Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Data Mining
Learning from Large Data Sets

Lecture 9 – Probabilistic clustering on large data sets

263-5200-00L

Andreas Krause

Course organization

- **Retrieval**
  - Given a query, find "most similar" item in a large data set
  - Determine relevance of search results
  - *Applications*: GoogleGoggles, Shazam, …
- **Supervised learning** (Classification, Regression)
  - Learn a concept (function mapping queries to labels)
  - *Applications*: Spam filtering, predicting price changes, …
- **Unsupervised learning** (Clustering, dimension reduction)
  - Identify clusters, "common patterns"; anomaly detection
  - *Applications*: Recommender systems, fraud detection, …
- **Learning with limited feedback**
  - Learn to optimize a function that's expensive to evaluate
  - *Applications*: Online advertising, opt. UI, learning rankings, …

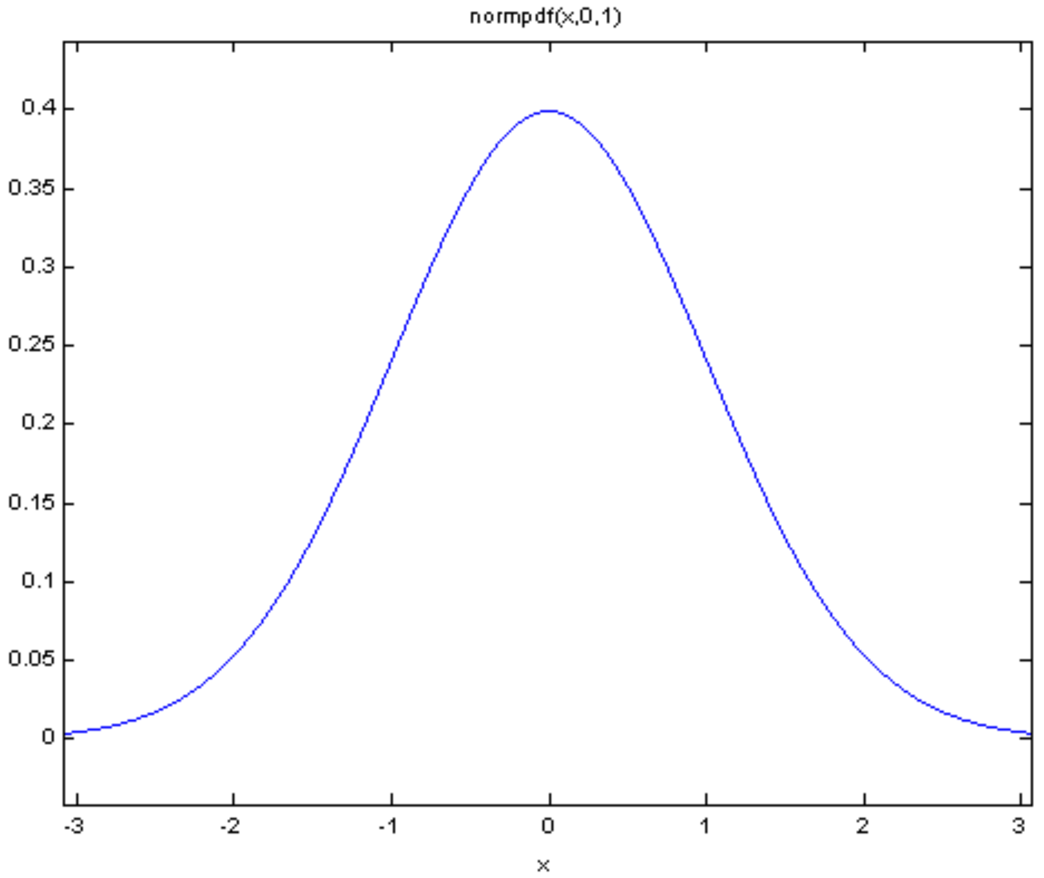
Today we will

- Clustering large data sets with probabilistic mixture models
- Discuss why probabilistic clustering is useful
- Briefly review the EM algorithm
- See analogues of online k-means and data set summarization (coresets)
- See some applications of classification and anomaly detection

Summary from last lecture

	Geometric (k-means)	**Probabilistic (GMM)**	
	Simple interpretation	More flexible; "confidence" (e.g. for anomaly detection, …)	
Batch	Classic k-means	EM	Slow
Online	Online k-means	???	Very fast but not flexible / robust
Compression	Coresets	???	Fast and accurate

Example: Gaussian distribution

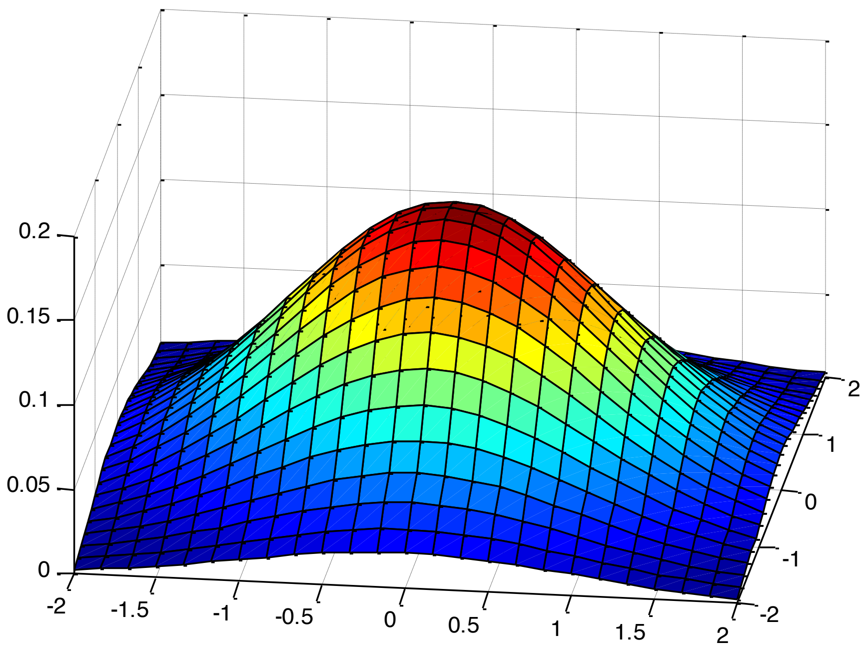


- σ = Standard deviation
- μ = mean

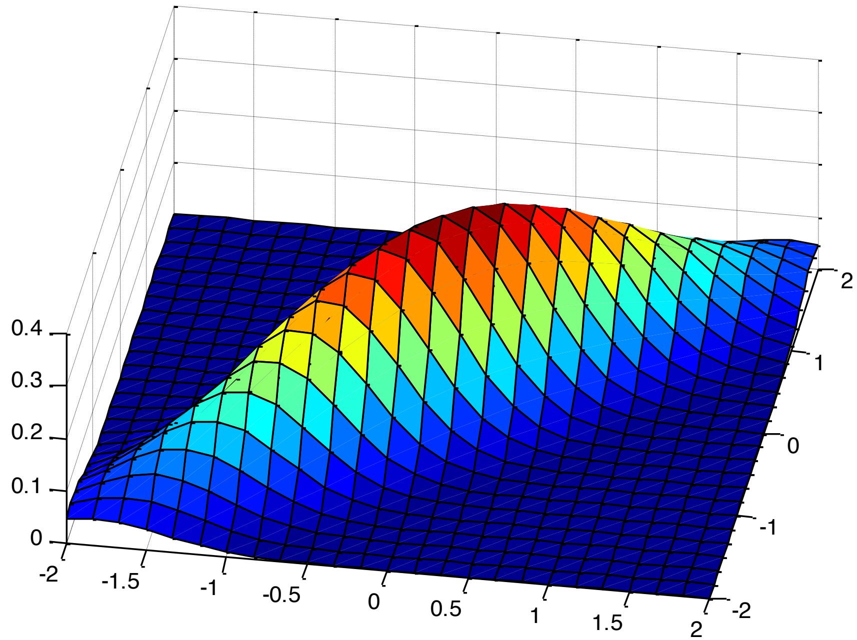
$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Multivariate Gaussian distribution

$$N(y;\mu,\Sigma)=\frac{1}{2\pi\sqrt{|\Sigma|}}\exp\left(-\frac{1}{2}(y-\mu)^T\Sigma^{-1}(y-\mu)\right)$$



$$\Sigma=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$$



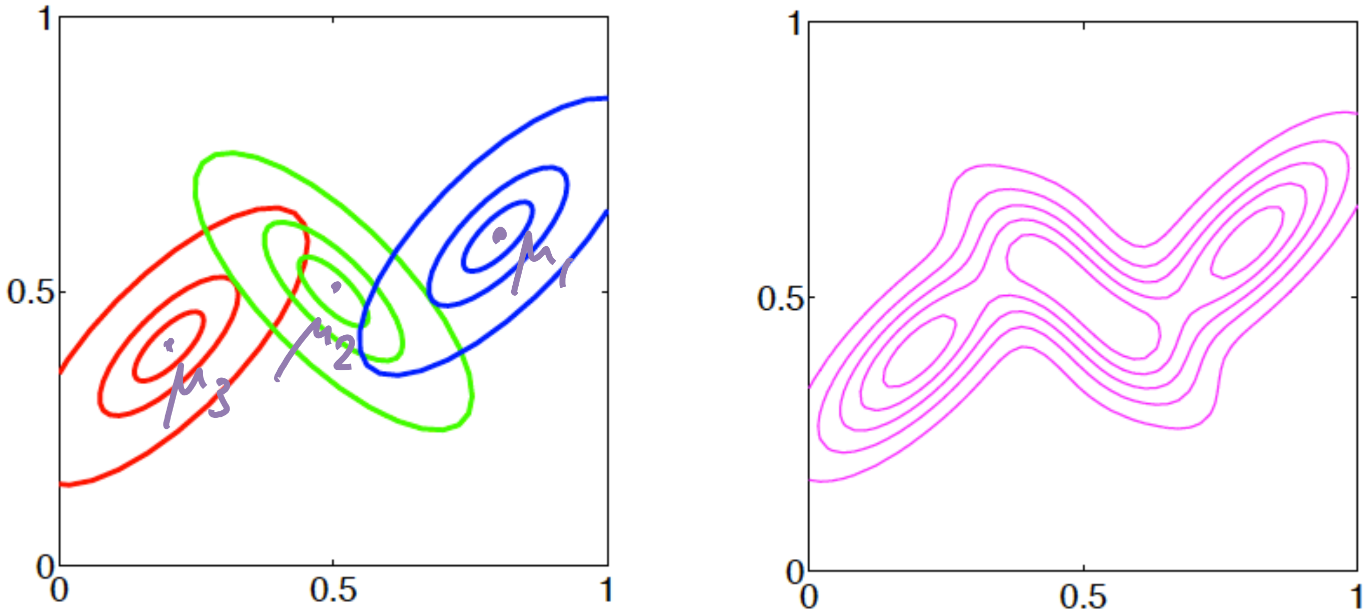
$$\Sigma=\begin{pmatrix}1 & 0.9\\ 0.9 & 1\end{pmatrix}$$

Gaussian mixtures

- Convex-combination of Gaussian distributions

$$P(\mathbf{x} \mid \mu, \Sigma) = \sum_i w_i \mathcal{N}(\mathbf{x}; \mu_i, \Sigma_i)$$

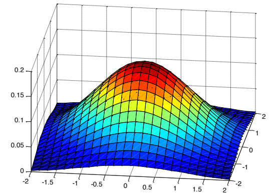
where $w_i \geq 0$ and $\sum_i w_i = 1$



Mixture modeling

- Model each cluster as a probability distribution

$$P(\mathbf{x} \mid \theta_j)$$



e.g. $[\mu_j, \Sigma_j]$

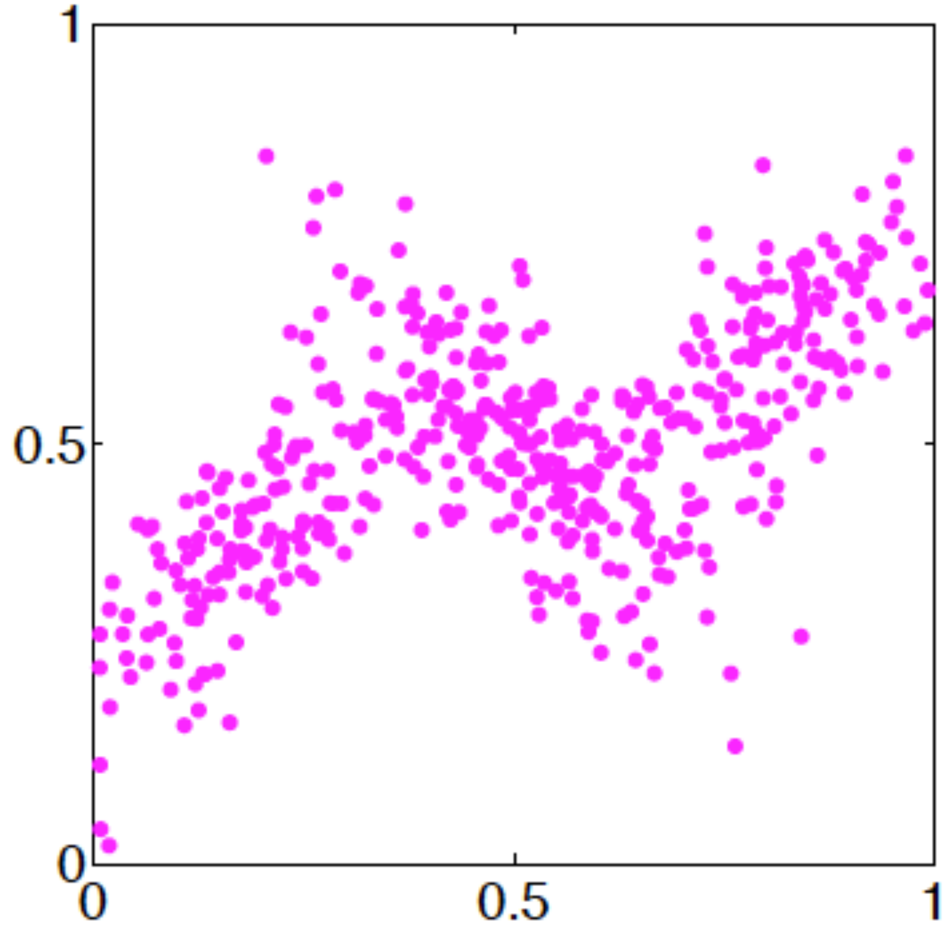
- Assuming data is sampled i.i.d., likelihood of data is

