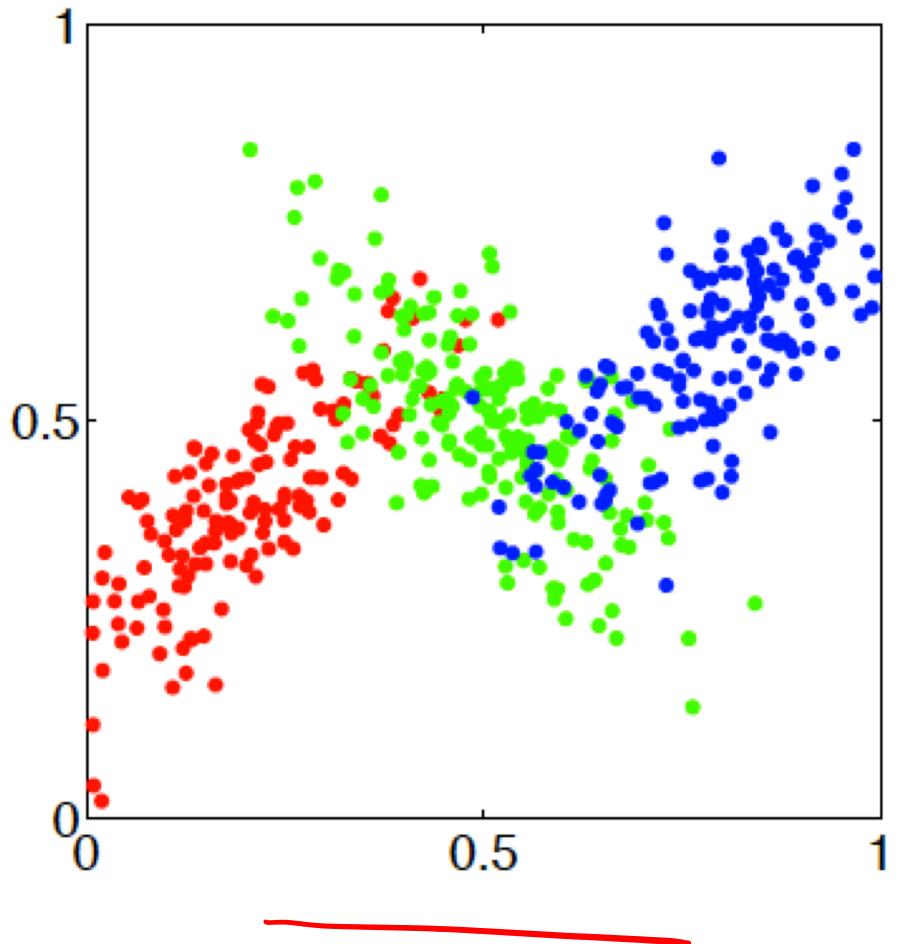
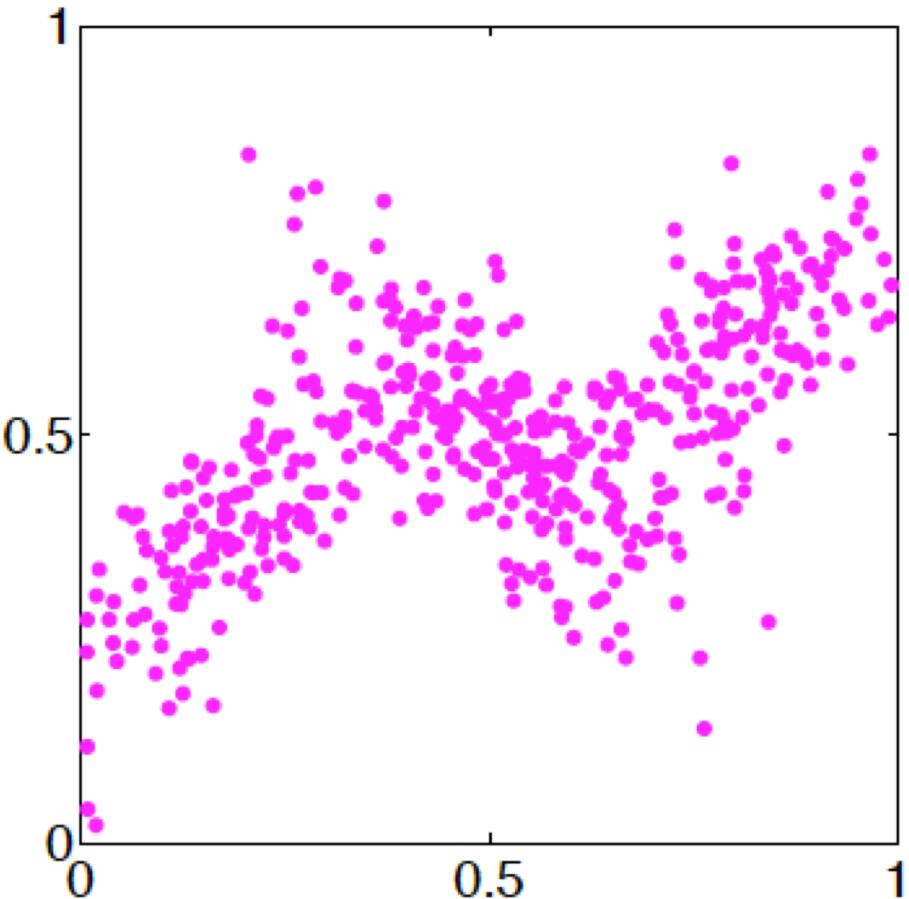


Introduction to Machine Learning

Dealing with missing data
(Unsupervised learning via prob. modeling)

Prof. Andreas Krause
Learning and Adaptive Systems (las.ethz.ch)

What if we assume $P(X | Y)$ is Gaussian?



