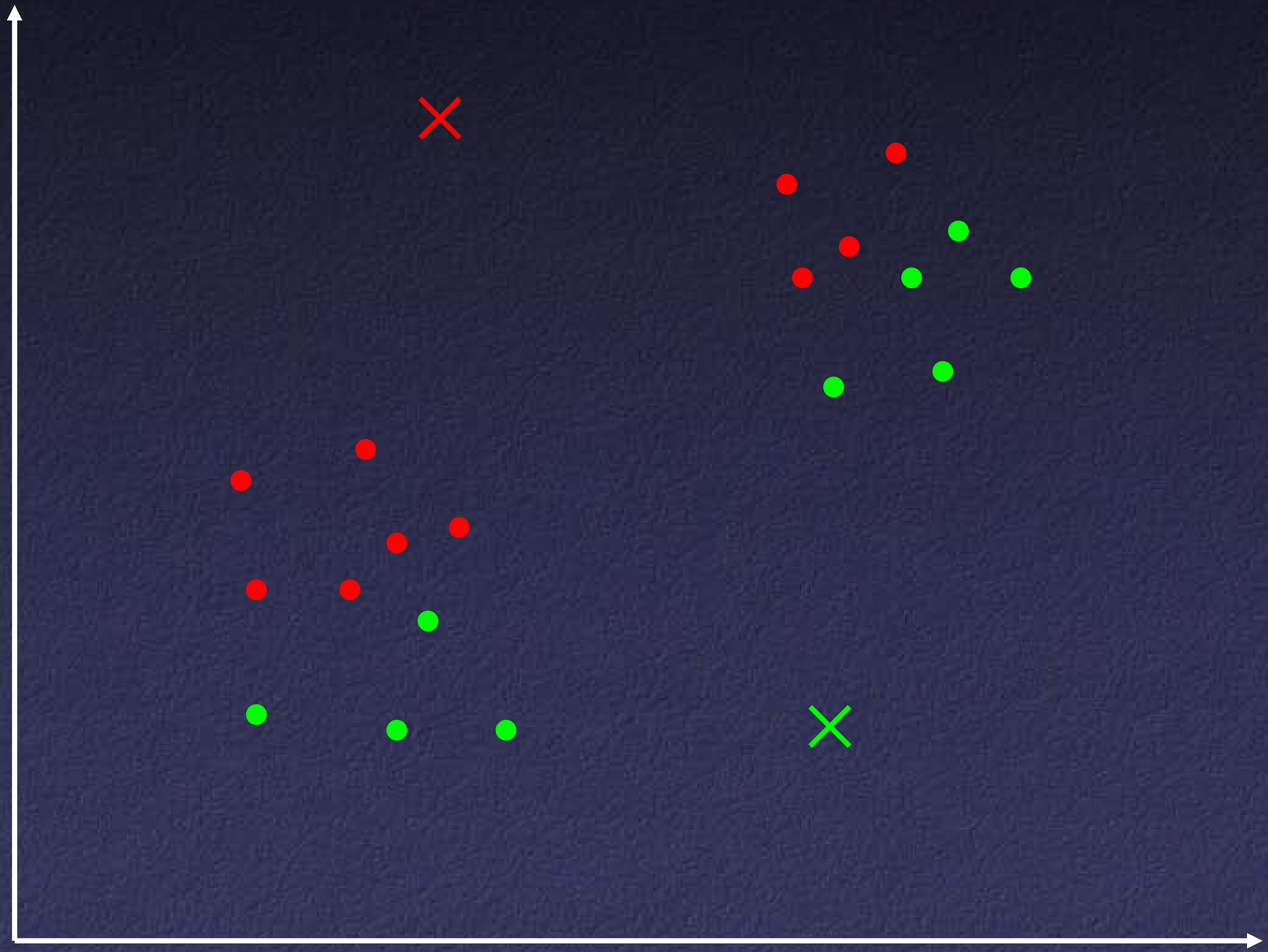
k-means



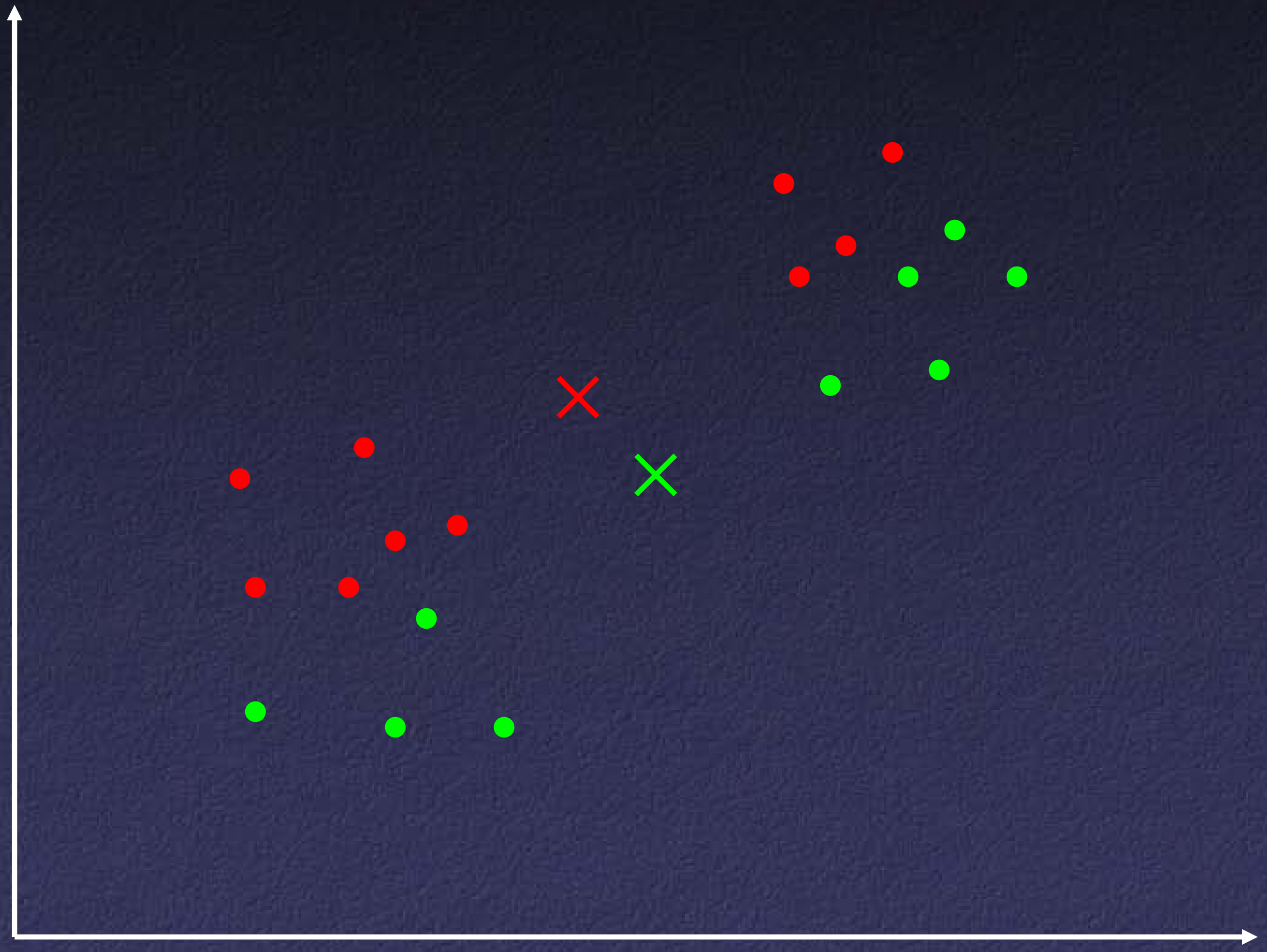
k-means



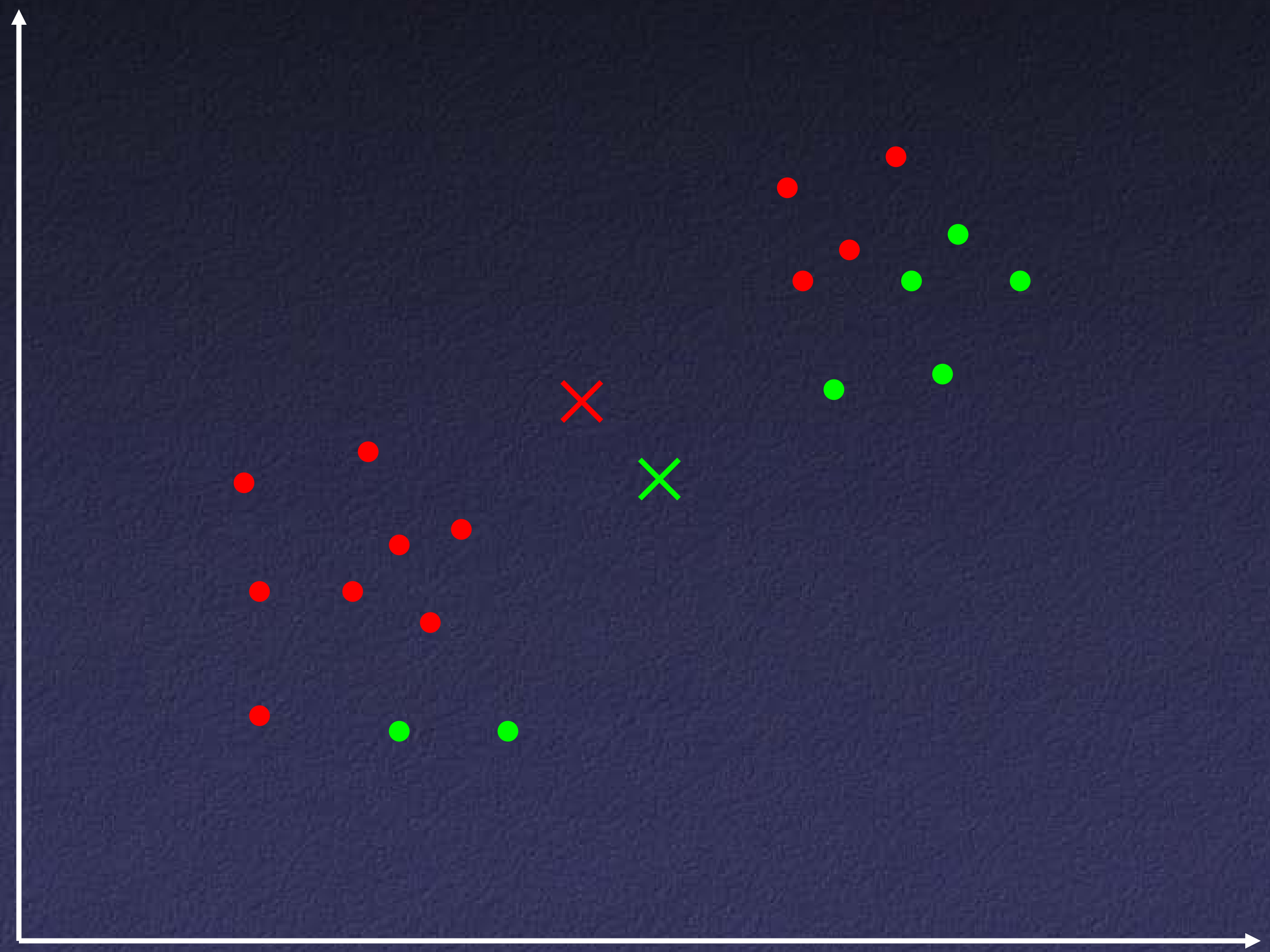
k-means



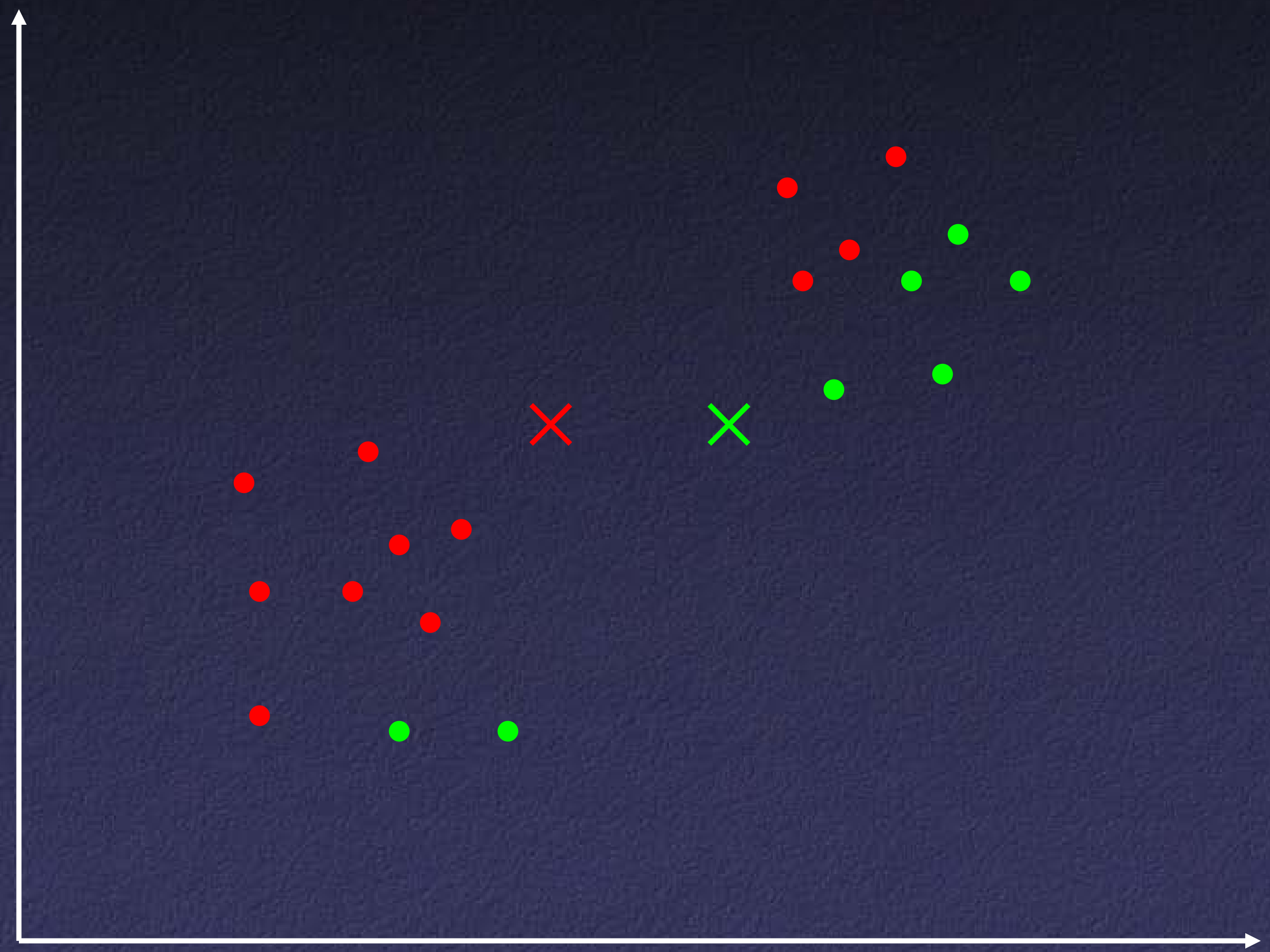
k-means



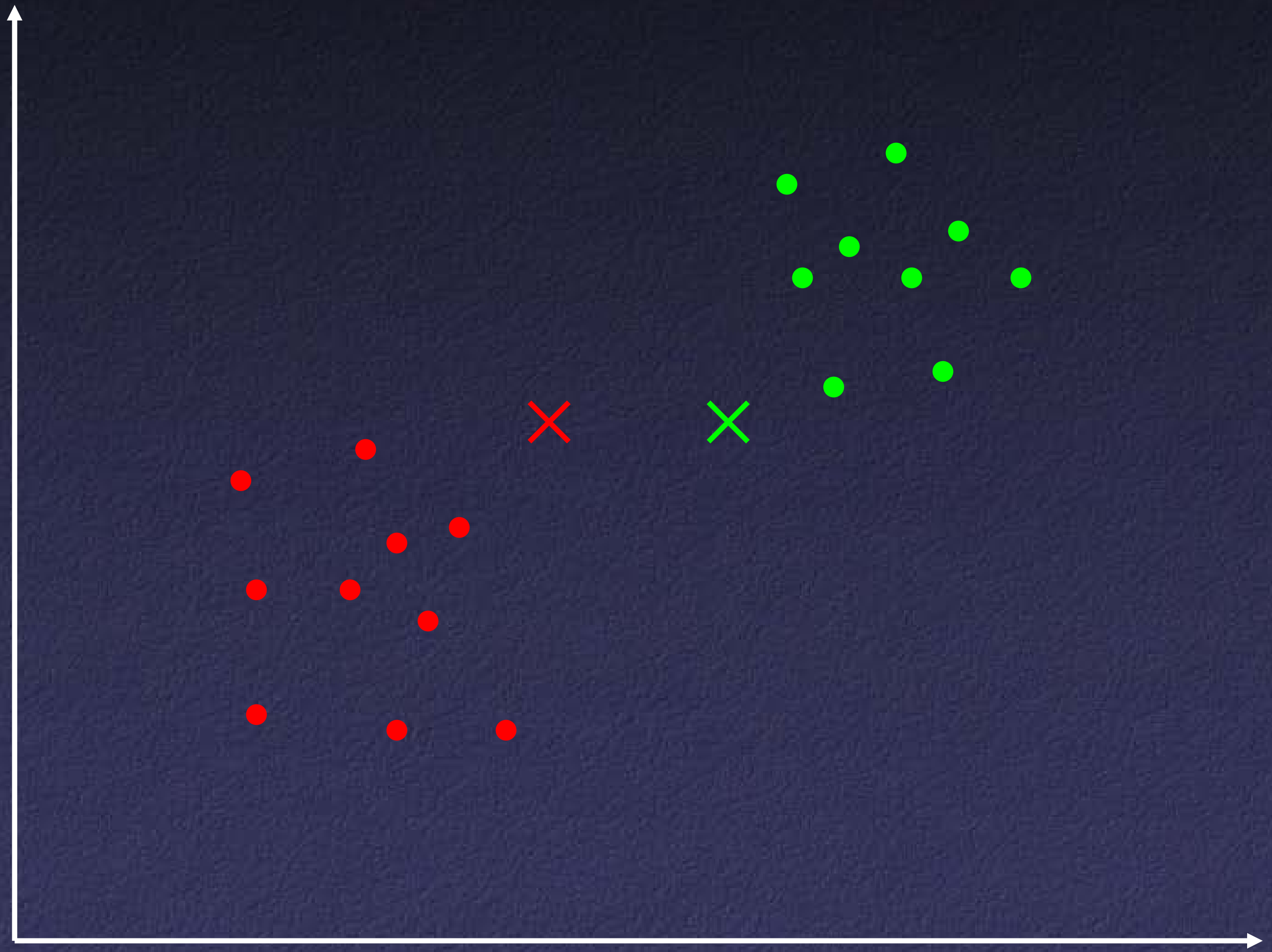
k-means



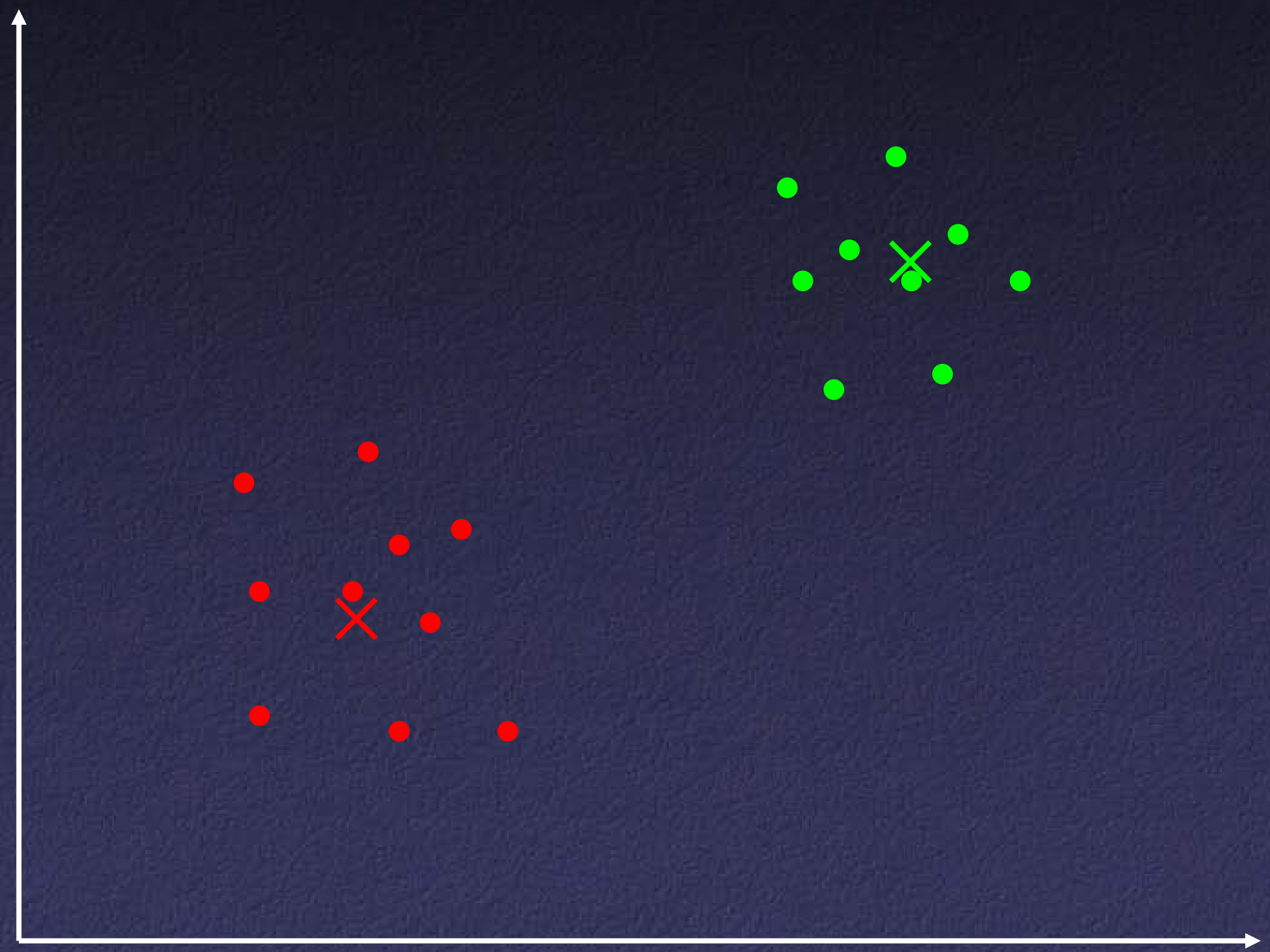
k-means



k-means



k-means



k-means

- This process always converges (to something)
  - Not necessarily globally-best assignment
- Informal proof: look at energy minimization

$$\mathcal{E} = \sum_{i \in \text{points}} \sum_{j \in \text{clusters}} \left\| x_i - \bar{x}_j \right\|^2 \cdot assigned_{ij}$$

  - Reclassifying points reduces (or maintains) energy
  - Recomputing centers reduces (or maintains) energy
  - Can't reduce energy forever

“Probabilistic *k*-means”

- Use Gaussian probabilities to assign point ↔ cluster weights

$$\pi_{p,j} = \frac{G_j(p)}{\sum_{j'} G_{j'}(p)}$$

“Probabilistic *k*-means”

- Use $\pi_{p,j}$ to compute weighted average and covariance for each cluster

$$\mu_j = \frac{\sum p\pi_{p,j}}{\sum \pi_{p,j}}$$

$$\Sigma_j = \frac{\sum (p-\mu_j)(p-\mu_j)^{\mathrm{T}} \pi_{p,j}}{\sum \pi_{p,j}}$$