$$P(D \mid \theta) = \prod_i \sum_j w_j P(\mathbf{x_i} \mid \theta_j)$$

- Choose parameters to minimize negative log likelihood

$$L(D; \theta) = -\sum_i \log \sum_j w_j P(\mathbf{x_i} \mid \theta_j)$$

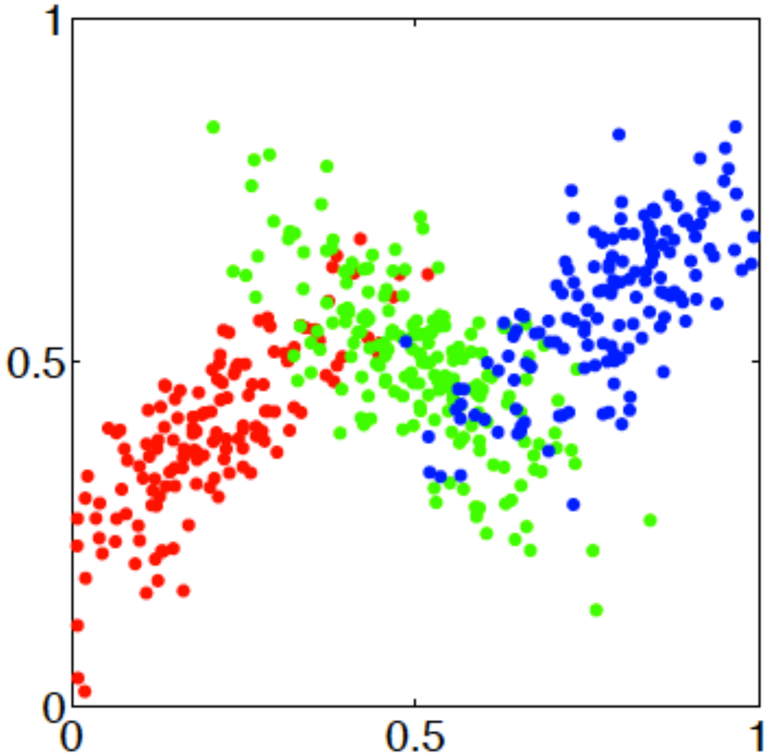
Clustering = Fitting a mixture model



$$(\mu^*, \Sigma^*, w^*) = \arg\min -\sum_i \log \sum_{j=1}^{k} w_j \mathcal{N}(\mathbf{x_i} \mid \mu_j, \Sigma_j)$$

Sampling from a Gaussian mixture

- To sample a data point i
  - Sample component indicator z_i so that $P(z_i = j) = w_j$
  - Then sample $\mathbf{x}_i$ from $\mathcal{N}(\mathbf{x}_i \mid \mu_{z_i}, \Sigma_{z_i})$

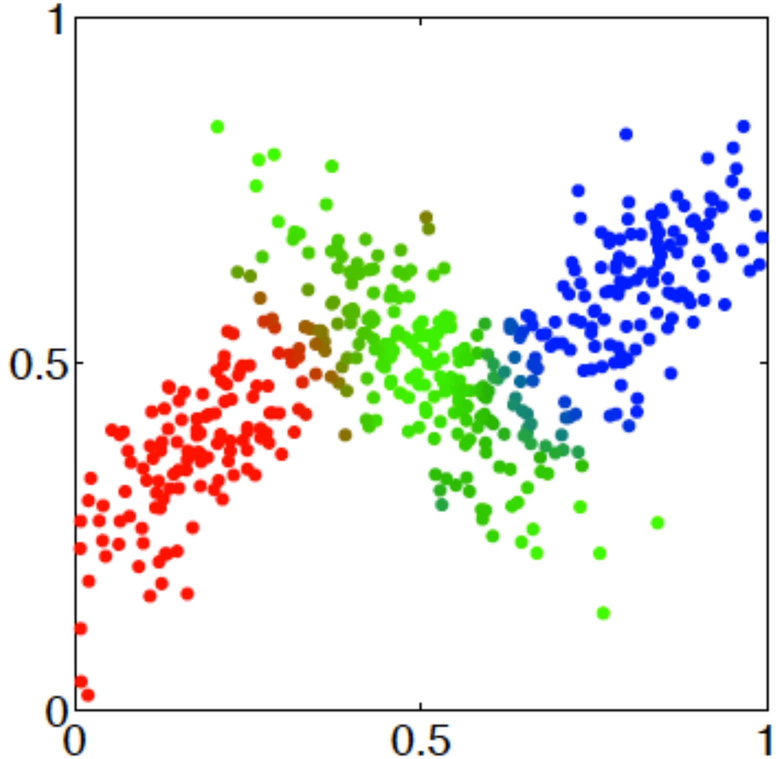


Posterior probabilities

- Suppose we're given a model $P(z|\theta)$ $P(x|z,\theta)$
- Then, for each data point, we can compute a posterior distribution over cluster membership
- This means inferring latent (hidden) variables z

$$\gamma_j(x) = P(z = j \mid \mathbf{x}, \Sigma, \mu)$$

$$= \frac{w_j P(\mathbf{x} \mid \Sigma_j, \mu_j)}{\sum_\ell w_\ell P(\mathbf{x} \mid \Sigma_\ell, \mu_\ell)}$$



Maximum likelihood estimation

- At MLE

$$(\mu^*, \Sigma^*, w^*) = \arg\min - \sum_i \log \sum_{j=1}^k w_j \mathcal{N}(\mathbf{x_i} \mid \mu_j, \Sigma_j)$$

it must hold that

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

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

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

These equations are coupled → difficult to solve jointly

Alternating optimization: EM

- While not converged
  - E-step: calculate cluster membership weights ("Expected sufficient statistics") for each point:

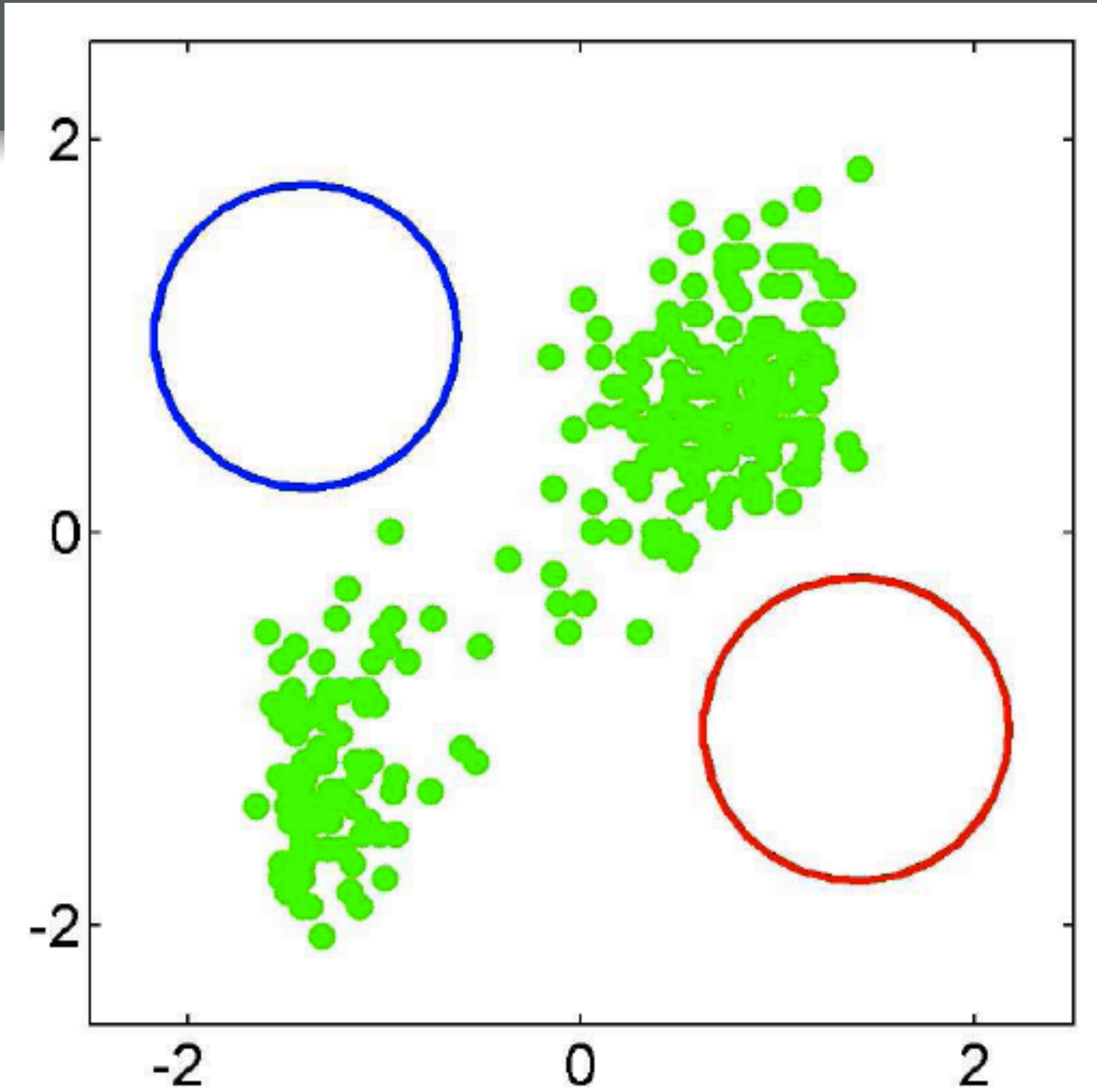
    Calculate $\gamma_j(\mathbf{x}_i)$ for each i and j given estimates of μ, Σ, w from previous iteration

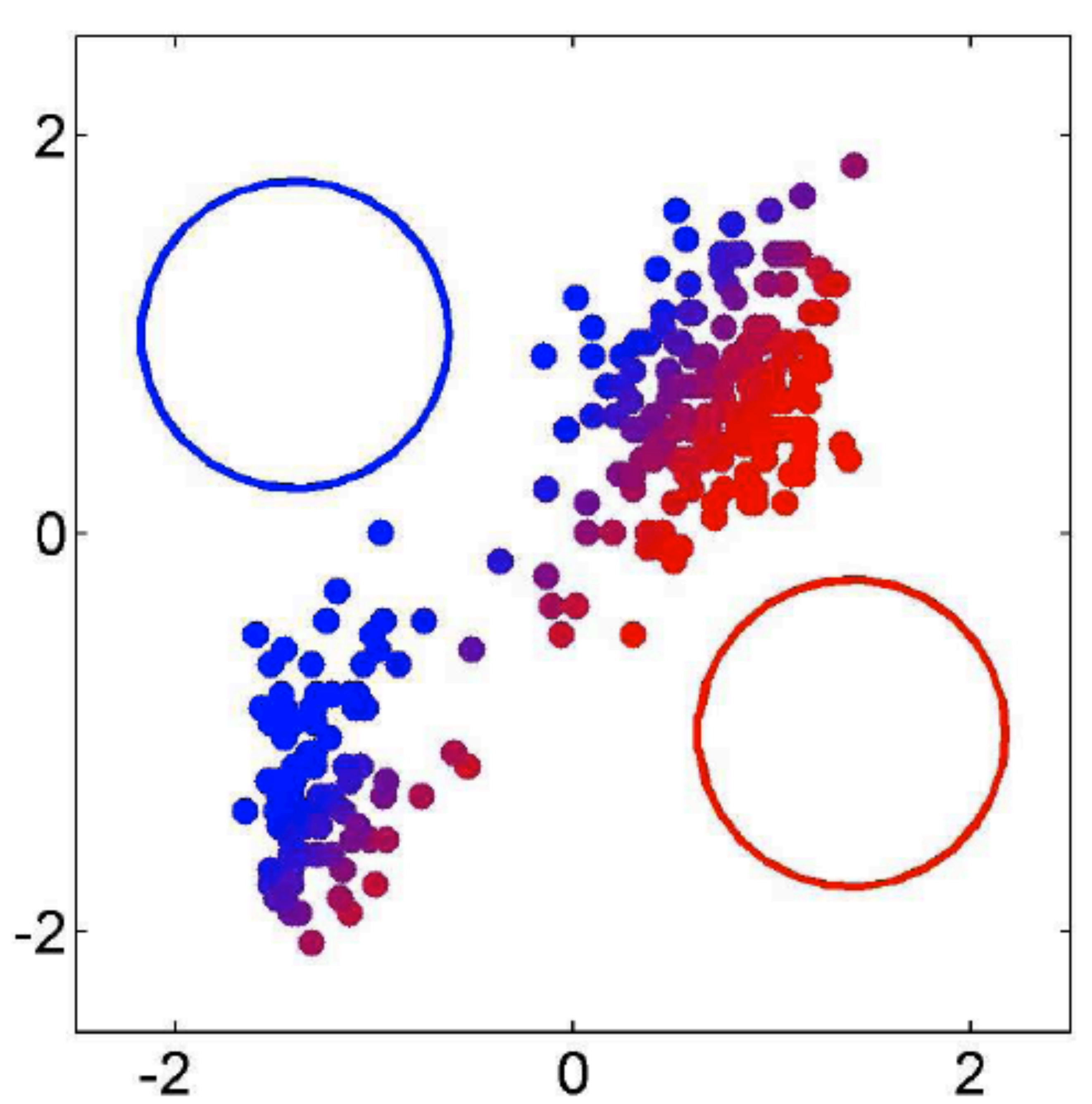
  - M-step: Fit clusters to weighted data points (closed form Maximum likelihood solution!)