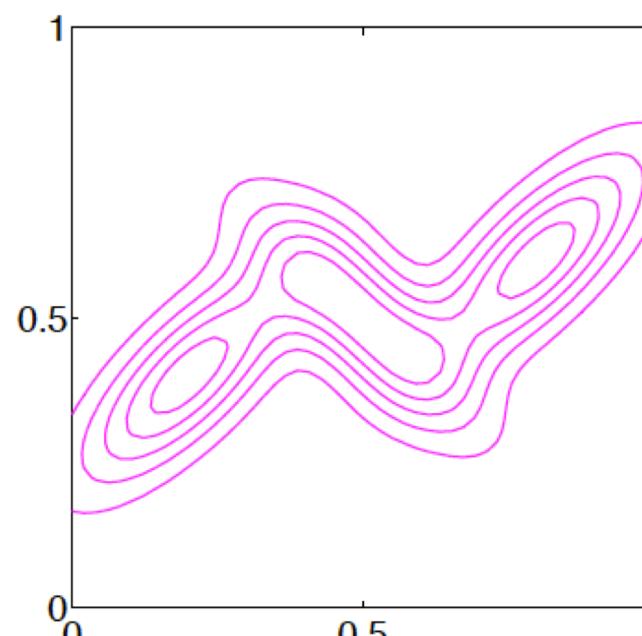
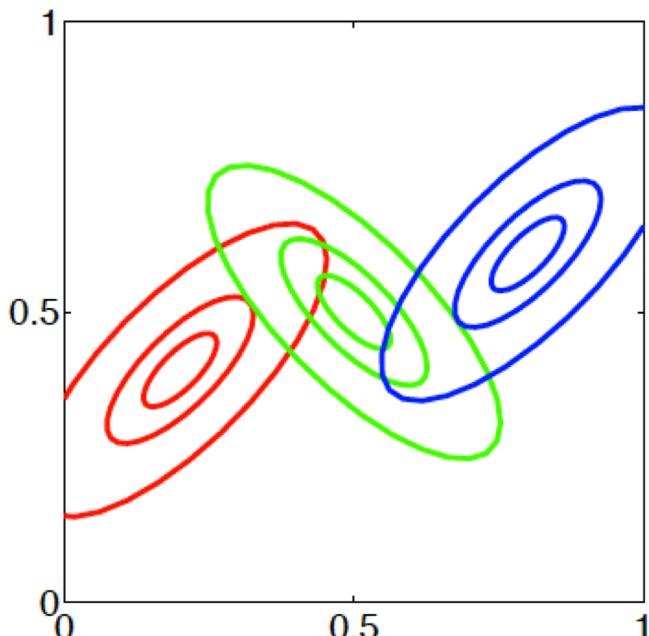
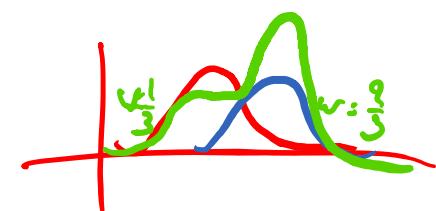
Gaussian mixtures

- Convex-combination of Gaussian distributions

$$P(\mathbf{x} \mid \theta) = P(\mathbf{x} \mid \mu, \Sigma, \mathbf{w}) = \sum_{i=1}^k w_i \mathcal{N}(\mathbf{x}; \mu_i, \Sigma_i)$$

where $w_i \geq 0$ and

$$\sum_i w_i = 1$$



GMMs vs. Gaussian Bayes Classifiers

- The joint distribution $P(z, \mathbf{x}) = w_z \mathcal{N}(\mathbf{x} \mid \mu_z, \Sigma_z)$ of *(cluster_index, features)* is identical to the generative model used by the Gaussian Bayes Classifier
- **Main difference:** In contrast to GBCs, in GMMs the (label) variable z is *unobserved!*
 - ➔ Fitting a GMM = Training a GBC without labels
 - ➔ Clustering = latent variable modeling
 - ➔ If we could just get the labels, could compute MLE in closed form!

Maximum likelihood estimation

- Can show: At MLE

$$(\mu^*, \Sigma^*, w^*) = \arg \min_i - \sum_{j=1}^k \log \sum w_j \mathcal{N}(\mathbf{x}_i \mid \mu_j, \Sigma_j)$$

it must hold that

$$\mu_j^* = \frac{\sum_{i=1}^n \gamma_j(\mathbf{x}_i) \mathbf{x}_i}{\sum_{i=1}^n \gamma_j(\mathbf{x}_i)}$$

$$\Sigma_j^* = \frac{\sum_{i=1}^n \gamma_j(\mathbf{x}_i) (\mathbf{x}_i - \mu_j^*) (\mathbf{x}_i - \mu_j^*)^T}{\sum_{i=1}^n \gamma_j(\mathbf{x}_i)}$$

$$\underline{w_j^*} = \frac{1}{n} \sum_{i=1}^n \gamma_j(\mathbf{x}_i)$$

These equations are *coupled* → difficult to solve jointly

Expectation-Maximization (Soft-EM)

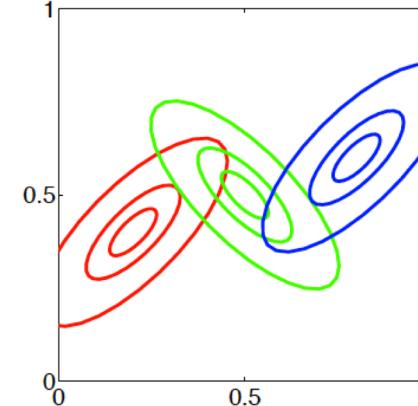
- While not converged
 - E-step: calculate cluster membership weights (“Expected sufficient statistics”) for each point (aka “responsibilities”):
Calculate $\gamma_j^{(t)}(\mathbf{x}_i)$ for each i and j given estimates of $\mu^{(t-1)}$, $\Sigma^{(t-1)}$, $\mathbf{w}^{(t-1)}$ from previous iteration
 - M-step: Fit clusters to weighted data points
(closed form Maximum likelihood solution!)

$$w_j^{(t)} \leftarrow \frac{1}{n} \sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i) \quad \mu_j^{(t)} \leftarrow \frac{\sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i) \mathbf{x}_i}{\sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i)}$$

$$\Sigma_j^{(t)} \leftarrow \frac{\sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i) (\mathbf{x}_i - \mu_j^{(t)}) (\mathbf{x}_i - \mu_j^{(t)})^T}{\sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i)}$$

Why are mixture models useful?

- Can encode assumptions about “shape” of clusters
 - E.g., fit ellipses instead of points