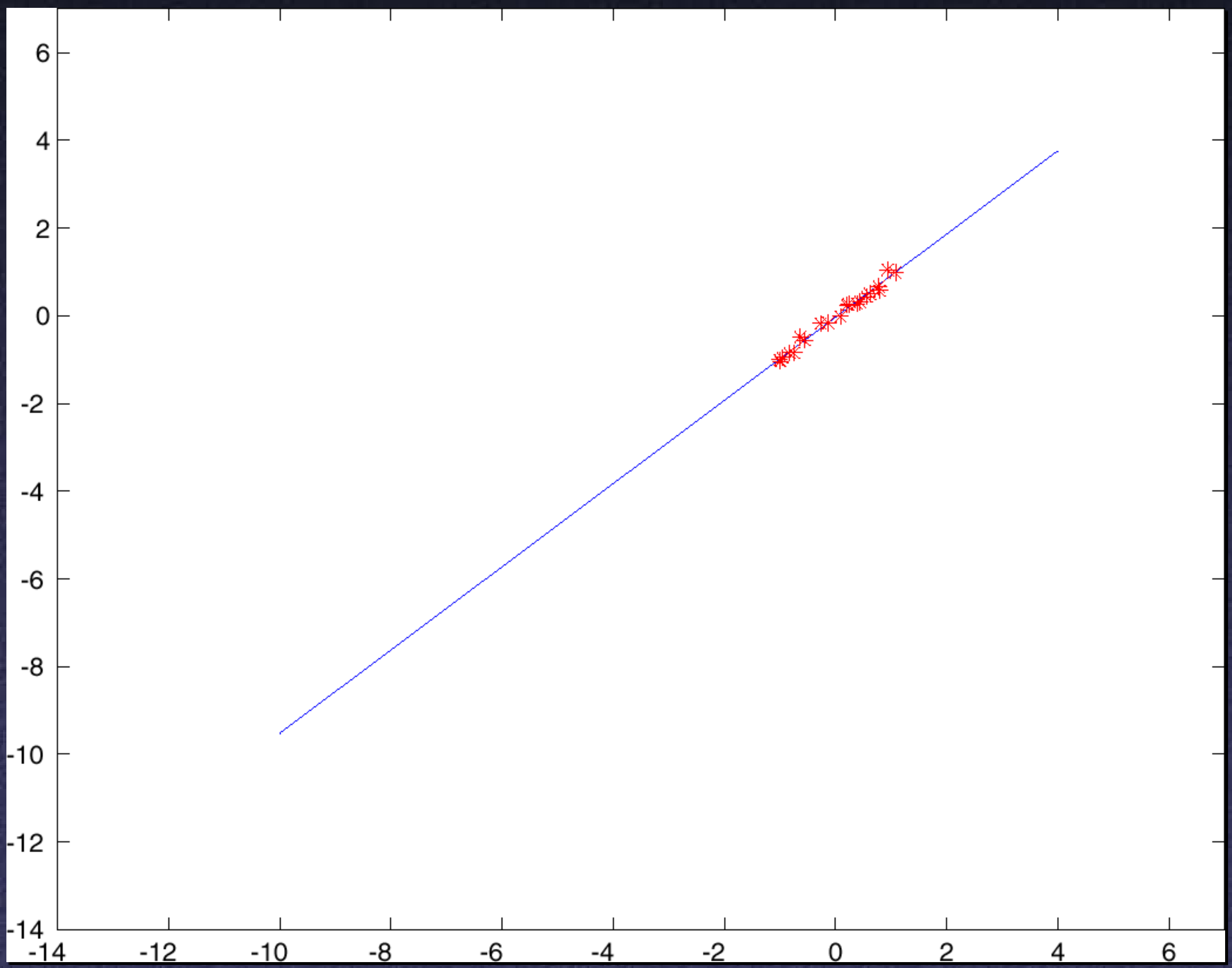
Expectation Maximization

- This is a special case of the expectation maximization algorithm
- General case: “missing data” framework
  - Have known data (feature vectors) and unknown data (assignment of points to clusters)
  - E step: use known data and current estimate of model to estimate unknown
  - M step: use current estimate of complete data to solve for optimal model

EM and Robustness

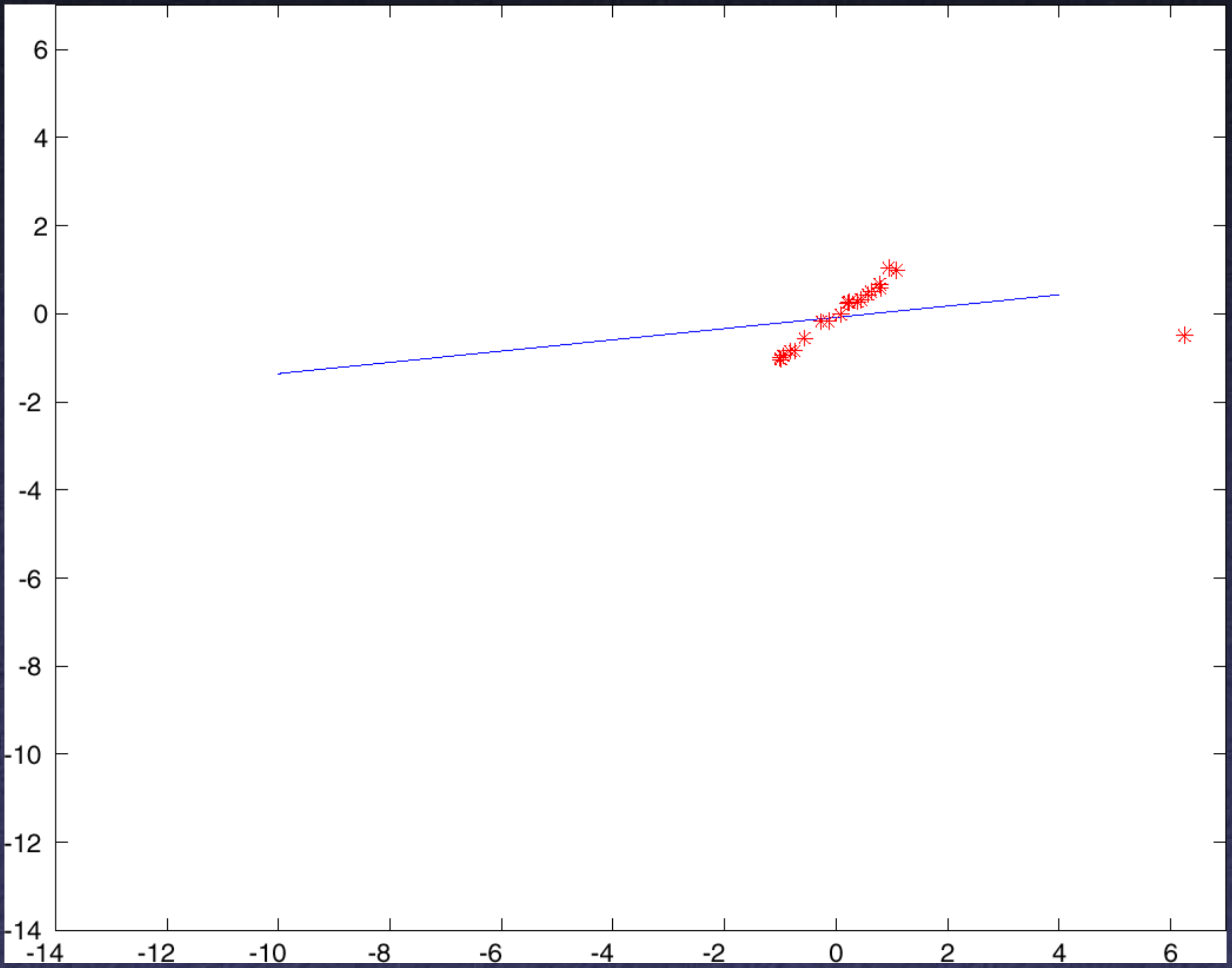
- One example of using generalized EM framework: robustness
- Make one category correspond to "outliers"
  - Use noise model if known
  - If not, assume e.g. uniform noise
  - Do not update parameters in M step