    Compute μ, Σ, w given $\gamma_j(\mathbf{x}_i)$

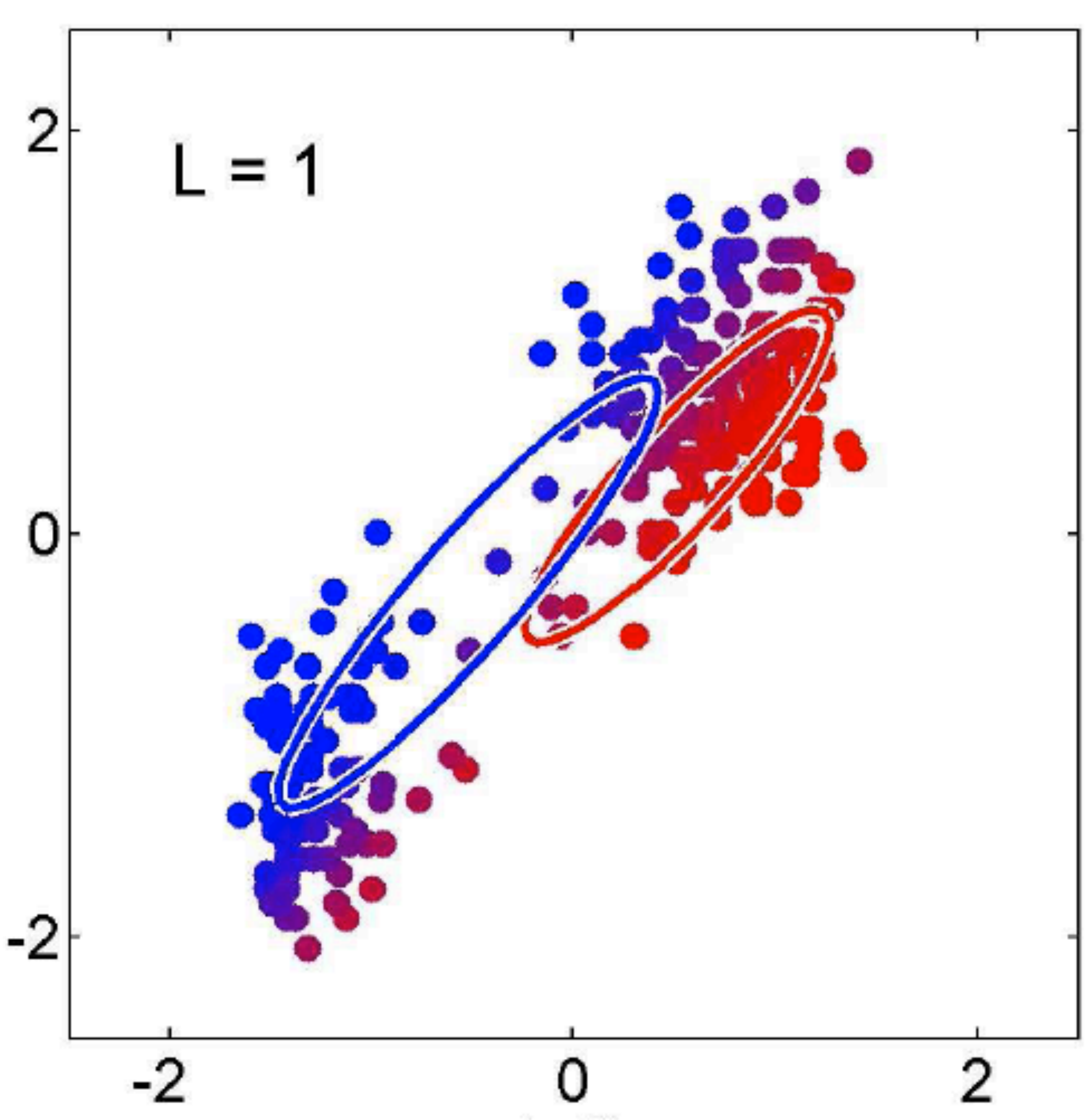
    e.g.,

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

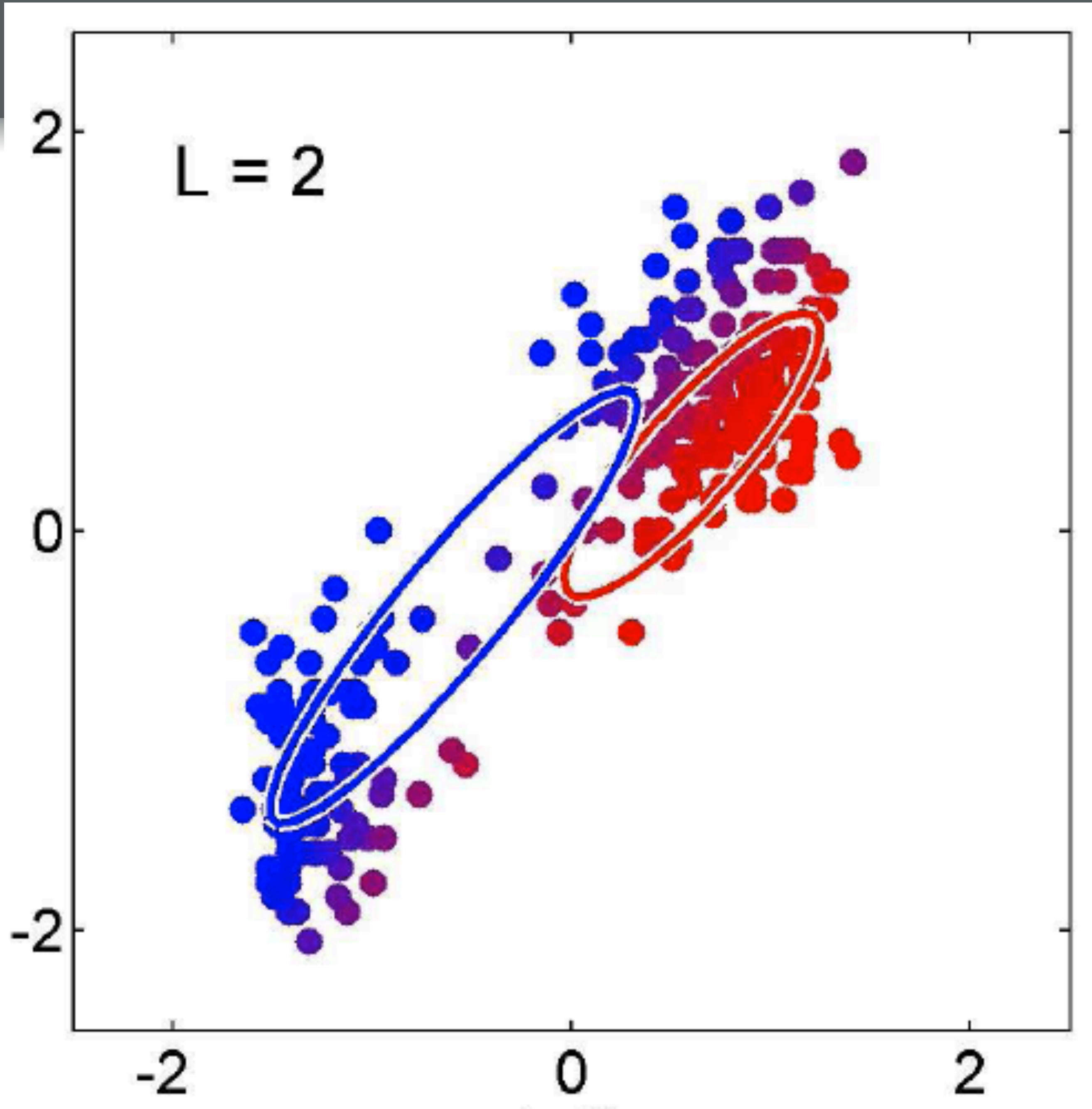

2
0
-2
-2
0
2



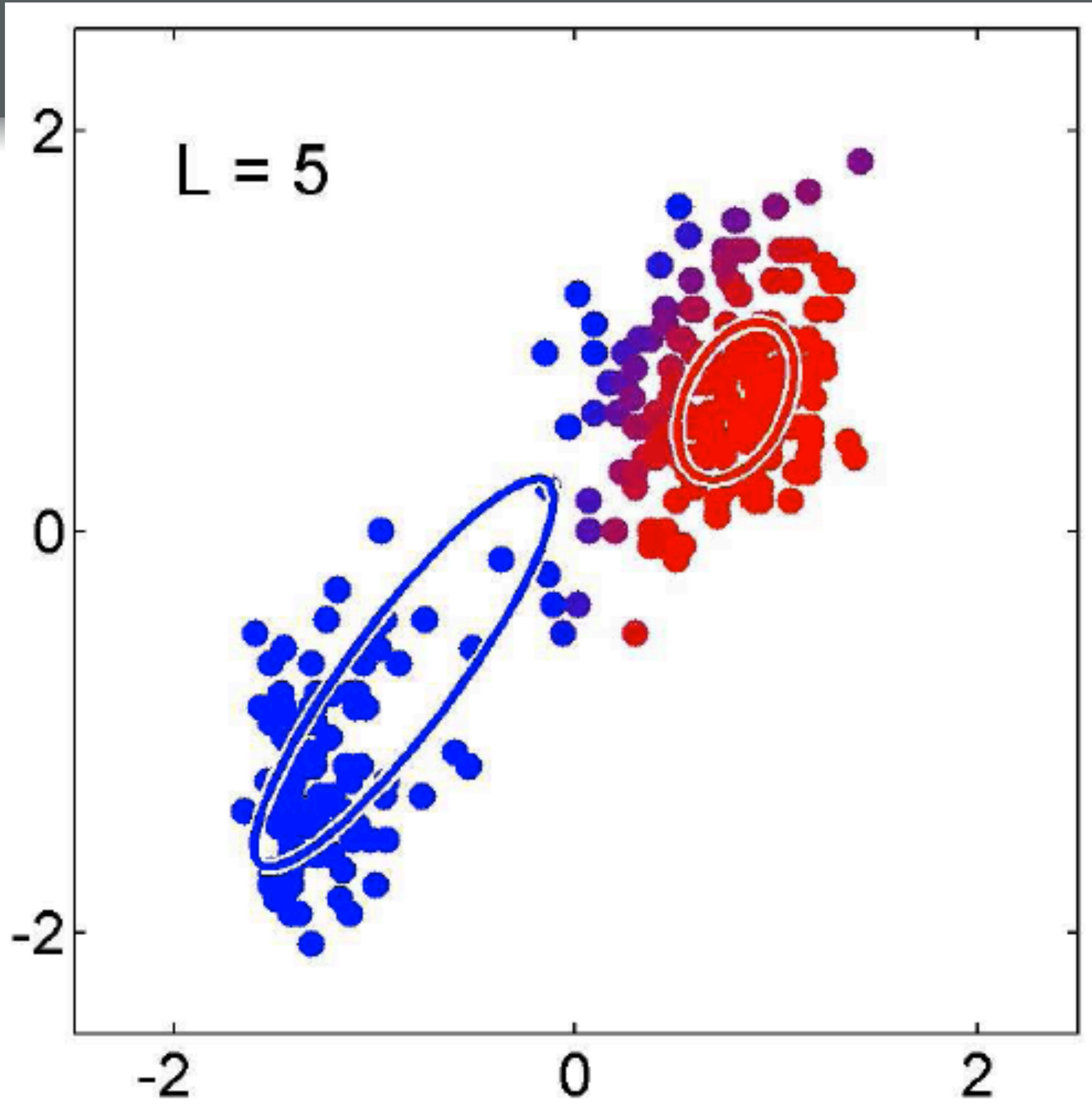
2
0
-2
-2
0
2

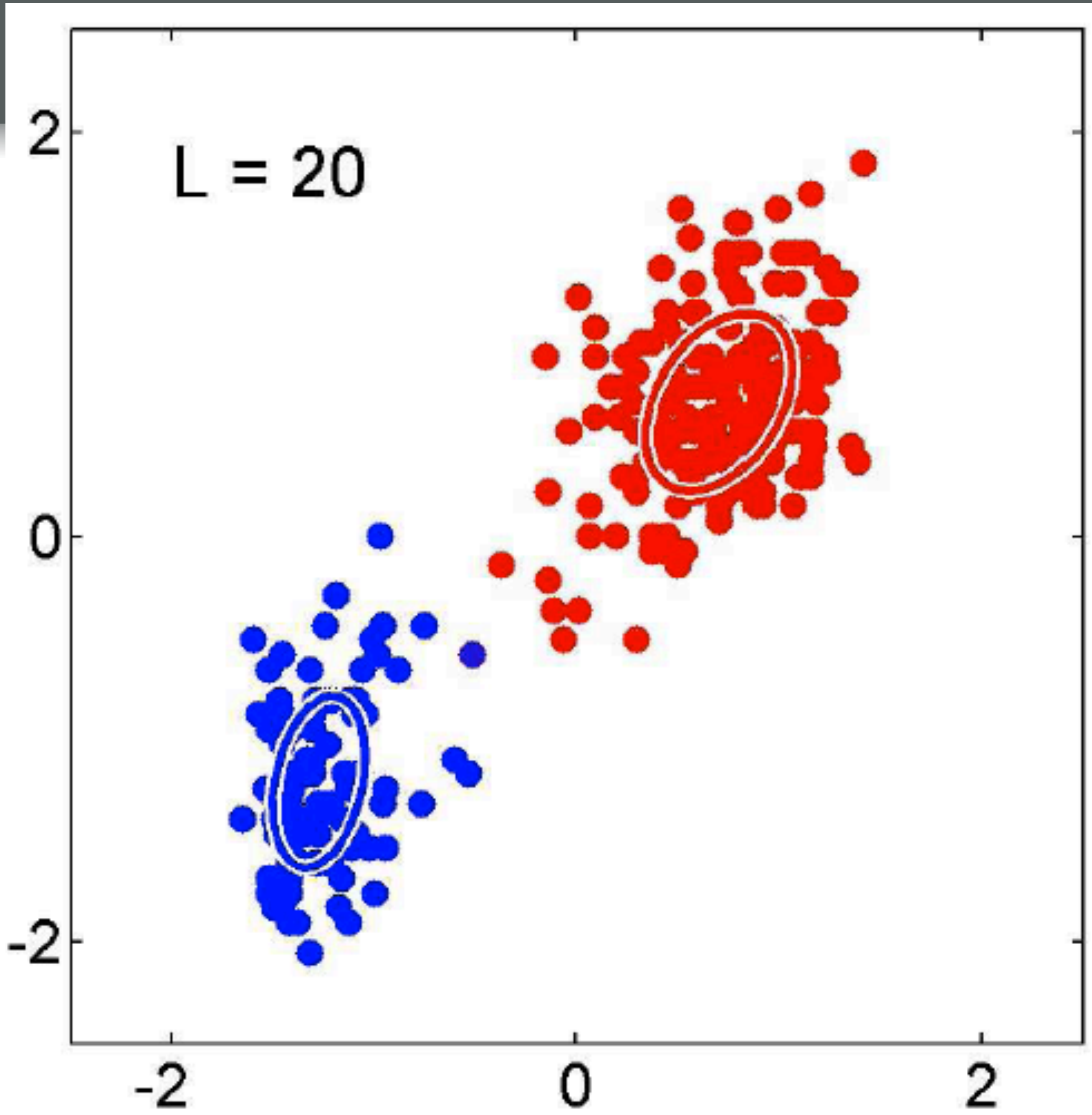


L = 1
2
0
-2
-2
0
2



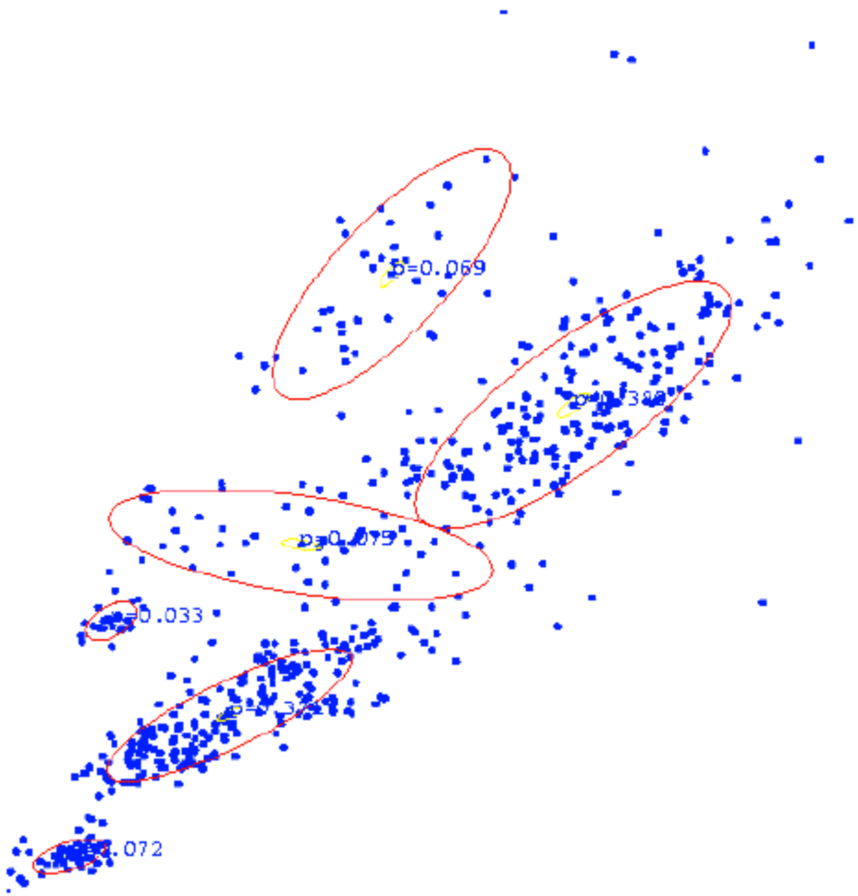
L = 2
2
0
-2
-2
0
2

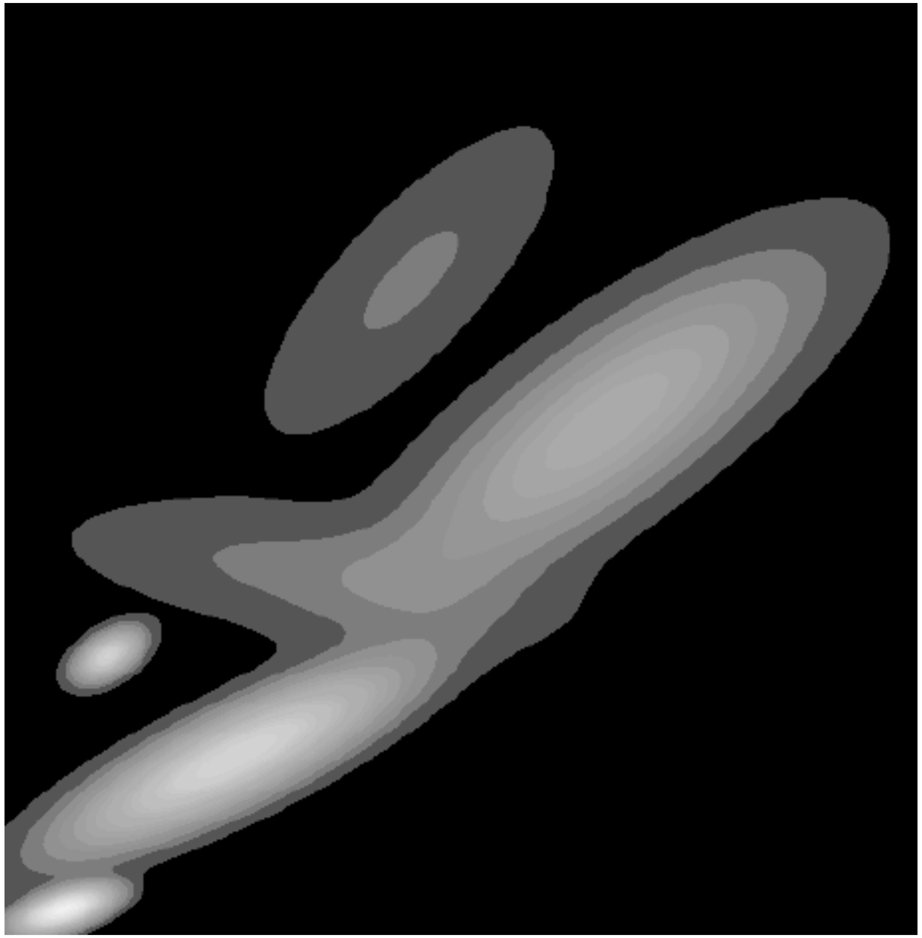

L = 5
2
0
-2
-2
0
2


L = 20
2
0
-2
-2
0
2

Example fit on Bio Assay data

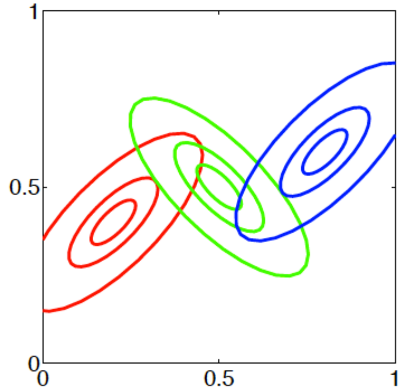
[Andrew Moore]





Why are mixture models useful?

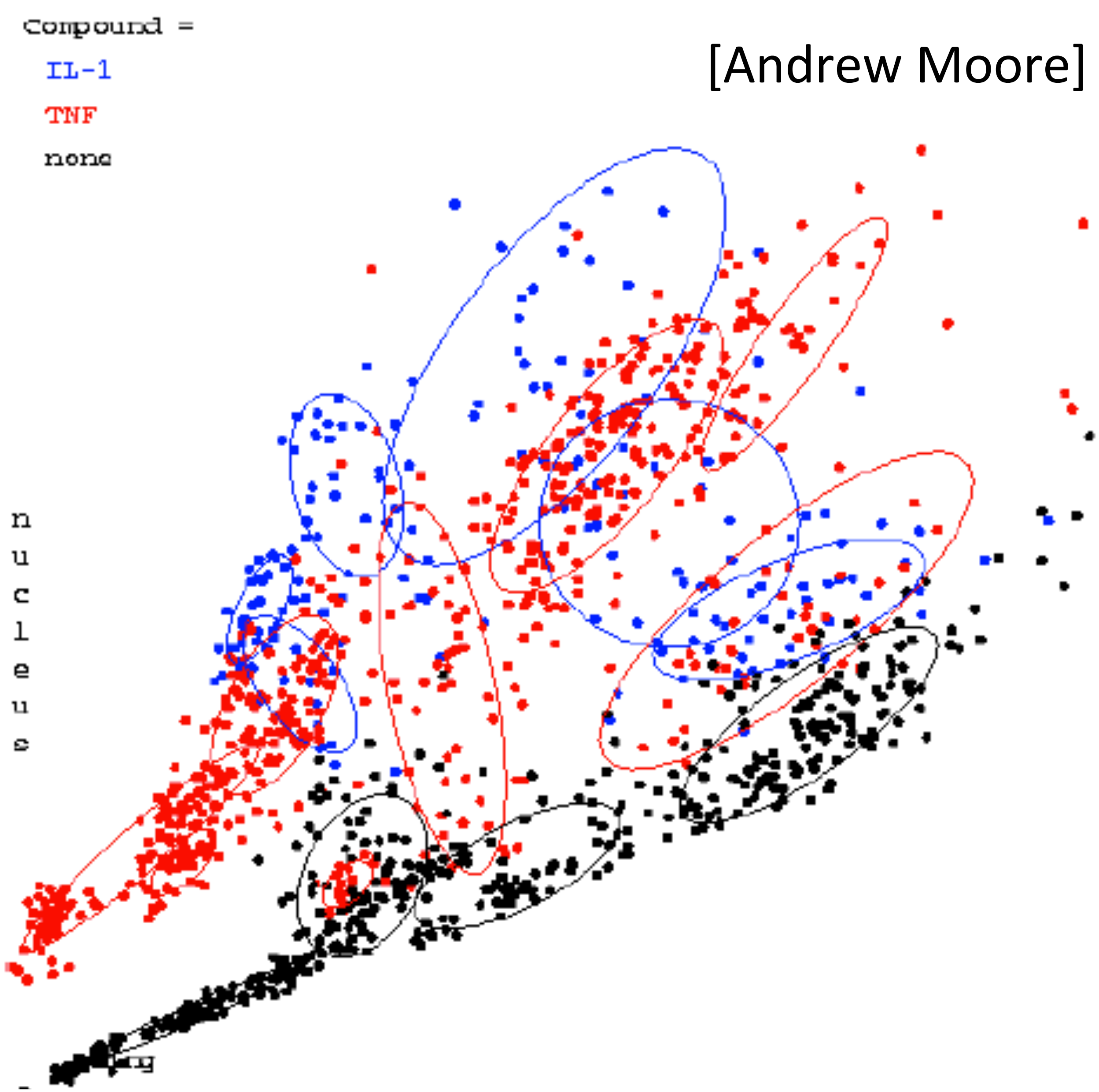