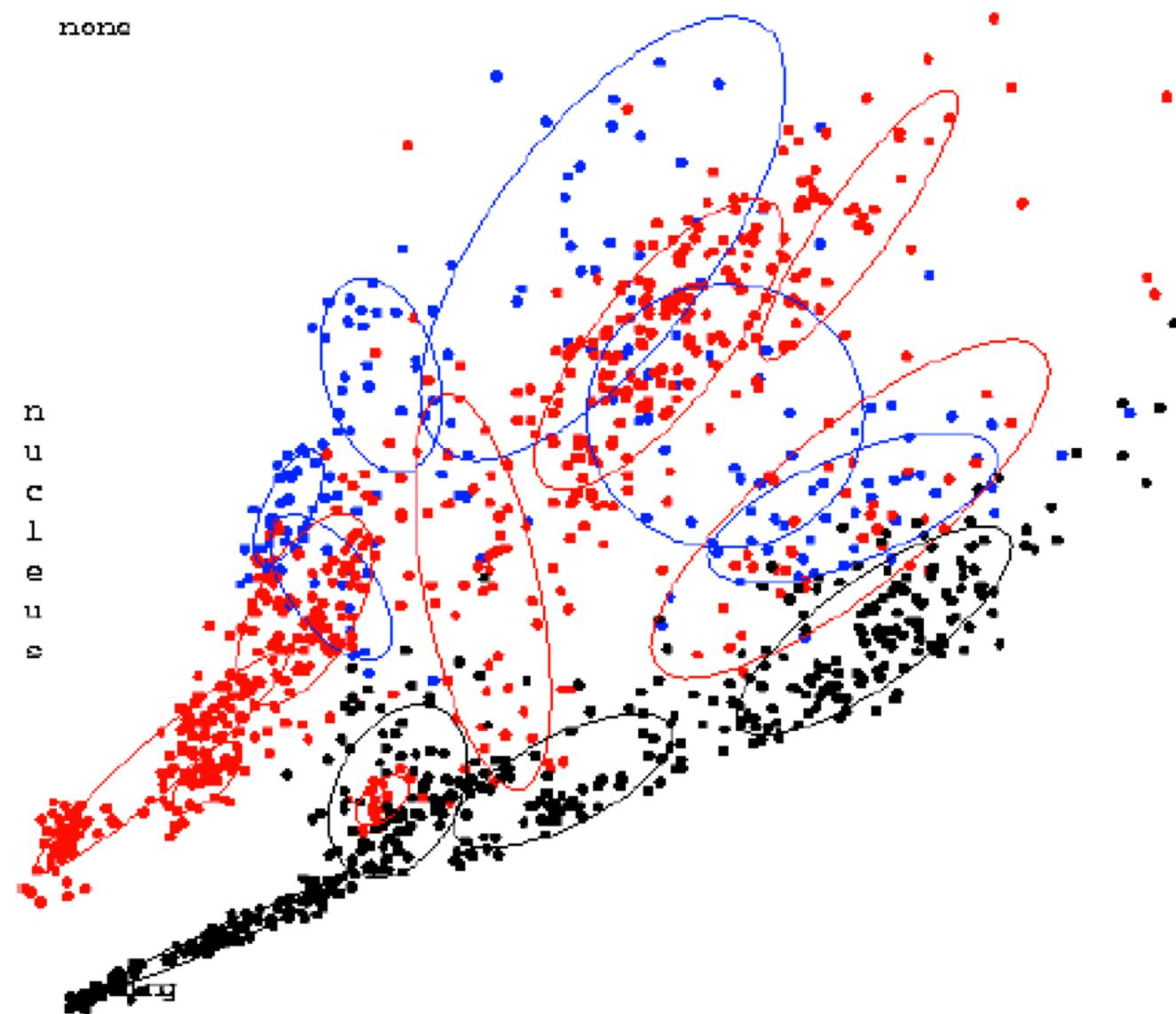


- Can be part of more complex statistical models
 - E.g., classifiers (or more general probabilistic models)
- Probabilistic models can output likelihood $P(x)$ of a point x
 - Useful for anomaly/outlier detection
- Can be naturally used for semi-supervised learning!

Clustering for (nonlinear) classification

Compound =
IL-1
TNF
none

[Andrew Moore]



Gaussian-Mixture Bayes classifiers

- Given *labeled* data set $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$
 - Label $y_i \in \{1, \dots, m\}$
 - Estimate class prior $P(y)$
 - Estimate conditional distribution *for each class*
- $$\underline{P(\mathbf{x} | y)} = \sum_{j=1}^{n_y} w_j^{(y)} \mathcal{N}(\mathbf{x}; \mu_j^{(y)}, \Sigma_j^{(y)})$$
as Gaussian mixture model
- How do we use this model for classification?

$$\begin{aligned} P(y|x) &= \frac{1}{2} P(y) P(x|y) \\ &= \frac{1}{2} P(y) \sum_{j=1}^{n_y} w_j^{(y)} \mathcal{N}(x; \mu_j^{(y)}, \Sigma_j^{(y)}) \end{aligned}$$

Gaussian-Mixture Bayes classifiers

- Given *labeled* data set $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$
 - Label $y_i \in \{1, \dots, m\}$
 - Estimate class prior $P(y)$
 - Estimate conditional distribution *for each class*

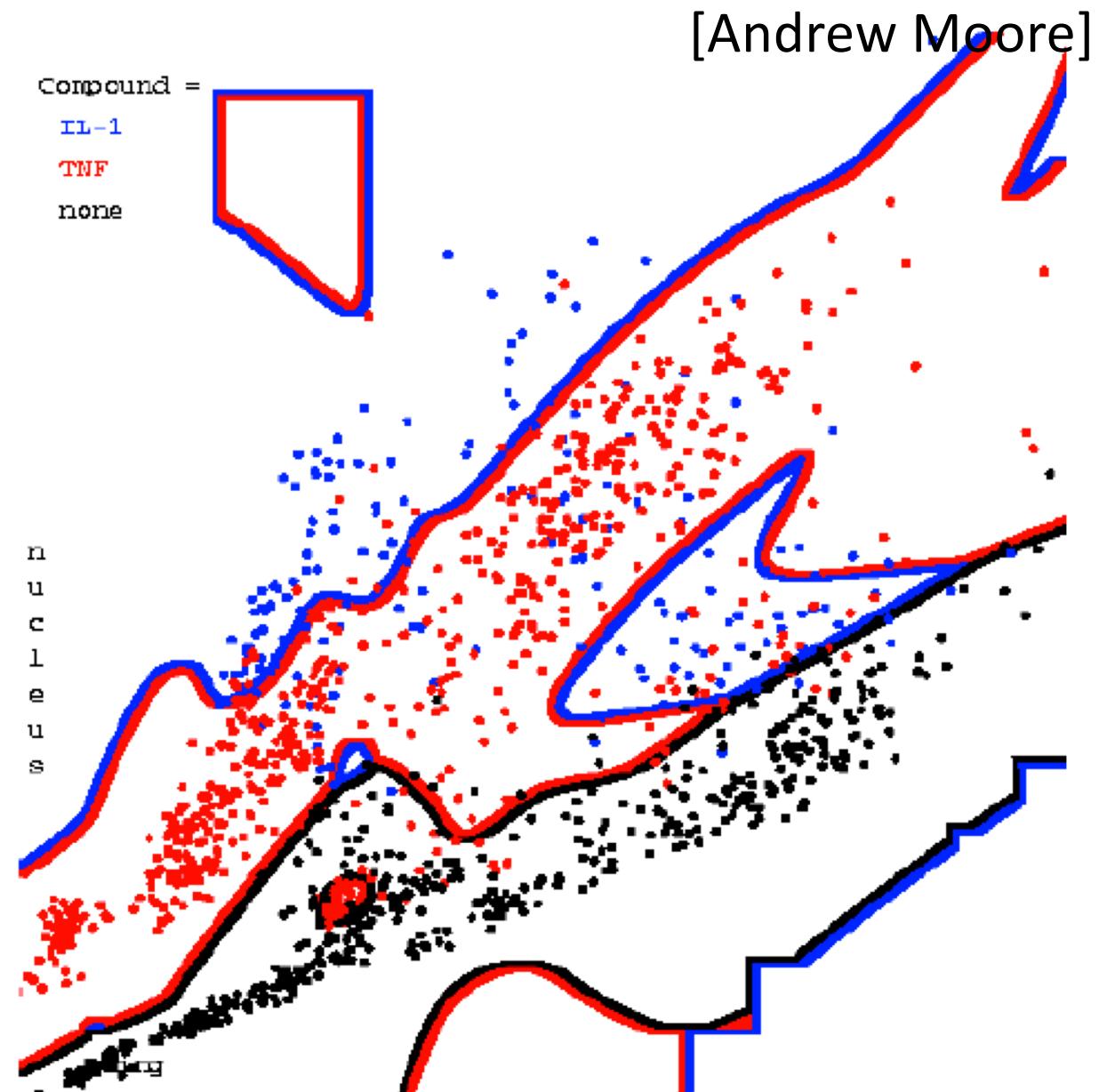
$$P(\mathbf{x} \mid y) = \sum_j w_j^{(y)} \mathcal{N}(\mathbf{x}; \mu_j^{(y)}, \Sigma_j^{(y)})$$