Example: Using EM to Fit to Lines



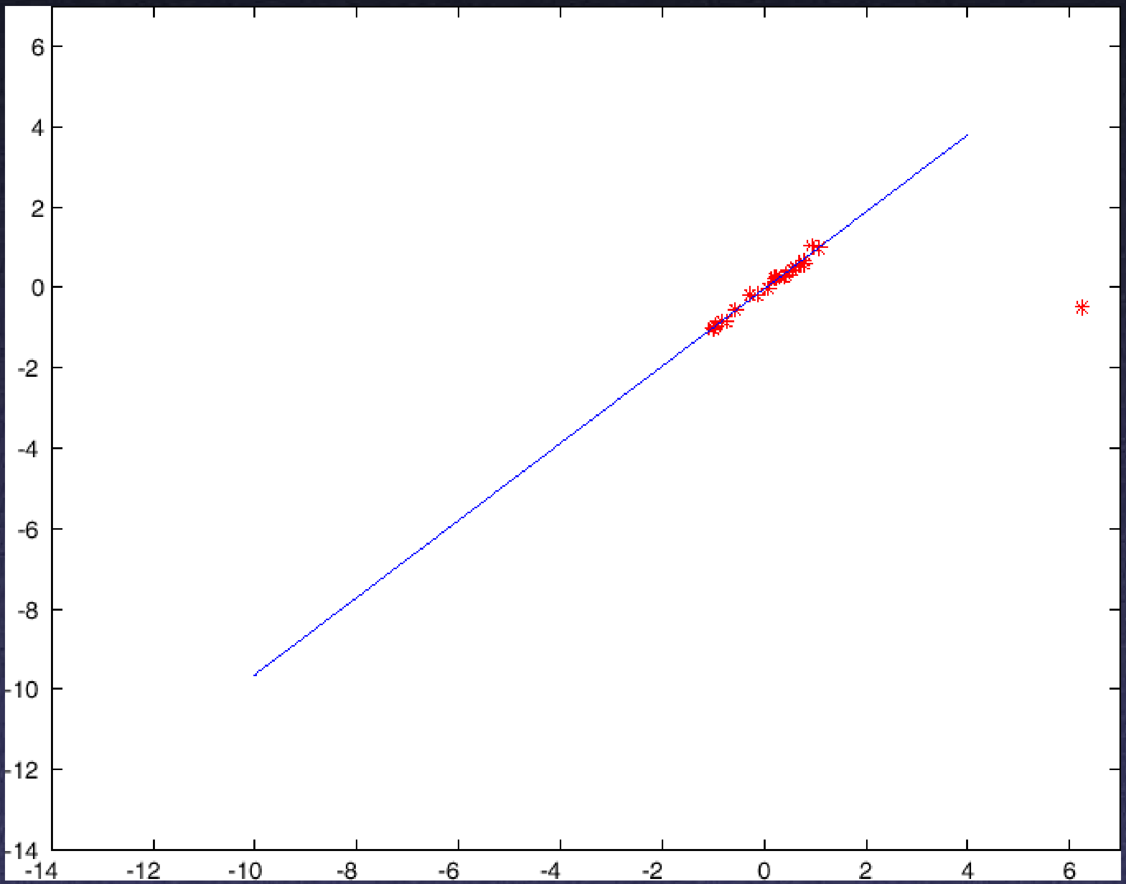
Good data

Example: Using EM to Fit to Lines

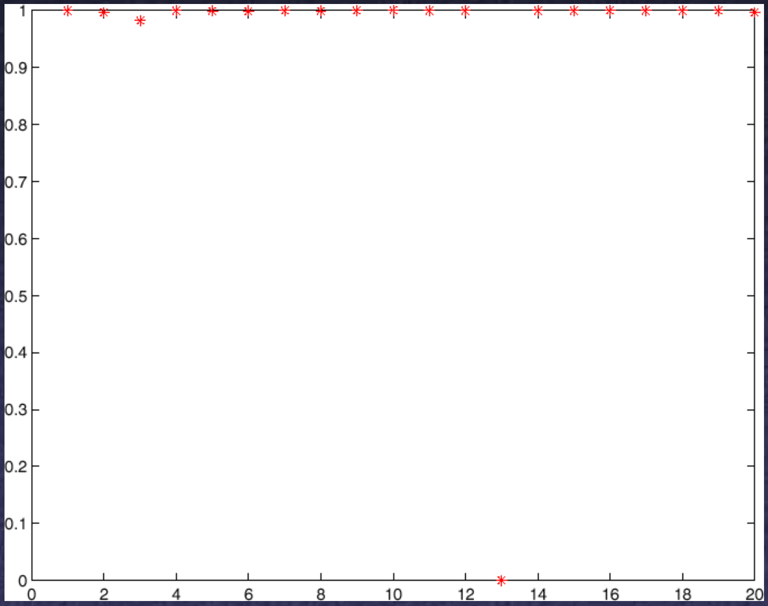


With outlier

Example: Using EM to Fit to Lines

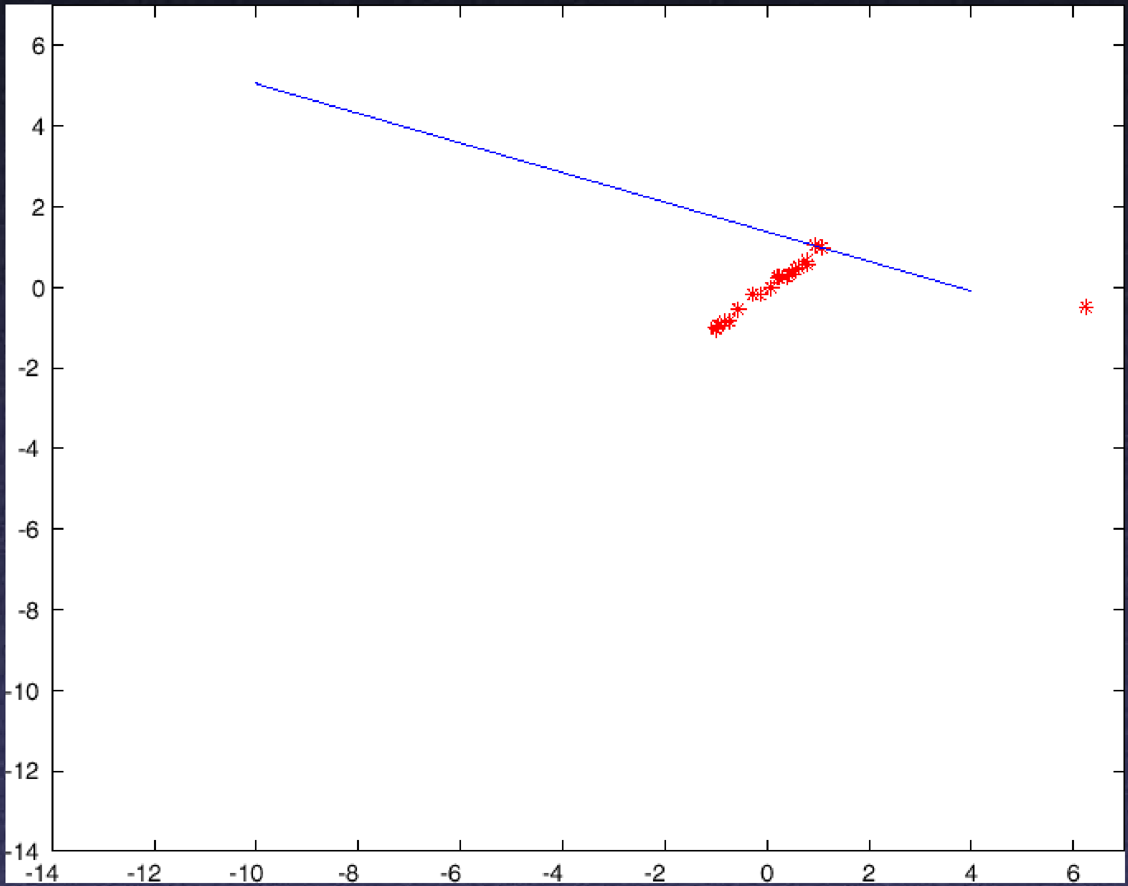


EM fit

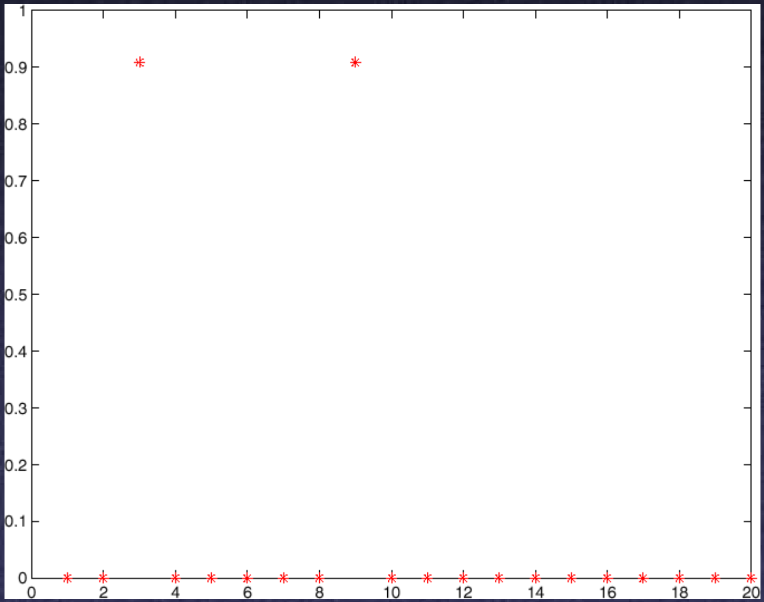


Weights of "line"
(vs. "noise")