- Can encode assumptions about "shape" of clusters
  - E.g., fit ellipses instead of points



- Can be part of more complex statistical models
  - E.g., classifiers (or more generally *graphical models*)
- Probabilistic models can output likelihood P(x) of a point x
  - Useful for anomaly detection

Clustering for (nonlinear) classification

[Andrew Moore]



Gaussian-Bayes classifiers

- Given *labeled* data set $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
  - Label $y_i \in \{1, \ldots m\}$
  - 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

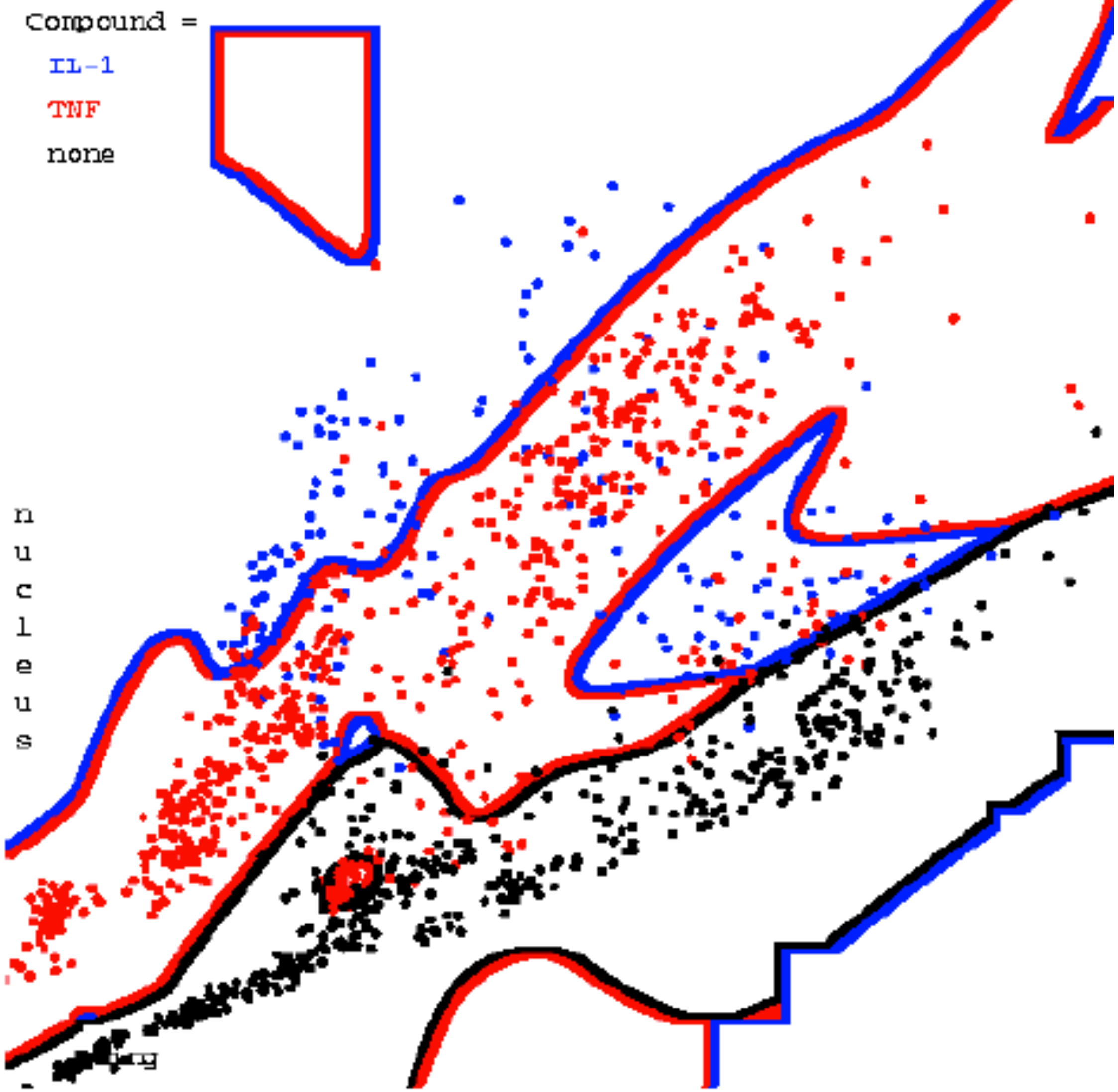
- How do we use this model for classification?