as Gaussian mixture model

- How do we use this model for classification?

$$P(y \mid \mathbf{x}) = \frac{1}{Z} p(y) \sum_{j=1}^{k_y} w_j^{(y)} \mathcal{N}(\mathbf{x}; \mu_j^{(y)}, \Sigma_j^{(y)})$$

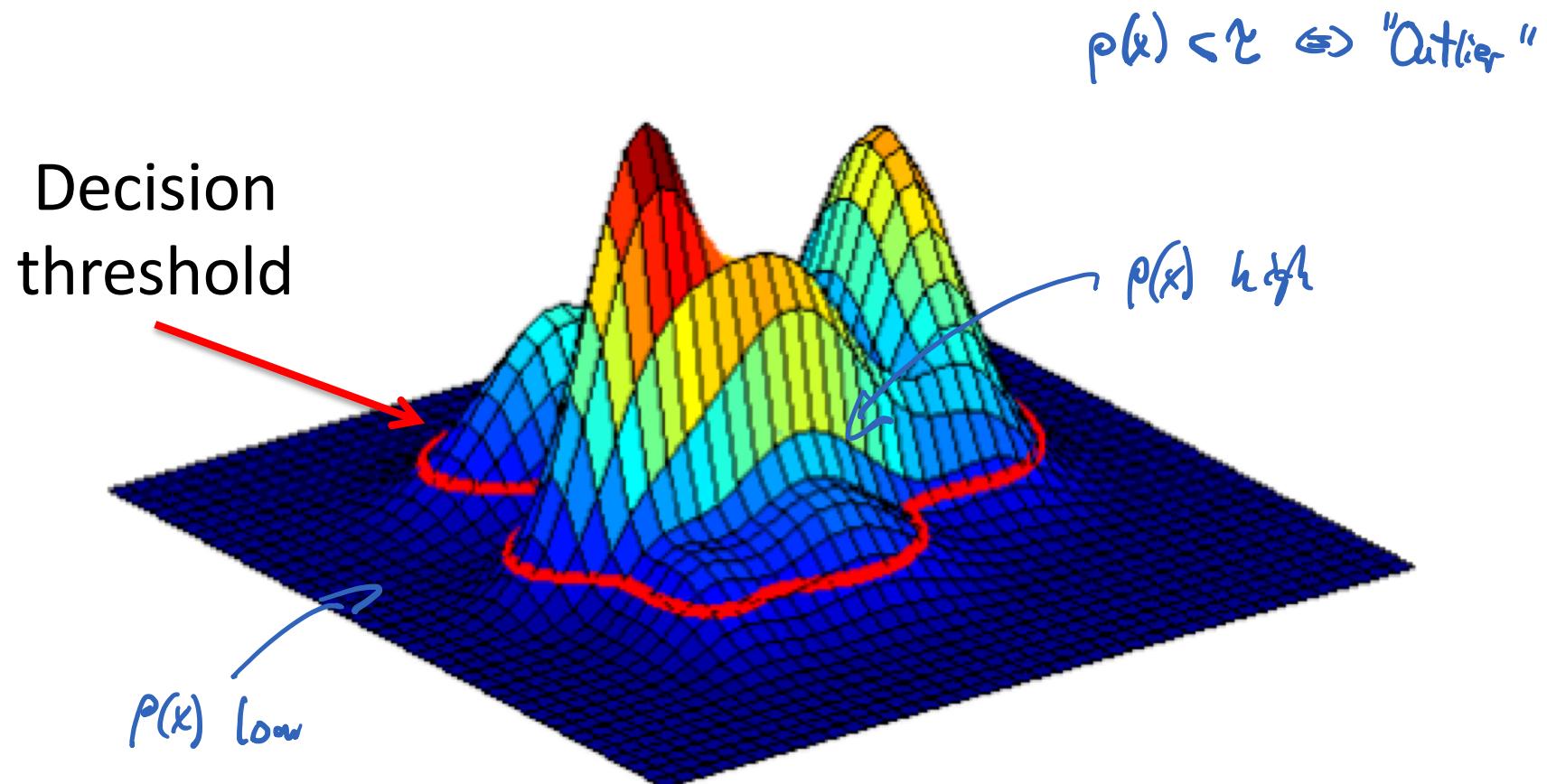
Resulting classifier



GMMs for density estimation

- We may be interested in fitting a Gaussian Mixture Model not for clustering but for **density estimation**
- E.g.,
 - Model $P(\mathbf{x})$ as Gaussian mixture
 - Model $P(y|\mathbf{x})$ using logistic regression, neural network, etc.
- Combines the advantage of accurate predictions / robustness from discriminative model, with the ability to detect outliers!