Example: Using EM to Fit to Lines

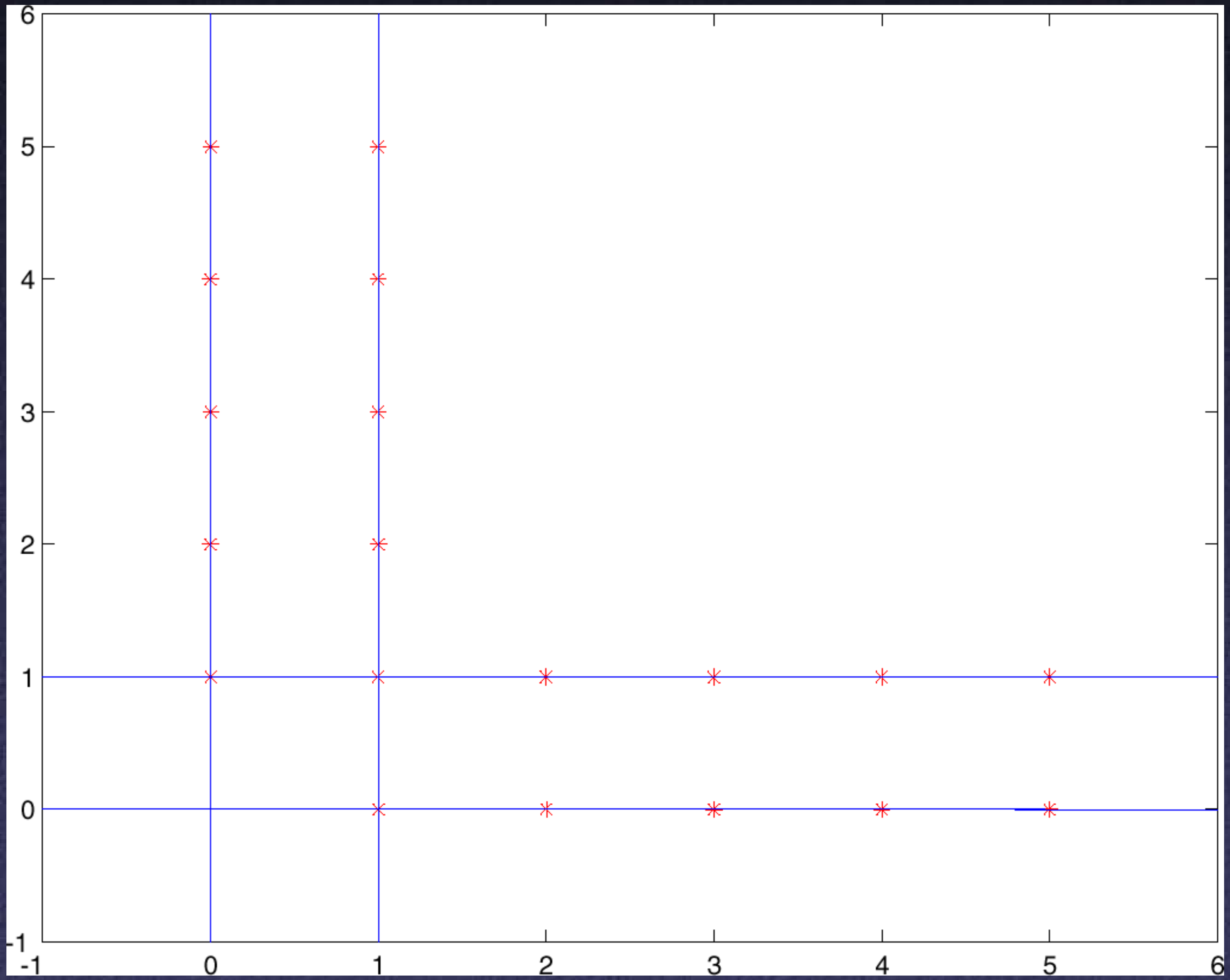


EM fit – bad local minimum



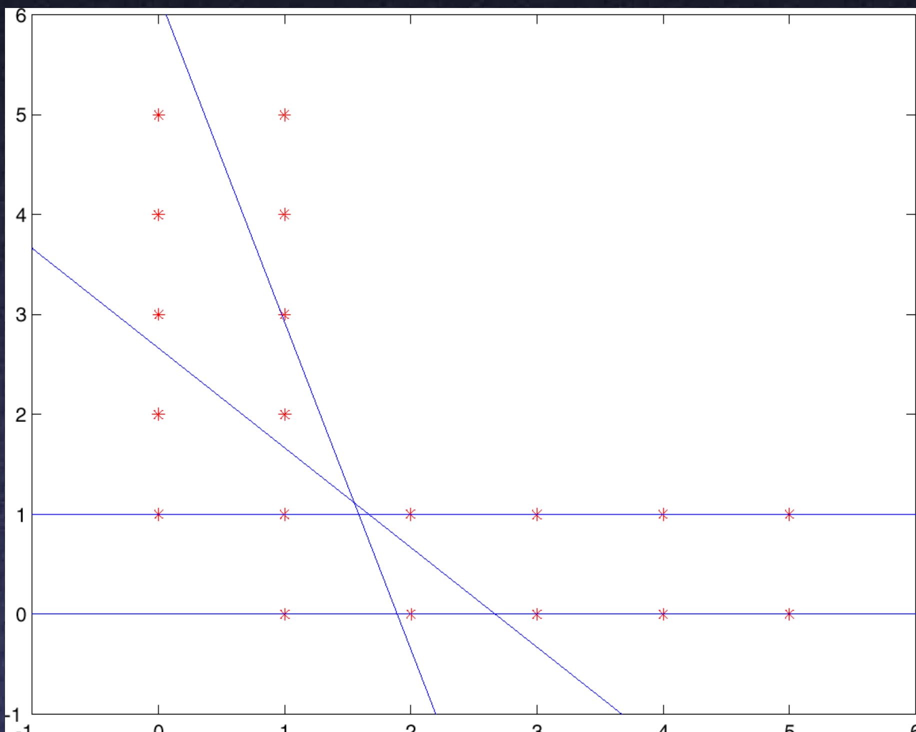
Weights of “line”
(vs. “noise”)

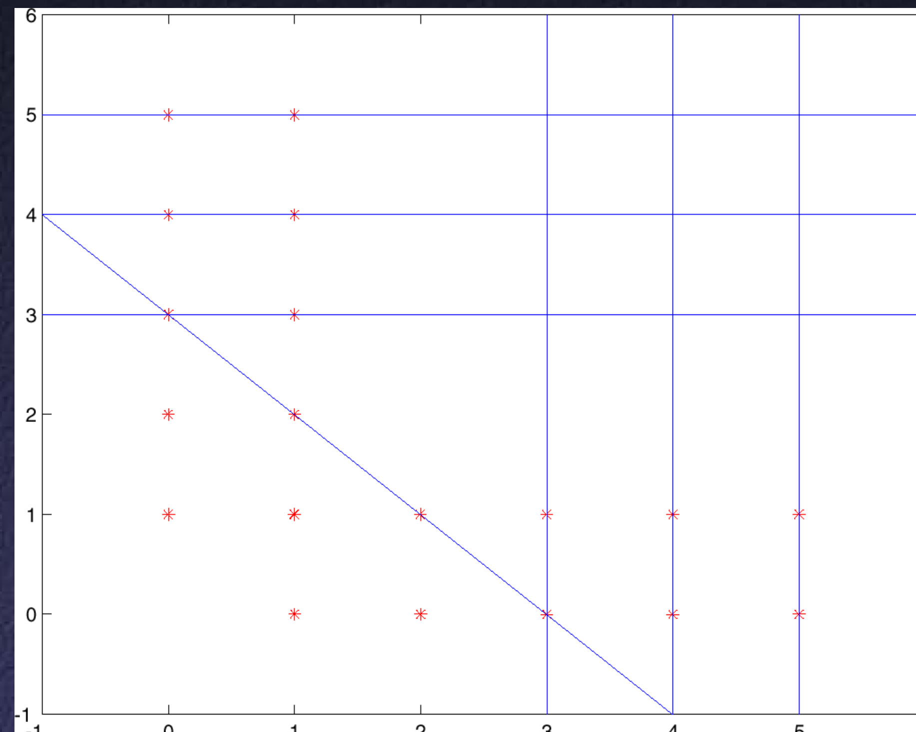
Example: Using EM to Fit to Lines



Fitting to multiple lines

Example: Using EM to Fit to Lines





Local minima

Weighted Observations

- In some applications, the datapoints are pixels
  - Weighted by intensity

$$\vec{\mu}_j = \frac{\sum w_p \pi_{p,j} \vec{x}_p}{\sum w_p \pi_{p,j}}$$

$$\Sigma_j = \frac{\sum w_p \pi_{p,j} (\vec{x}_p - \vec{\mu}_j)(\vec{x}_p - \vec{\mu}_j)^{\mathrm{T}}}{\sum w_p \pi_{p,j}}$$

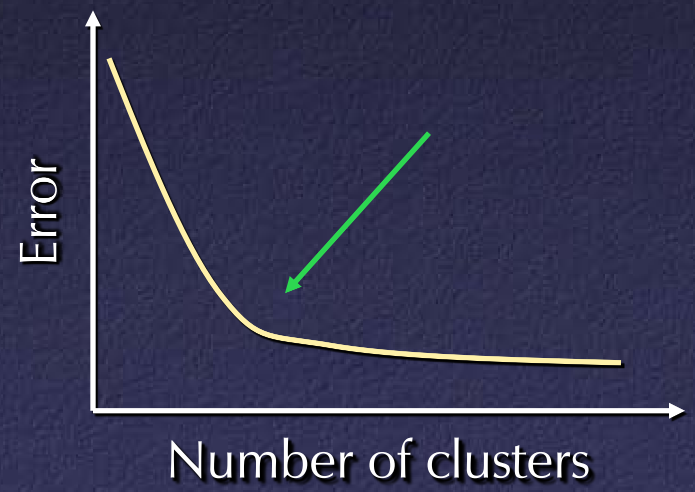
EM Demo

Eliminating Local Minima

- Re-run with multiple starting conditions
- Evaluate results based on
  - Number of points assigned to each (non-noise) group
  - Variance of each group
  - How many starting positions converge to each local maximum
- With many starting positions, can accommodate many outliers

Selecting Number of Clusters

- Re-run with different numbers of clusters, look at total error
- Will often see "knee" in the curve



Noise in data vs. error in model

Overfitting

- Why not use many clusters, get low error?
- Complex models bad at filtering noise (with *k* clusters can fit *k* data points exactly)
- Complex models have less predictive power
- Occam's razor: *entia non multiplicanda sunt praeter necessitatem* ("Things should not be multiplied beyond necessity")

Training / Test Data

- One way to see if you have overfitting problems:
  - Divide your data into two sets
  - Use the first set ("training set") to train model
  - Compute error on the second set of data ("test set")
  - If error not comparable to training, have overfitting