$$P(y|x) = \frac{P(y) \cdot P(x|y)}{\sum_{y'} P(y') P(x|y')} = \frac{1}{Z} P(y) \cdot P(x|y)$$

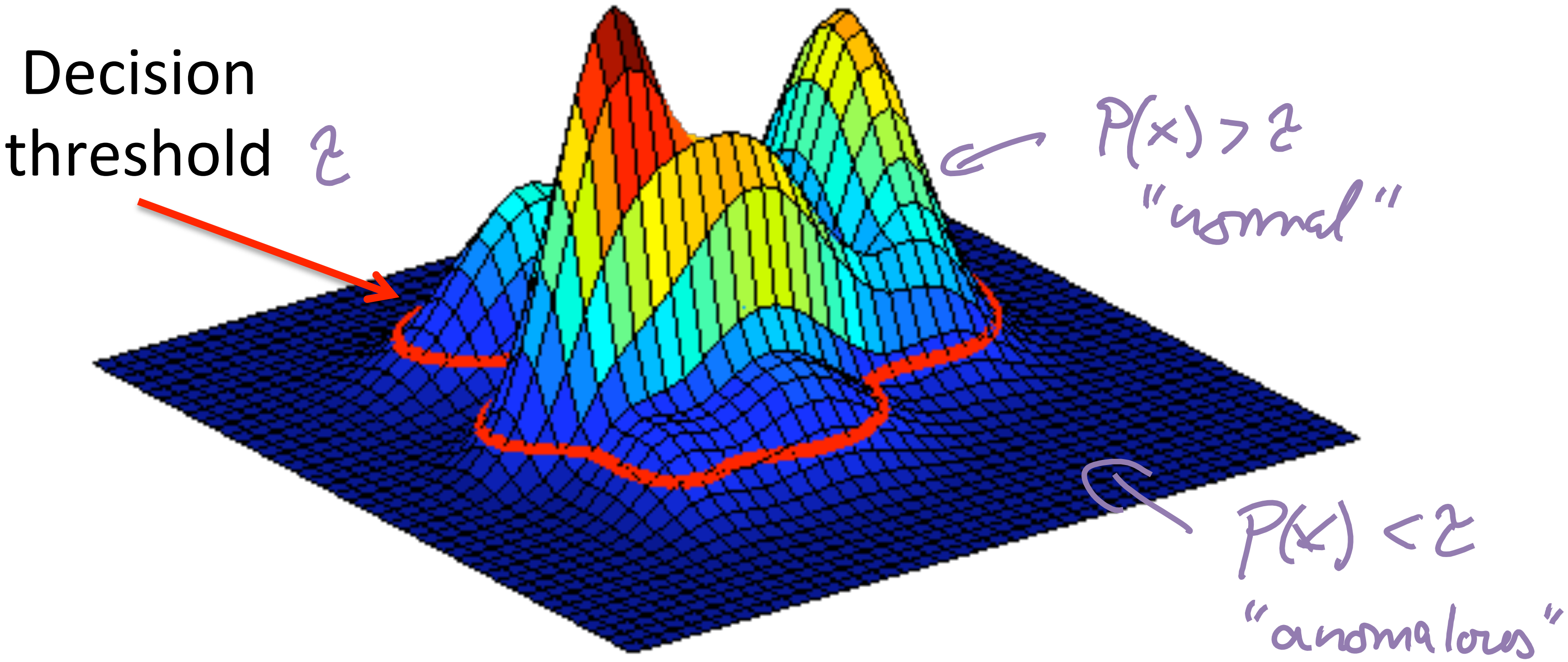
Classify acc. to $\arg\max_y P(y|x) = \arg\max_y P(y) P(x|y)$

Resulting classifier

[Andrew Moore]



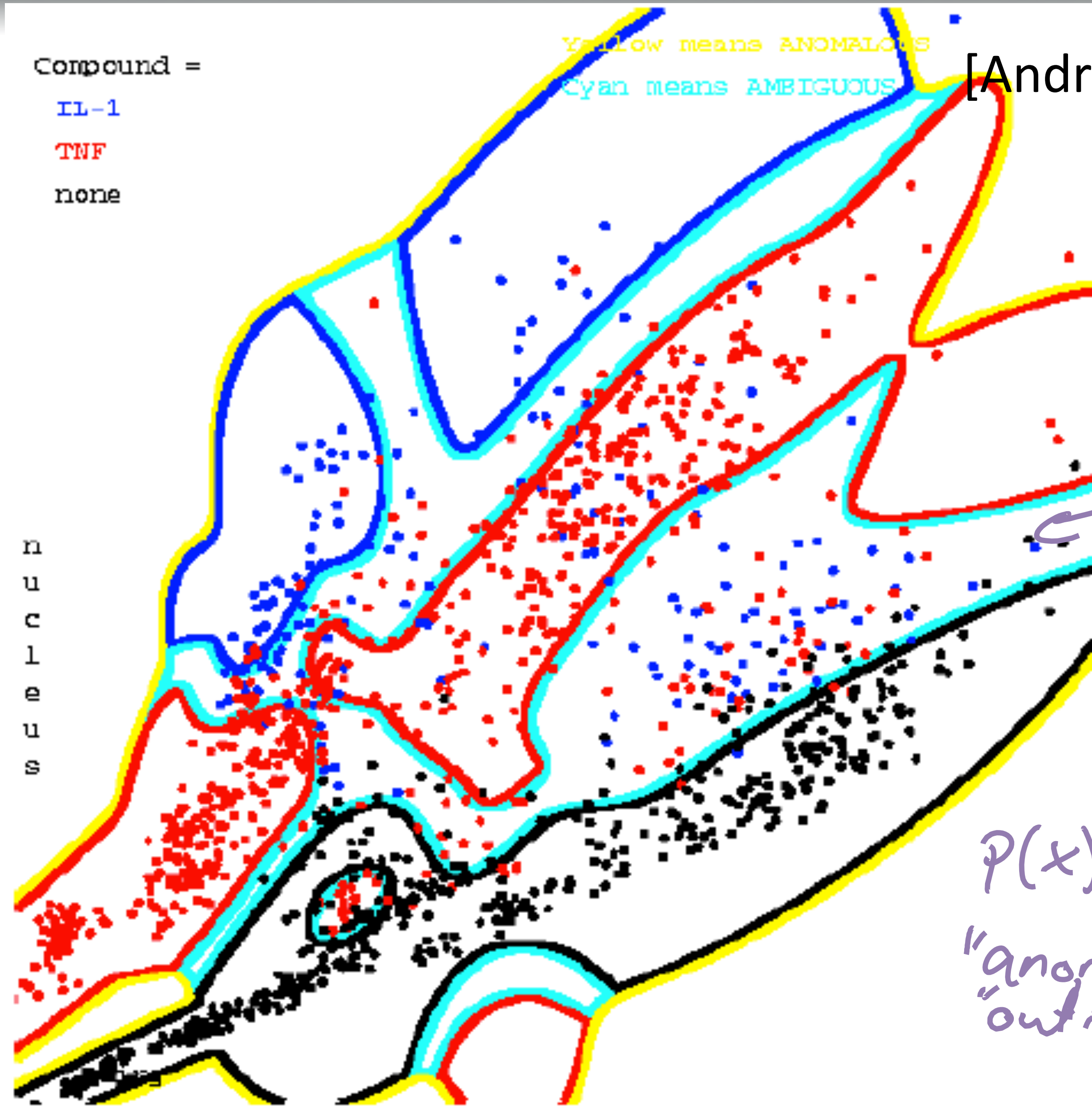
Anomaly detection with mixture models



- Can classify data points according to estimated probability density

Anomaly detection

[Andrew Moore]



ambiguous

$P(x) < \varepsilon$

"anomalous" / "outlier"

Probabilistic clustering for large data sets

- EM has similar drawbacks as k-means for large data sets
  - Need to make one pass through the entire data set per iteration

- Can we use similar tricks as for k-means to scale to large data sets?
  - Online optimization?
  - Compressed representation?

EM once again:

- While not converged
  - E-step: calculate cluster membership weights ("Expected sufficient statistics") for each point:

    Calculate $\gamma_j(\mathbf{x}_i)$ for each i and j given estimates of μ, Σ, w from previous iteration

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

    Compute μ, Σ, w given $\gamma_j(\mathbf{x}_i)$

    e.g.,

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

Another way to look at EM

- Initialize t=0, $\mu^{(0)}, \Sigma^{(0)}, w^{(0)}$
- While not converged
  - Reset: $\hat{\mu}_j = 0, \hat{\Sigma}_j = 0, \hat{w}_j = 0$
  - For each example i and component j do

    compute $\gamma_j(\mathbf{x}_i) = \gamma_j(\mathbf{x}_i \mid \mu^{(t)}, \Sigma^{(t)}, w^{(t)})$

    compute
$$\hat{\mu}_j \leftarrow \hat{\mu}_j + \gamma_j(\mathbf{x}_i)\mathbf{x}_i$$
$$\hat{\Sigma}_j \leftarrow \hat{\Sigma}_j + \gamma_j(\mathbf{x}_i)\mathbf{x}_i\mathbf{x}_i^T$$
$$\hat{w}_j \leftarrow \hat{w}_j + \gamma_j(\mathbf{x}_i)$$

  Set t=t+1, and

$$\mu_j^{(t)} = \hat{\mu}_j / \hat{w}_j \quad \Sigma_j^{(t)} = \hat{\Sigma}_j / \hat{w}_j \quad w_j^{(t)} = \hat{w}_j / N$$

Can we make EM incremental?

- *Idea*: Update estimates of μ, Σ, w after each example
- Similar as online k-means

Stepwise EM

- Initialize t=0, $\mu^{(0)}, \Sigma^{(0)}, w^{(0)}$
- While not converged
  - For each example $\mathbf{x}_t$ and component j do

compute $\gamma_j(\mathbf{x}_t) = \gamma_j(\mathbf{x}_t \mid \mu^{(t)}, \Sigma^{(t)}, w^{(t)})$

compute
$$\hat{\mu}_j \leftarrow \hat{\mu}_j + \eta_t \gamma_j(\mathbf{x}_t)(\mathbf{x}_t - \hat{\mu}_j)$$
$$\hat{\Sigma}_j \leftarrow \hat{\Sigma}_j + \eta_t \gamma_j(\mathbf{x}_t)(\mathbf{x}_t \mathbf{x}_t^T - \hat{\Sigma}_j)$$
$$\hat{w}_j \leftarrow \hat{w}_j + \eta_t(\gamma_j(\mathbf{x}_t) - \hat{w}_j)$$

Set t=t+1, and $\mu_j^{(t)} = \hat{\mu}_j / \hat{w}_j \;\; \Sigma_j^{(t)} = \hat{\Sigma}_j / \hat{w}_j \;\; w_j^{(t)} = w_j$

[see also Sato & Ishii '00, Liang & Klein '09]

Stepwise EM more generally

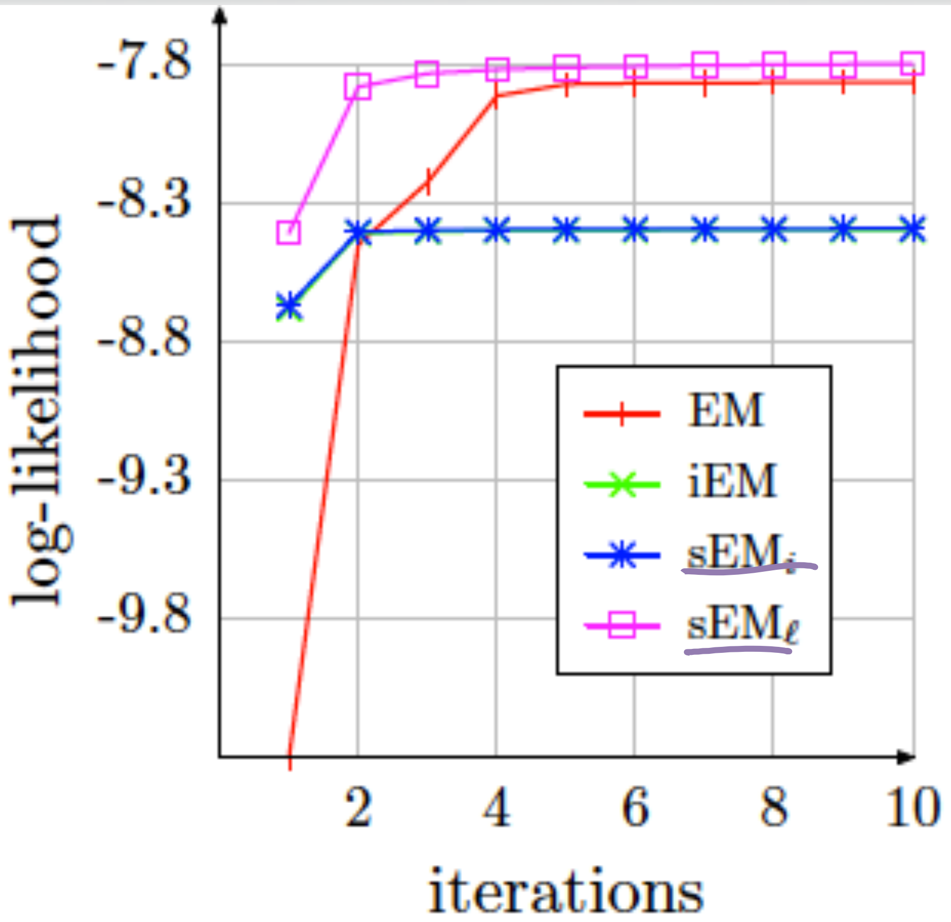
Stepwise EM (sEM)