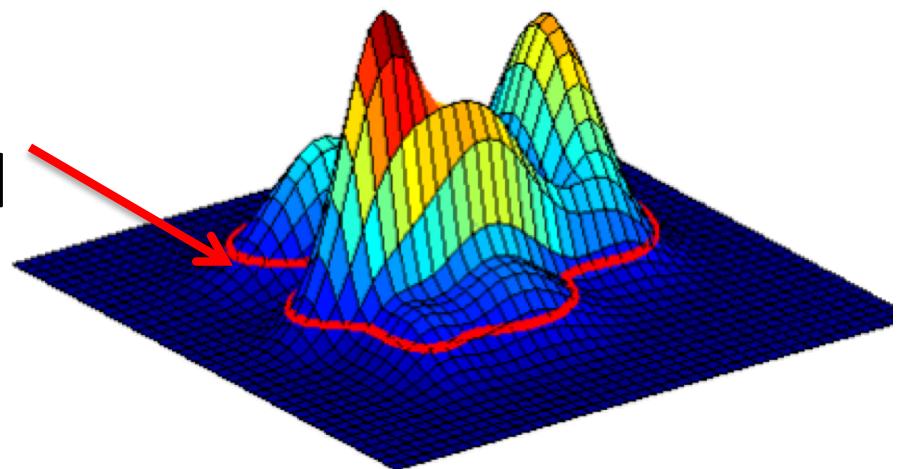
Anomaly detection with mixture models



- Can determine outliers by comparing the estimated density of a data point against a threshold

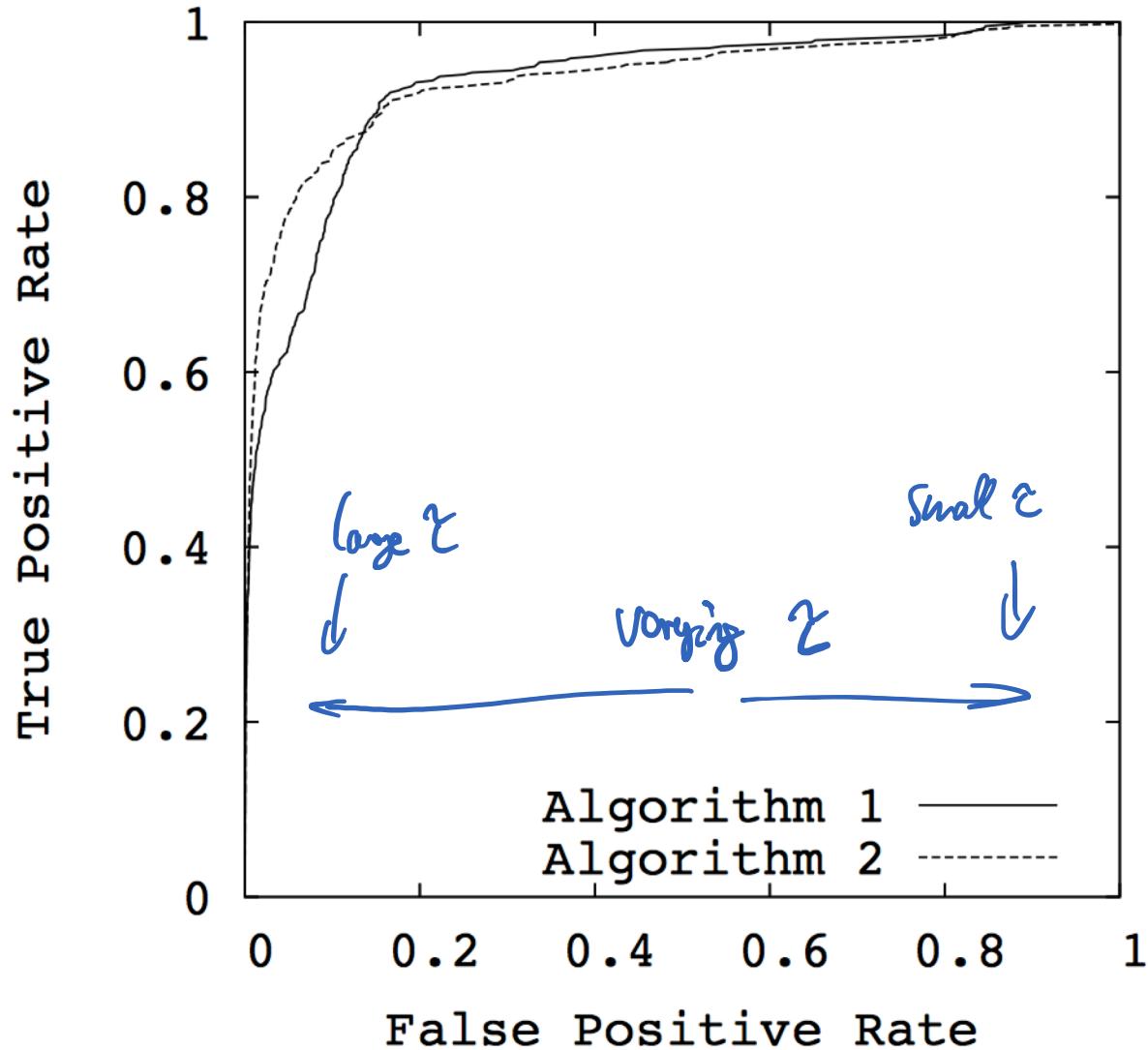
How to pick the threshold?