$\mu \leftarrow$ initialization; $k = 0$
for each iteration $t = 1, \ldots, T$:
  for each example $i = 1, \ldots, n$ in random order:
    $s'_i \leftarrow \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{x}^{(i)}; \theta(\mu))\, \phi(\mathbf{x}^{(i)}, \mathbf{z})$ [inference]
    $\mu \leftarrow (1-\eta_k)\mu + \eta_k s'_i; k \leftarrow k+1$ [towards new]

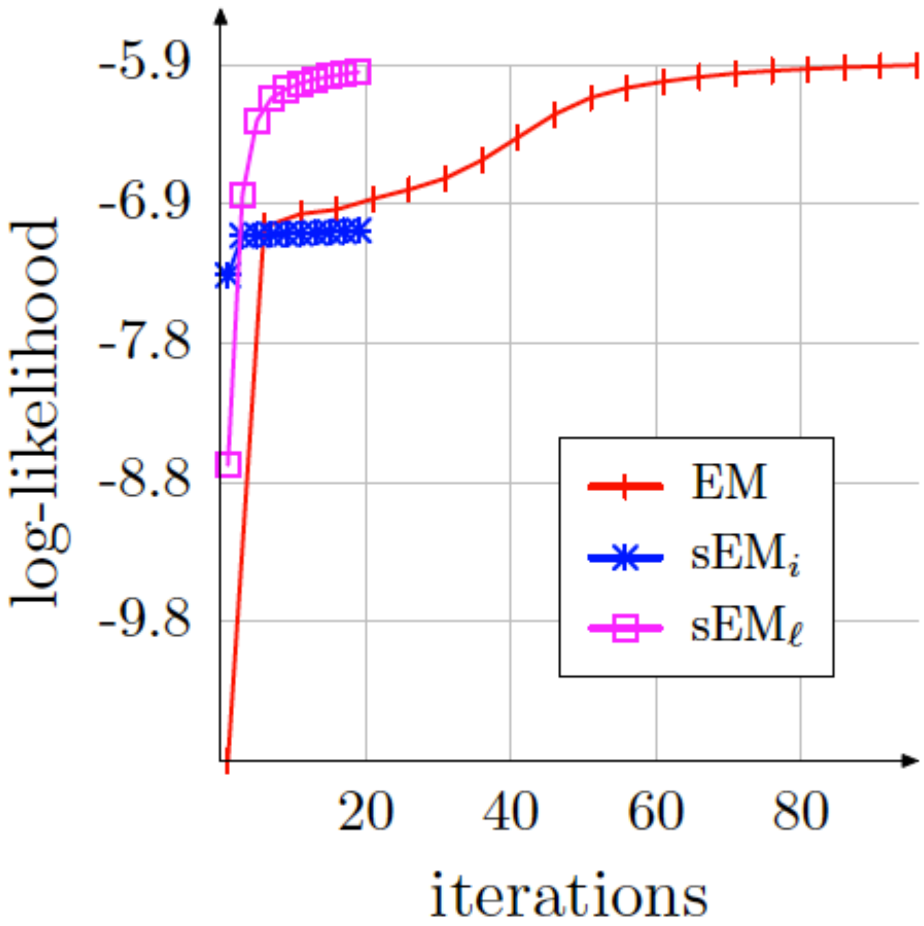
- Works for other latent variable models as well (e.g., HMMs, …)
- Instead of updating parameters after each example, often works better when using “mini-batches”

[Liang & Klein ‘09]

Performance of online EM



Document clustering



POS Tagging

[Liang & Klein '09]

Summary so far

	Geometric (k-means)	**Probabilistic (GMM)**	
Batch	Classic K-means	EM	Slow
Online	Online k-means	Online (stepwise) EM	Very fast but not flexible / robust
Compression	Coresets	???	Fast and accurate
	Simple interpretation	More flexible; “confidence” (e.g. for anomaly detection;	

A Geometric Perspective

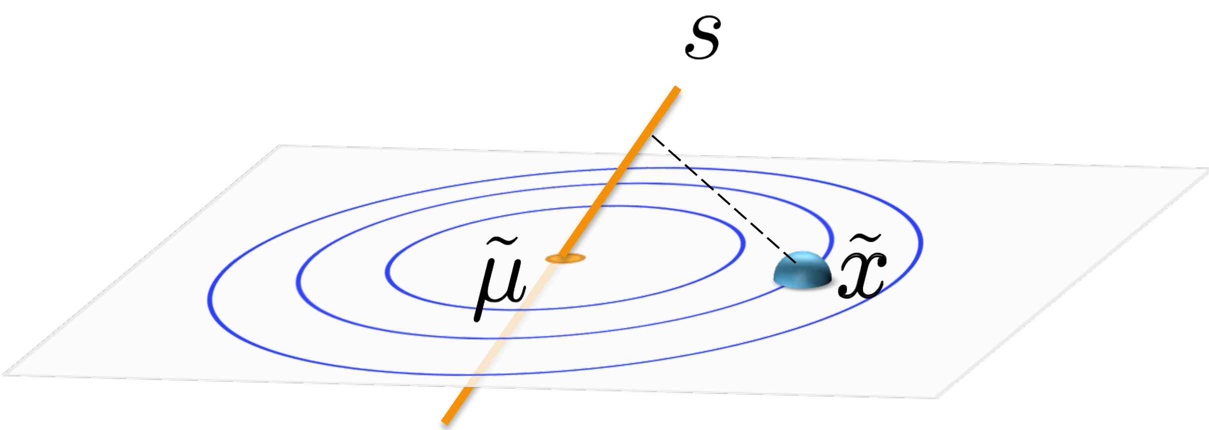
Gaussian level sets can be expressed purely geometrically:

$$\mathcal{N}(x;\mu,\Sigma) = \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

$$= \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-W\,\mathrm{dist}(\tilde{x},\mathbf{s})^2\right)$$

affine subspace

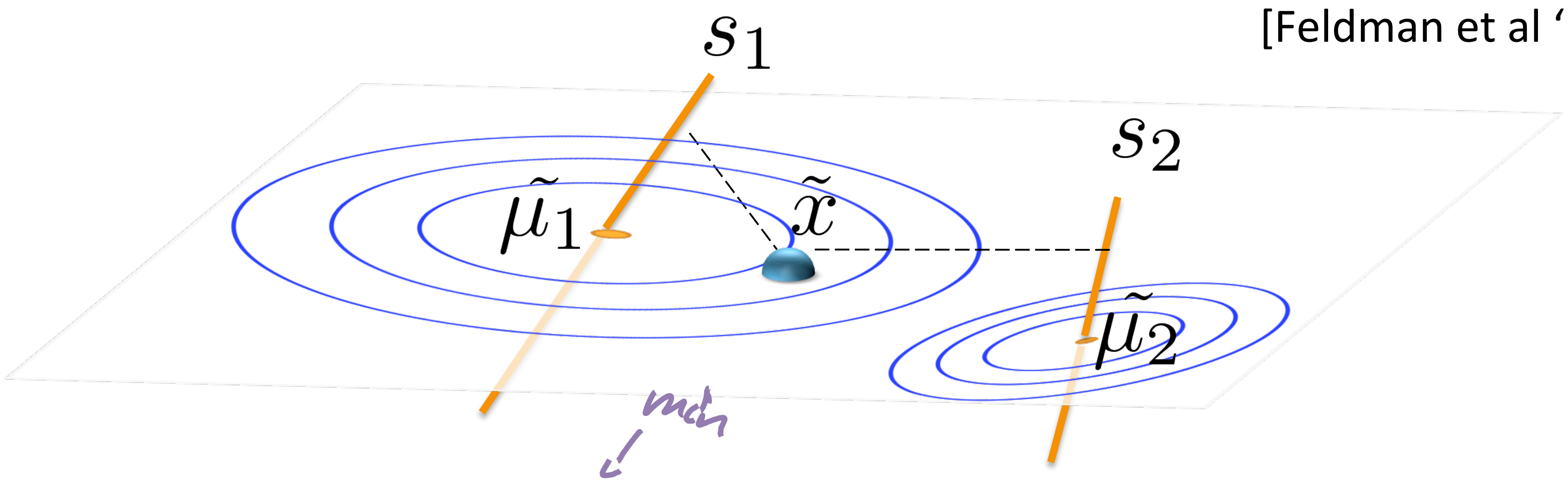
$\mathbf{s} = \mathbf{s}(\mu,\Sigma) \subset \mathbb{R}^{2d}$



$\tilde{x}, \tilde{\mu} \in \mathbb{R}^{2d}$

Geometric Reduction

[Feldman et al '11]



$$\ln P(x|\theta) \propto -\ln \sum_{i=1}^{k} w_i' \exp\left(-W_i \text{dist}(\tilde{x}, \mathbf{s_i})^2\right)$$

Soft-min

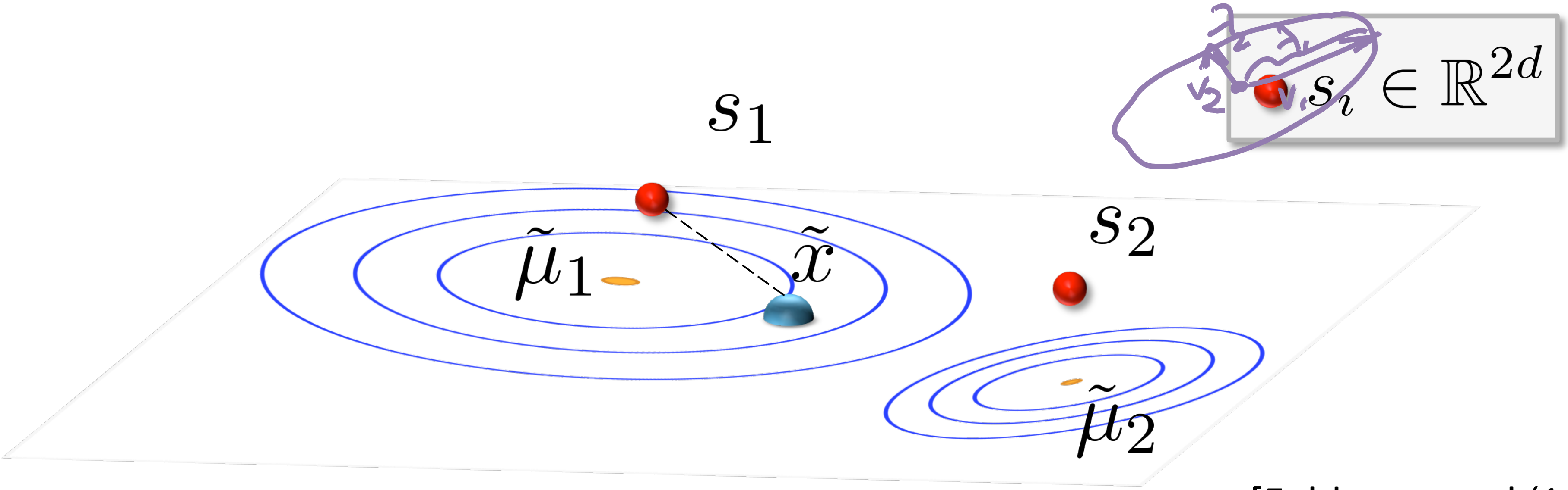
$\ln P(x|\theta) \geq \min_i W_i \text{dist}(\tilde{x}, \mathbf{s_i})$ Projective Clustering!

Bound using generalized $\triangle$-inequality

➔ **Can apply geometric coreset tools to mixture models**

Semi-Spherical Gaussian Mixtures

Subspaces $\mathbf{s_i}$ can be chosen as points for Semi-spherical GMMs (covariance eigenvalues $\lambda_{\text{min}} \leq \lambda_i \leq \lambda_{\text{max}}$)



[Feldman et al '11]

Thm. An ϵ-coreset for k-means in the transformed space gives a $(k, \epsilon\lambda^2_{max}/\lambda^2_{min})$-coreset for semi-spherical GMMs

$$(1 - \epsilon\frac{\lambda^2_{\text{max}}}{\lambda^2_{\text{min}}})\mathcal{L}(\theta|D) \leq \mathcal{L}(\theta|C) \leq (1 + \epsilon\frac{\lambda^2_{\text{max}}}{\lambda^2_{\text{min}}})\mathcal{L}(\theta|D) \text{ w.h.p}$$

Coresets via Adaptive Sampling

[Feldman et al '11]

$B \leftarrow \emptyset \quad D' \leftarrow D$

while $D' \neq \emptyset$

$\quad S \leftarrow$ uniformly sample $10dk\ln(\frac{1}{\epsilon})$ points from D'

$\quad$ Remove $\frac{|D'|}{2}$ points nearest to S from D'

$\quad B \leftarrow B \cup S$

Partition D into Voronoi cells D_b centered at $b \in B$

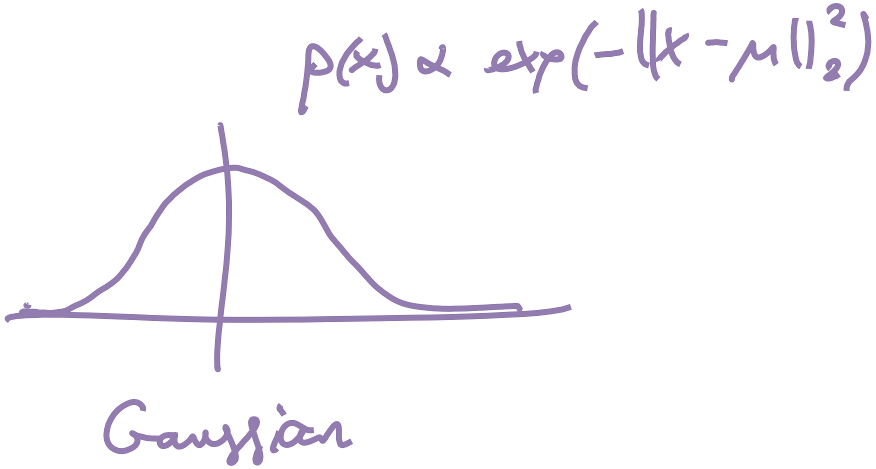
$$q(x) \propto \lceil \frac{5}{|D_b|} + \frac{\text{dist}(x,B)^2}{\sum_{x'} \text{dist}(x',B)^2} \rceil, \qquad \gamma(x) = \frac{1}{|C|q(x)}$$

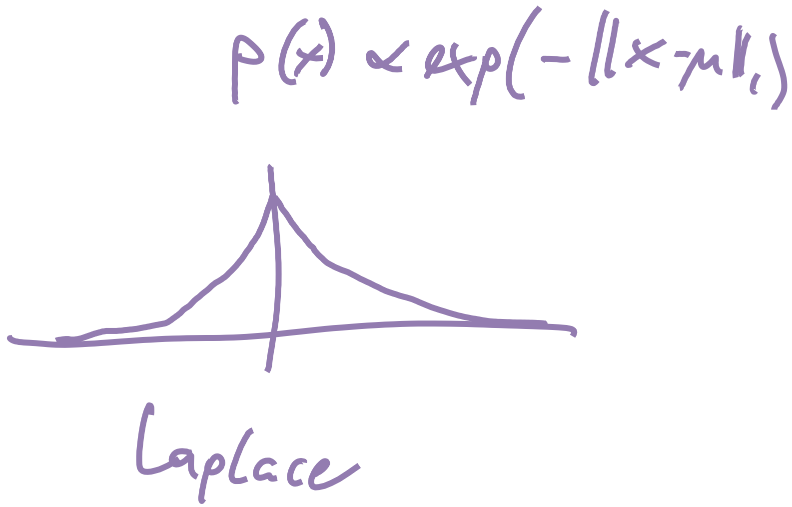
$C \leftarrow$ sample $10\lceil dk \log^2 n \log(\frac{1}{\delta})/\epsilon^2 \rceil$ from D via q

Thm. (C, γ) is a (k, ϵ)-coreset for semi-spherical GMMs whose covariance matrices have bounded eigenvalues $\lambda_{min} \leq \lambda_i \leq \lambda_{max}$

Extensions and Generalizations

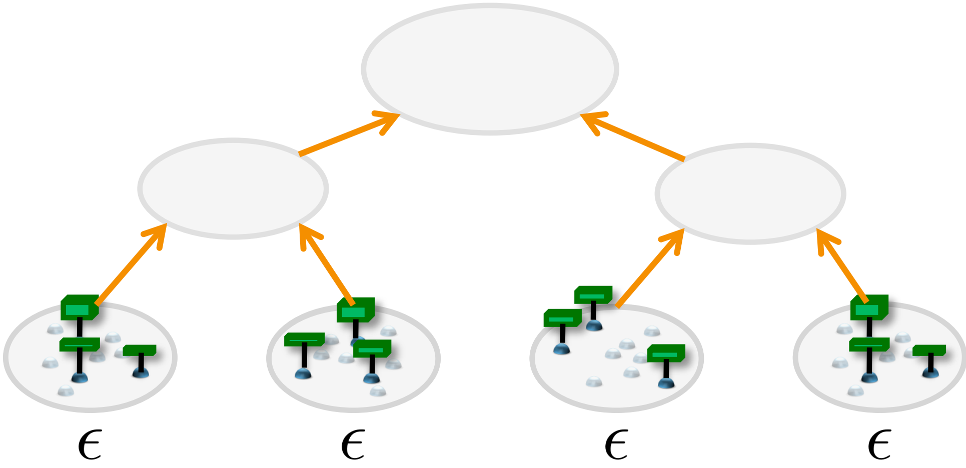
- Coresets for **non-spherical GMMs** can be obtained via reduction to recent projective clustering coresets
- Other mixtures (e.g. Laplace) based on $\boldsymbol{\ell_q}$ **distances** and **other norms** via generalized $\triangle$-inequality
- Efficient implementations in **Parallel (MapReduce)** and **Streaming** settings





GMM Coresets on Streams / in parallel

[Feldman et al '11]

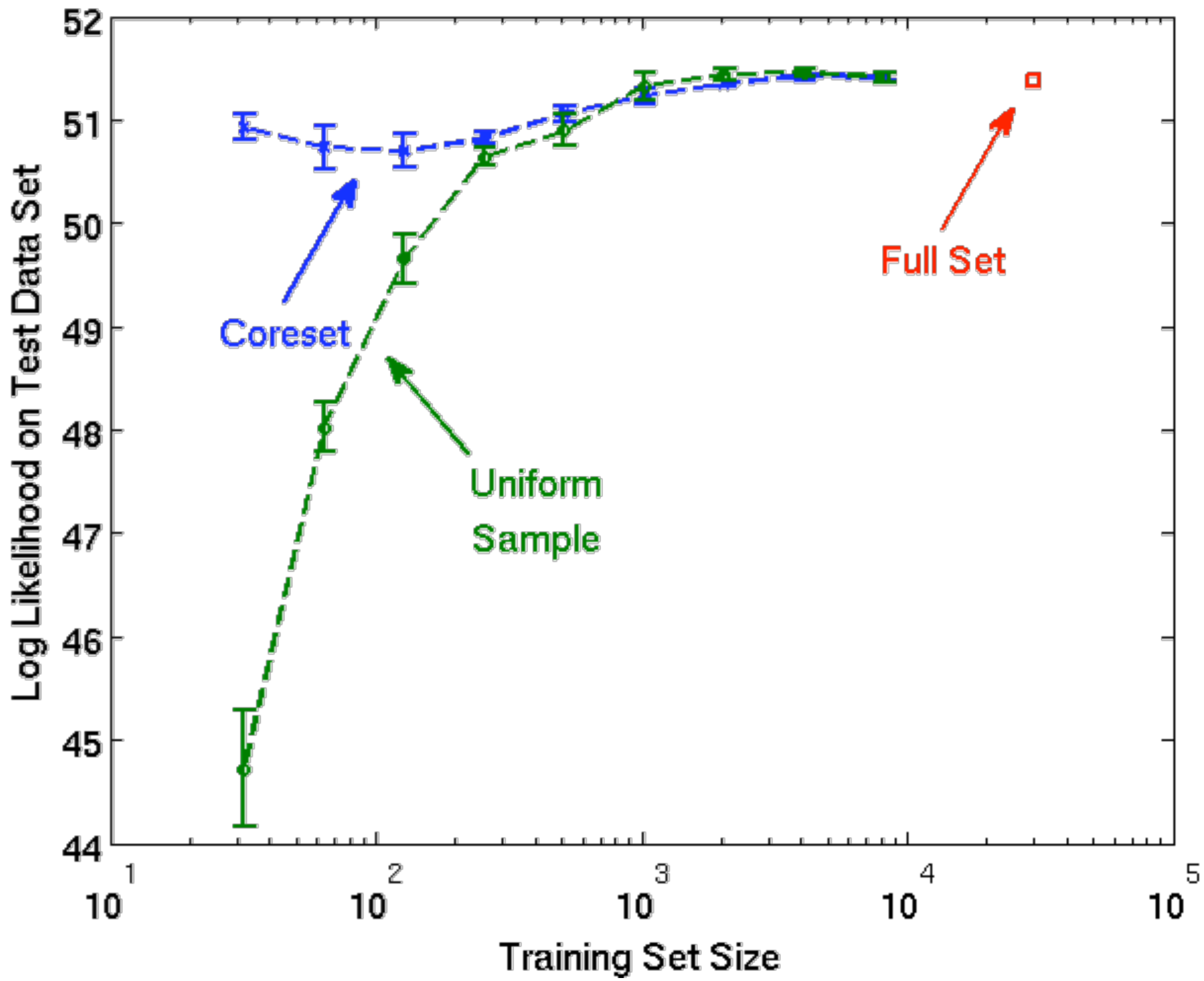


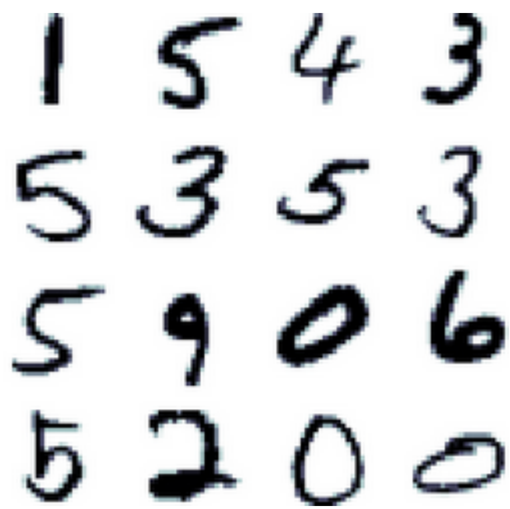
THM: a (k, ϵ)-coreset for a stream of n points $\in \mathbb{R}^d$ can be computed for ϵ-semi-spherical GMM with prob. $\geq (1-\delta)$ in space and update time $\text{poly}(dk\epsilon^{-1} \log(1/\delta) \log n)$

THM: a (k, ϵ)-coreset for n points $\in \mathbb{R}^d$ can be computed for ϵ-semi-spherical GMM with prob. $\geq 1-\delta$ using **m machines** in time $(\mathbf{n}/\mathbf{m})\text{poly}(dk\epsilon^{-1} \log(1/\delta) \log n)$

Handwritten Digits

Obtain 100-dimensional features from 28x28 pixel images via PCA. Fit GMM with k=10 components.





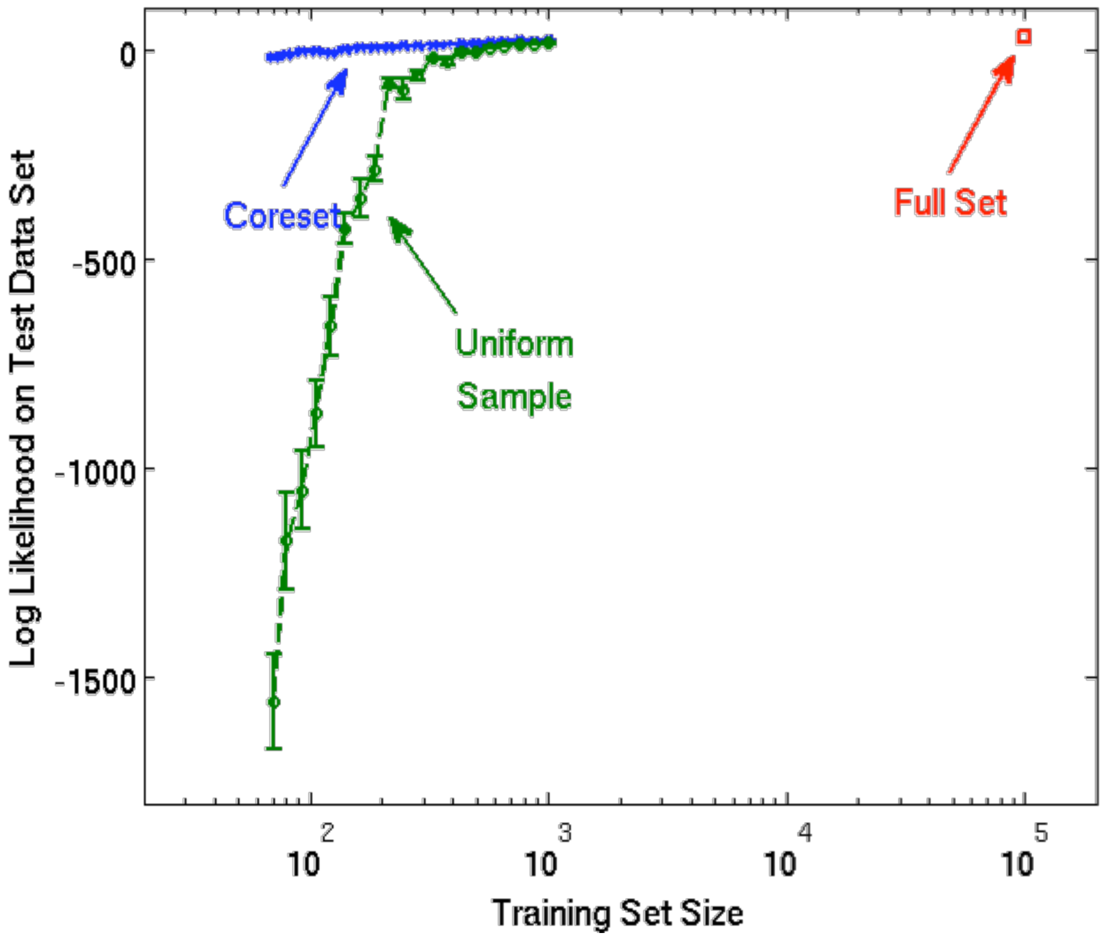
MNIST data:

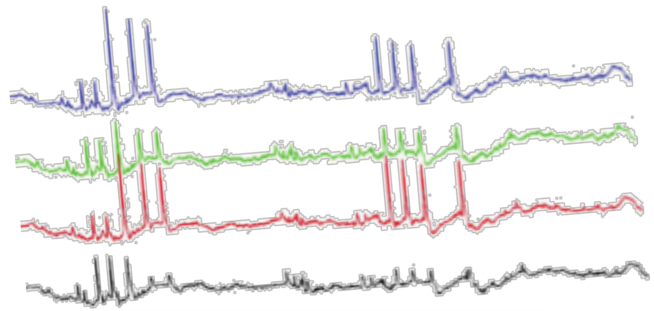
60,000 training,

10,000 testing

Neural Tetrode Recordings

Waveforms of neural activity at four co-located electrodes in a live rat hippocampus. 4 x 38 samples = 152 dimensions.