Decision
threshold

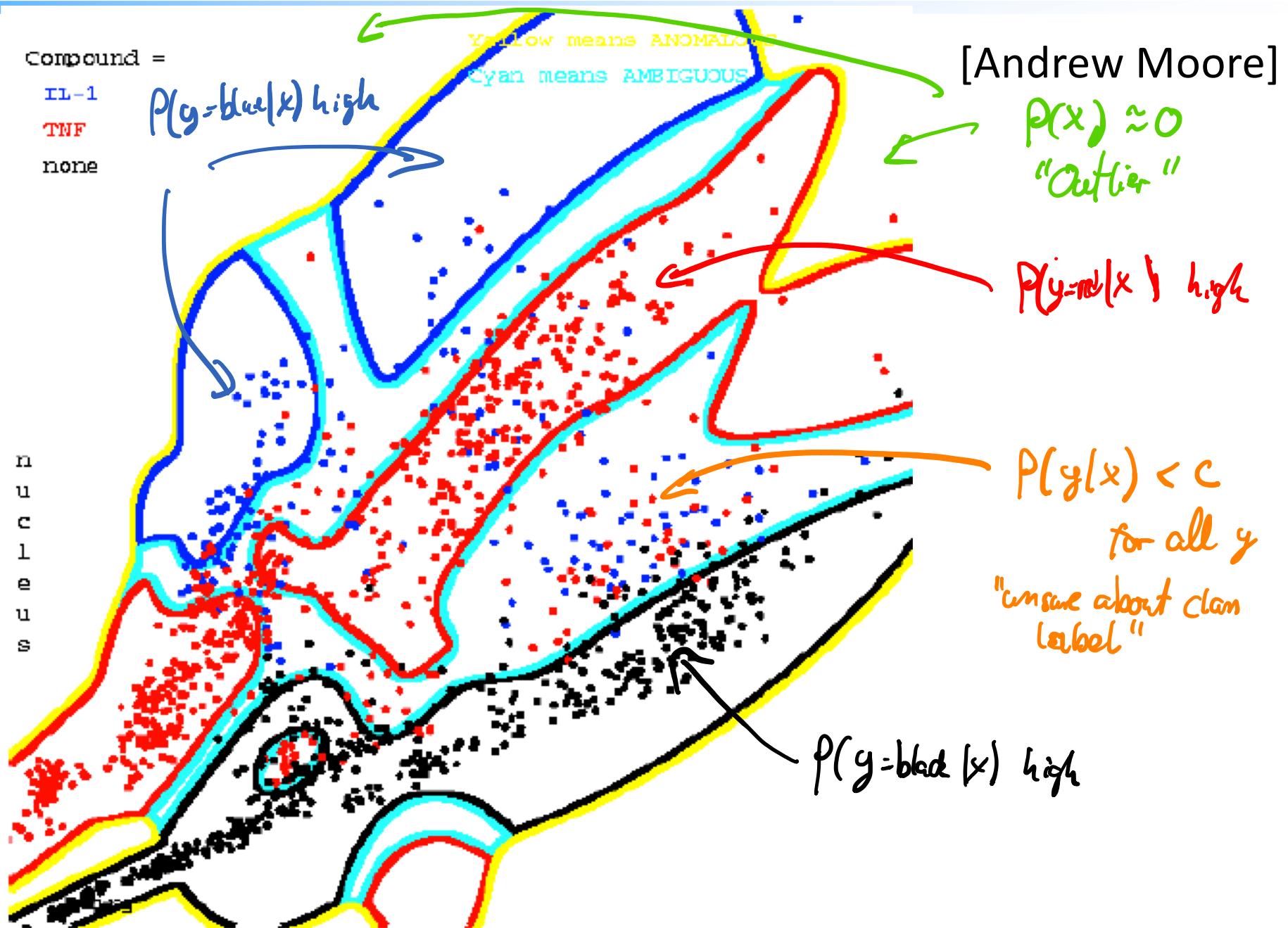


- If we have no examples of anomalies, this is challenging
- If we do have some examples:
 - Varying the threshold trades false-positives and false-negatives
 - Can use precision-recall curves / ROC curves as evaluation criterion; e.g., maximize F1 score
 - This allows to optimize the threshold, e.g., via cross-validation!

Recall: ROC curves



Classification + Anomaly detection



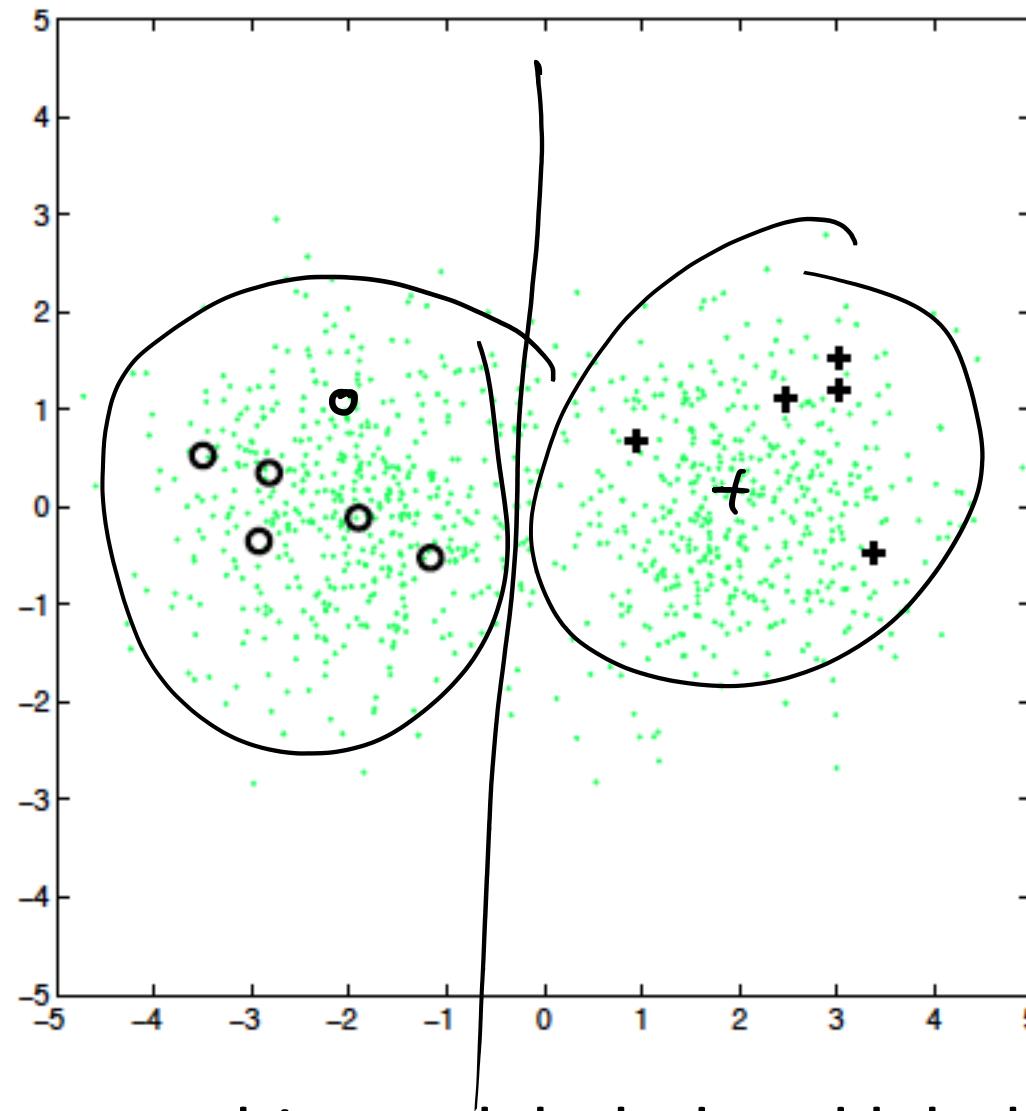
GMMs for density estimation

- We may be interested in fitting a Gaussian Mixture Model not for clustering but for **density estimation**
- E.g.,
 - Model $P(x)$ as Gaussian mixture
 - Model $P(y/x)$ using logistic regression, neural network, etc.
- Combines the advantage of accurate predictions / robustness from discriminative model, with the ability to detect outliers / anomalies!

$$P(x,y) = P(y) P(x|y) = p(x) p(y|x)$$

Semi-supervised learning with GMMs

[Zhu]



- Want to combine unlabeled and labeled data!

Semi-supervised learning = partially missing labels

X_1	X_2	...	X_d	Y
1.1	.35	N/A
.5	.47	+1
...
-.4	-.53	N/A

Semi-supervised learning with GMMs

- We have discussed
 - **Supervised Learning:** Data consists of (feature,label)-pairs
 - **Unsupervised Learning:** Data consists of feature vectors **only**
- Often, one can obtain large amounts of unlabeled data, but labeled data is expensive
- **Semi-supervised learning** = learning from (large amounts of) unlabeled data **and** (small amounts of) labeled data
- Easy to do with Gaussian mixture models!

Semi-supervised learning with GMMs

- Recall, for GMMs, at the MLE it holds that

$$\mu_j^* = \frac{\sum_{i=1}^n \gamma_j(\mathbf{x}_i) \mathbf{x}_i}{\sum_{i=1}^n \gamma_j(\mathbf{x}_i)}$$

$$\Sigma_j^* = \frac{\sum_{i=1}^n \gamma_j(\mathbf{x}_i) (\mathbf{x}_i - \mu_j^*) (\mathbf{x}_i - \mu_j^*)^T}{\sum_{i=1}^n \gamma_j(\mathbf{x}_i)}$$

$$w_j^* = \frac{1}{n} \sum_{i=1}^n \gamma_j(\mathbf{x}_i)$$

- where $\gamma_j(\mathbf{x}) = P(z = j \mid \mathbf{x}, \Sigma^*, \mu^*, \mathbf{w}^*)$

- In SSL, for instances \mathbf{x} with observed labels y , must hold:

$$\underline{\gamma_j(\mathbf{x}_i)} = [j = y_i]$$

EM for semi-supervised learning with GMMs

- While not converged

- E-step: For unlabeled points:

$$\gamma_j^{(t)}(\mathbf{x}_i) = P(Z = j \mid \mathbf{x}_i, \mu^{(t-1)}, \Sigma^{(t-1)}, \mathbf{w}^{(t-1)})$$

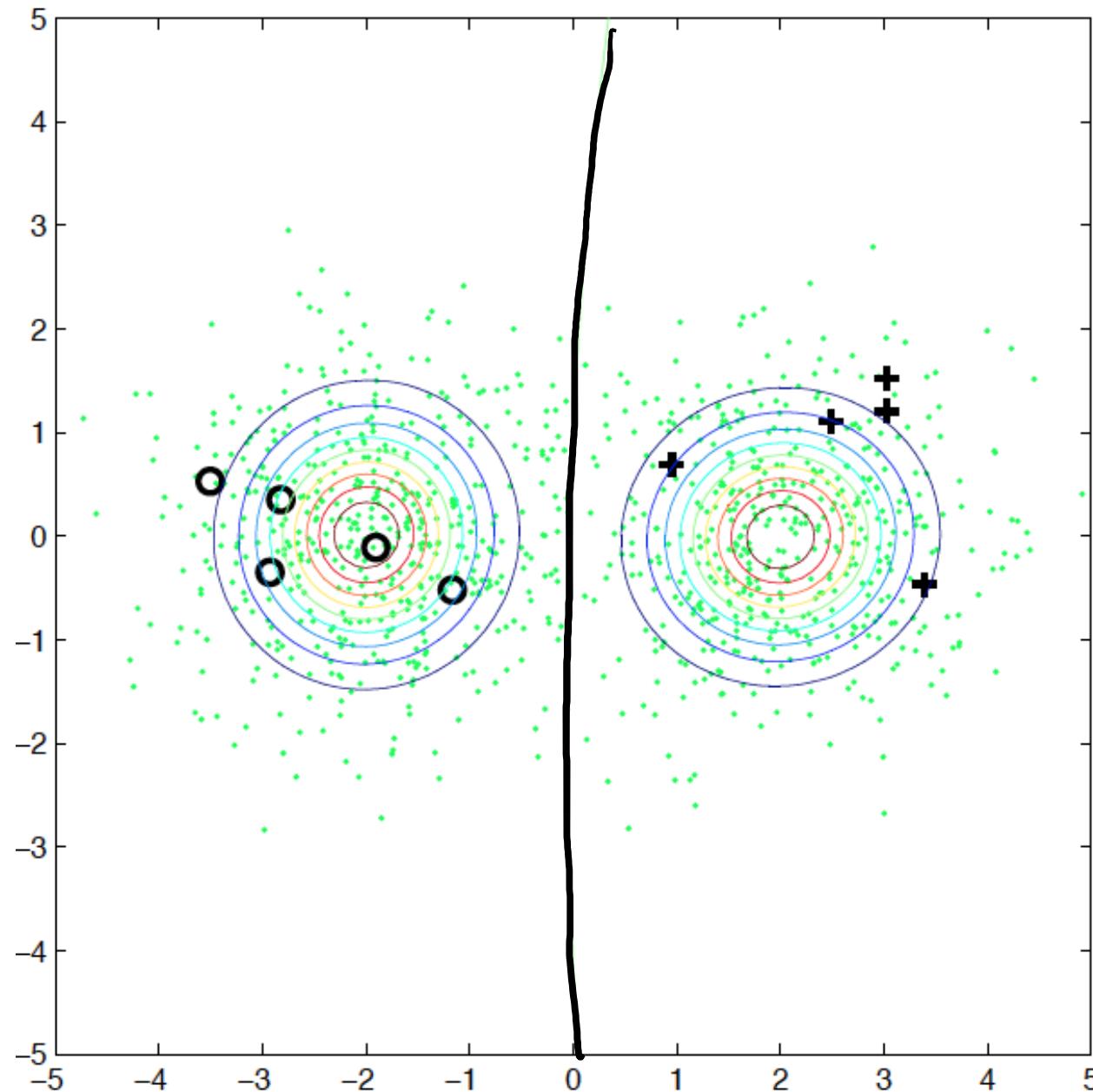
For points with label y_i : $\gamma_j^{(t)}(\mathbf{x}_i) = [j = y_i]$

- M-step: Fit clusters to weighted data points
(closed form Maximum likelihood solution!)

$$w_j^{(t)} \leftarrow \frac{1}{n} \sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i) \quad \mu_j^{(t)} \leftarrow \frac{\sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i) \mathbf{x}_i}{\sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i)}$$

$$\Sigma_j^{(t)} \leftarrow \frac{\sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i) (\mathbf{x}_i - \mu_j^{(t)}) (\mathbf{x}_i - \mu_j^{(t)})^T}{\sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i)}$$

Semi-supervised learning solution



Theory behind the EM Algorithm

- Can show that the EM algorithm is equivalent to the following procedure:
- E-Step: Calculate the expected complete data log-likelihood (= function of θ)

$$Q(\theta; \theta^{(t-1)}) = \mathbb{E}_{\mathbf{z}_{1:n}} \left[\log P(\mathbf{x}_{1:n}, \mathbf{z}_{1:n} \mid \theta) \mid \mathbf{x}_{1:n}, \theta^{(t-1)} \right]$$
$$= \sum_{\mathbf{z}_{1:n}} \underbrace{\log P(\mathbf{z}_{1:n} \mid \mathbf{x}_{1:n}, \theta^{(t-1)})}_{\text{Complete data log-likelihood}}$$

- M-Step: Maximize $\underline{\theta^{(t)}} = \arg \max_{\theta} Q(\theta; \underline{\theta^{(t-1)}})$

Details

Notation: $\tilde{\theta} := \theta^{(t-1)}$

$x := x_{1:n}$

$z := z_{1:n}$

- EM-Objective function

$$Q(\theta; \underbrace{\theta^{(t-1)}}_{\tilde{\theta}}) = \mathbb{E}_{z_{1:n}} \left[\log P(x_{1:n}, z_{1:n} \mid \theta) \mid x_{1:n}, \theta^{(t-1)} \right]$$

can be simplified:

$$\begin{aligned} Q(\theta; \tilde{\theta}) &\stackrel{iid}{=} \mathbb{E}_z \left[\sum_{i=1}^n \log P(x_i, z_i \mid \theta) \mid x, \tilde{\theta} \right] \\ &= \sum_{i=1}^n \mathbb{E}_{z_{1:n}} \left[\log P(x_i, z_i \mid \theta) \mid x_{1:n}, \tilde{\theta} \right] \\ &= \sum_{i=1}^n \mathbb{E}_{z_i} \left[\log P(x_i, z_i \mid \theta) \mid x_i, \tilde{\theta} \right] \\ &= \sum_{i=1}^n \underbrace{\sum_{j=1}^k p(z_i=j \mid x_i, \tilde{\theta})}_{p_j(x_i)} \underbrace{\log \underbrace{w_j \mathcal{N}(x_i; \mu_j, \Sigma_j)}_{w_j \mathcal{N}(x_i; \mu_j, \Sigma_j)}}_{\log w_j \mathcal{N}(x_i; \mu_j, \Sigma_j)} \end{aligned}$$

Details for E-Step

- Thus, the EM objective function is equivalent to

$$Q(\theta; \theta^{(t-1)}) = \sum_{i=1}^n \sum_{z_i=1}^k \gamma_{z_i}(\mathbf{x}_i) \log P(\mathbf{x}_i, z_i \mid \theta)$$

where $\gamma_z(\mathbf{x}) = P(z \mid \mathbf{x}, \theta^{(t-1)})$

- Thus, E-step is equivalent to computing $\gamma_z(\mathbf{x}_i)$ (called **Expected Sufficient Statistics**)

Details for M-Step

$$Q(\theta; \theta^{(t-1)}) = \sum_{i=1}^n \sum_{z_i=1}^k \gamma_{z_i}(\mathbf{x}_i) \log P(\mathbf{x}_i, z_i \mid \theta)$$

- In M-Step, need to compute

$$\theta^{(t)} = \arg \max_{\theta} Q(\theta; \theta^{(t-1)})$$

- Note similarity to MLE in Gaussian Bayes Classifiers:

$$\theta^* = \arg \max_{\theta} \sum_{i=1}^n \log P(\mathbf{x}_i, z_i \mid \theta)$$

- Thus, each iteration in the M-step is equivalent to training a GBC with weighted data!
- Closed form solution (given in EM pseudo-code)

Convergence of EM Algorithm

- Can prove that the EM Algorithm **monotonically increases the likelihood**

$$\log P(\mathbf{x}_{1:n} \mid \theta^{(t)}) \geq \log P(\mathbf{x}_{1:n} \mid \theta^{(t-1)})$$

- For Gaussian mixtures, EM is guaranteed to converge to a local maximum*
- Quality of solution **highly depends on initialization!** (as in k-Means)
- **Common strategy:** Rerun algorithm multiple times, and use the solution with largest likelihood

Convergence proof

- Goal: Maximize $\theta^* = \arg \max_{\theta} P(\mathbf{x}_{1:n} \mid \theta)$
- Can rewrite: $P(\mathbf{x}_{1:n} \mid \theta) = \frac{P(\mathbf{x}_{1:n}, \mathbf{z}_{1:n} \mid \theta)}{P(\mathbf{z}_{1:n} \mid \mathbf{x}_{1:n}, \theta)}$
- Take logs, and expectation w.r.t. $P(\mathbf{z}_{1:n} \mid \mathbf{x}_{1:n}, \hat{\theta}^{(t-1)})$

$$\begin{aligned} \underbrace{\mathbb{E}_z [\log P(x|\theta) \mid x, \hat{\theta}]}_{\log P(x|\theta)} &= \mathbb{E}_z \left[\log \frac{P(x, z|\theta)}{P(z|x,\theta)} \mid x, \hat{\theta} \right] \\ &= \underbrace{\mathbb{E}_z [\log P(x,z|\theta) \mid x, \hat{\theta}]}_{Q(\theta, \hat{\theta})} - \underbrace{\mathbb{E}_z [\log P(z|x,\theta) \mid x, \hat{\theta}]}_? \end{aligned}$$

Convergence proof

- So far, we have

$$\log \underline{P(\mathbf{x} \mid \theta)} = \underline{Q(\theta, \theta^{(t-1)})} - \underbrace{\mathbb{E}_{\mathbf{z}}[\log P(\mathbf{z} \mid \mathbf{x}, \theta) \mid \mathbf{x}, \theta^{(t-1)}]}_{f(\theta)}$$

- Want to show $\log P(\mathbf{x} \mid \theta^{(t)}) \geq \log P(\mathbf{x} \mid \theta^{(t-1)})$

Suffices to show that

$$(1) \quad Q(\theta^{(t)}, \theta^{(t-1)}) \geq Q(\theta^{(t-1)}, \theta^{(t-1)}) \quad \checkmark \quad \begin{array}{l} \text{since it's a step} \\ \theta^{(t)} = \arg \max_{\theta} Q(\theta, \theta^{(t-1)}) \end{array}$$
$$(2) \quad f(\theta^{(t)}) \leq f(\theta^{(t-1)})$$

Convergence proof

- So far, we have

$$\log P(\mathbf{x} \mid \theta) = Q(\theta, \theta^{(t-1)}) - \mathbb{E}_{\mathbf{z}}[\log P(\mathbf{z} \mid \mathbf{x}, \theta) \mid \mathbf{x}, \theta^{(t-1)}]$$

- Want to show $\log P(\mathbf{x} \mid \theta^{(t)}) \geq \log P(\mathbf{x} \mid \theta^{(t-1)})$
- Compare term by term:
 - EM-Algorithm guarantees (strict inequality unless converged):

$$Q(\theta^{(t)}, \theta^{(t-1)}) \geq Q(\theta^{(t-1)}, \theta^{(t-1)})$$

- Remains to show:

$$\mathbb{E}_{\mathbf{z}}[\log P(\mathbf{z} \mid \mathbf{x}, \theta^{(t-1)}) \mid \mathbf{x}, \theta^{(t-1)}] \geq \mathbb{E}_{\mathbf{z}}[\log P(\mathbf{z} \mid \mathbf{x}, \theta^{(t)}) \mid \mathbf{x}, \theta^{(t-1)}]$$