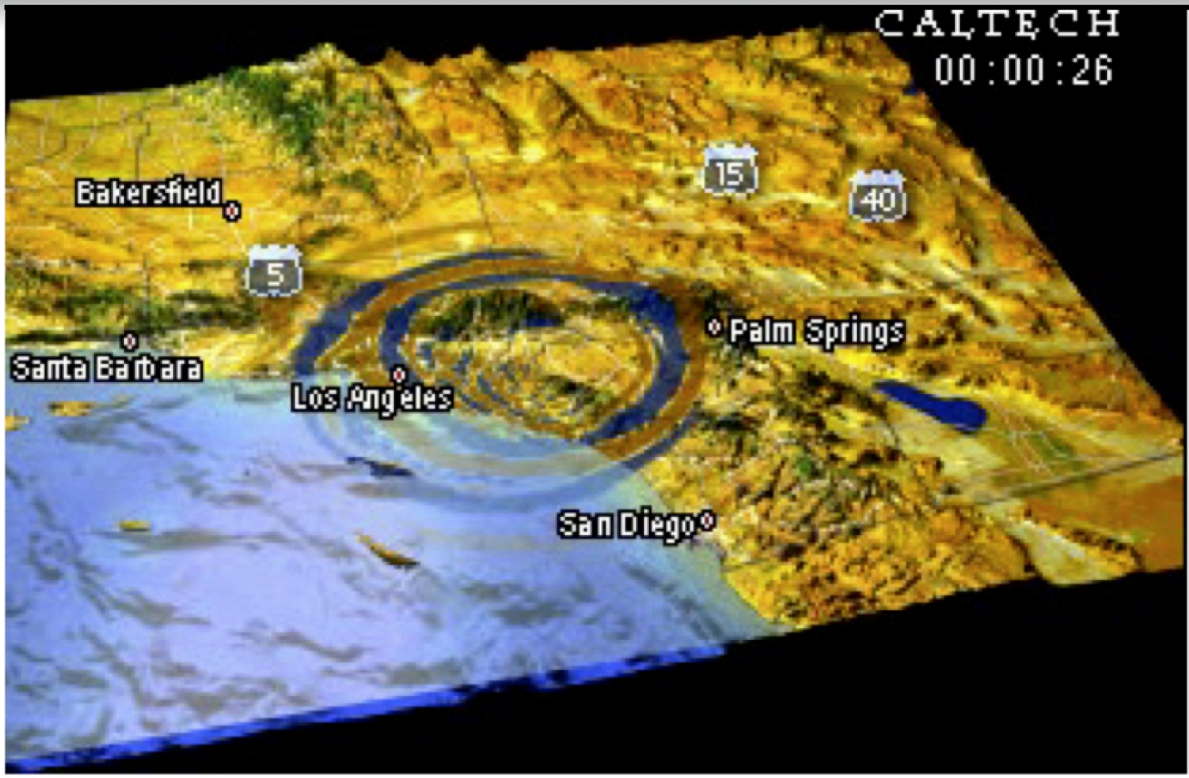


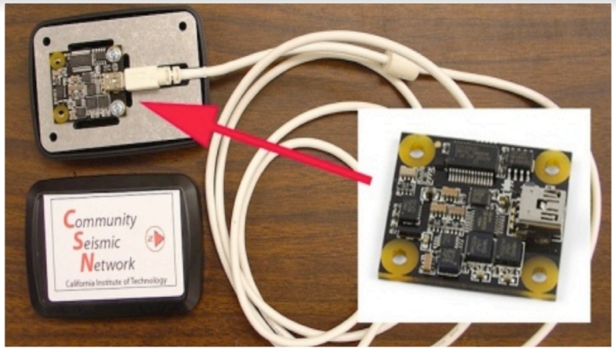
T. Siapas et al, Caltech

Method comparison

	Geometric (k-means)	**Probabilistic (GMM)**	
Batch	Classic K-means	EM	Slow
Online	Online k-means	Online (stepwise) EM	Very fast but not flexible / robust
Compression	Coresets	Coresets	Fast and accurate
	Simple interpretation	More flexible; “confidence” (e.g. for anomaly detection;	

Case study: Community Seismic Network
[w Clayton, Heaton, Chandy et al.]





Detect and monitor earthquakes using inexpensive accelerometers in cell phones and other consumer devices

Classical Hypothesis Testing

Naïve: send all accelerometer data to fusion center that decides ***Quake*** (E = 1) vs. ***No Quake*** (E = 0)

$$\frac{L_1(\text{accelerations})}{L_0(\text{accelerations})} > \tau$$

$$L_i(\text{accel.}) = \mathbb{P}[\text{accel.} \mid E = i]$$

L_1 → $\frac{P(E=1 \mid x)}{P(E=0 \mid x)}$ ← L_0

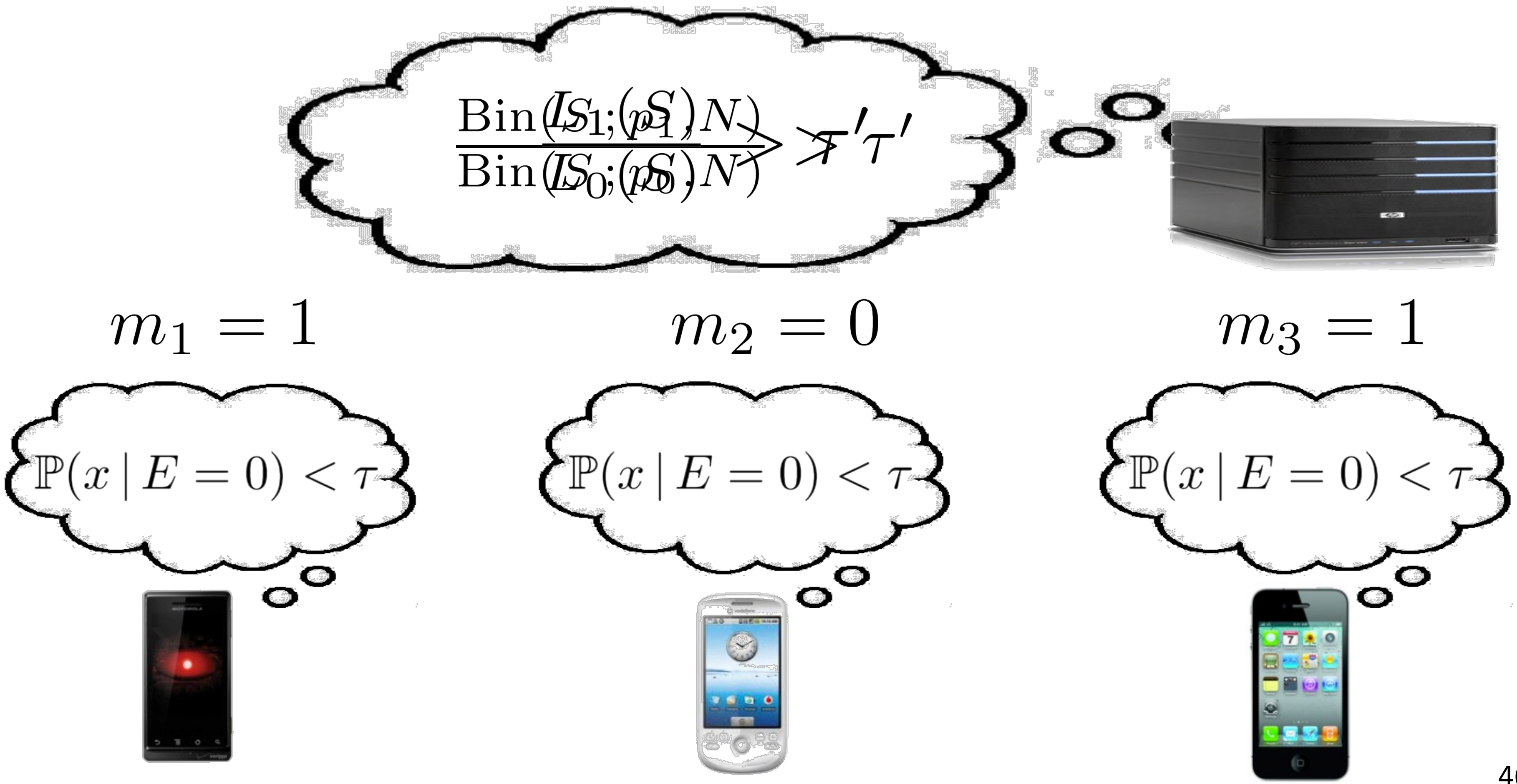
Fusion Center

1M phones produce 30TB of acceleration data a day!

Centralized solution does not scale.

Decentralized Anomaly Detection

The fusion center receives $S = \sum_{i=1}^{n} m_i$ "picks" from *N* sensors. The optimal decision rule is the hypothesis test:



Controlling False Positive Rates

For rare events, nearly *all* positives are *false* positives.

1. False Pick rate p_0
2. System-wide False Alarm rate P_F

$\mathbb{P}[x \,|\, E=0] < \tau$ ← τ controls false pick rate

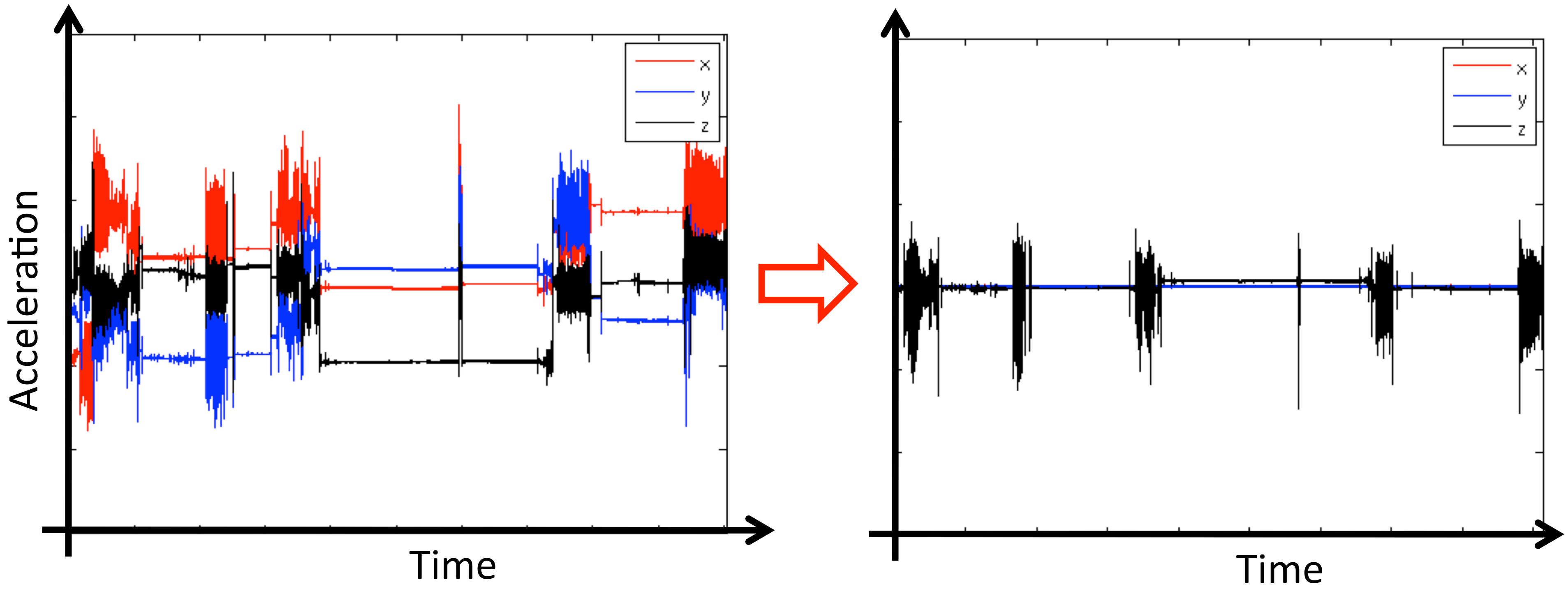
Can learn τ, e.g. online percentile estimation

$$P_F = \sum_{S:\text{"alarm"}} \text{Bin}(S\,;\, p_0, N)$$

← Don't depend on p_1 (true pick rate)

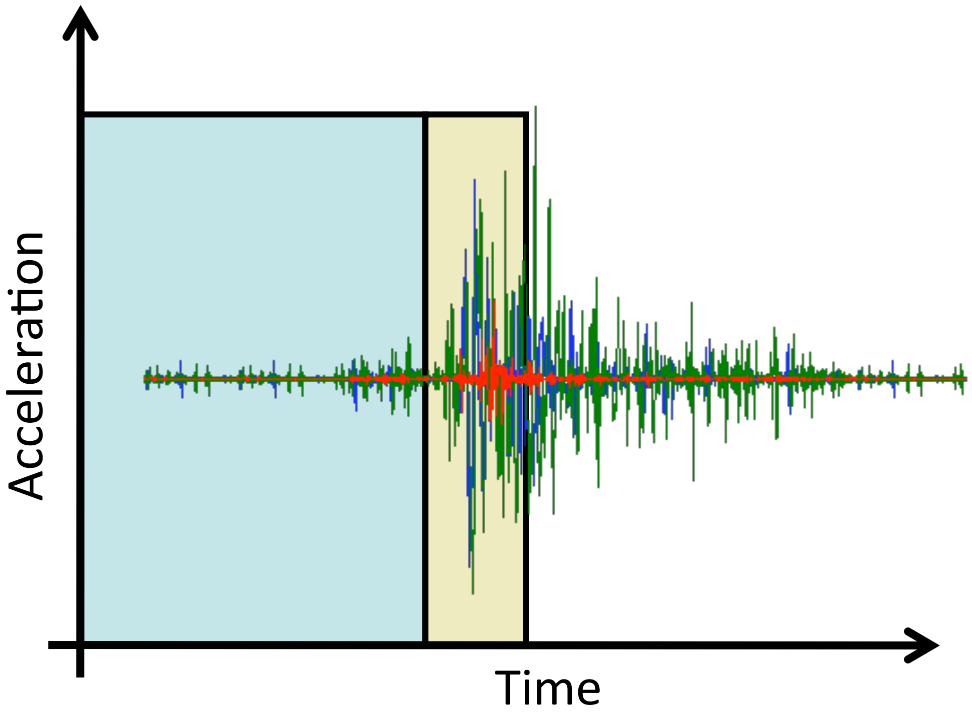
Controls messages and false alarms without $\mathbb{P}(x \,|\, E=1)$!

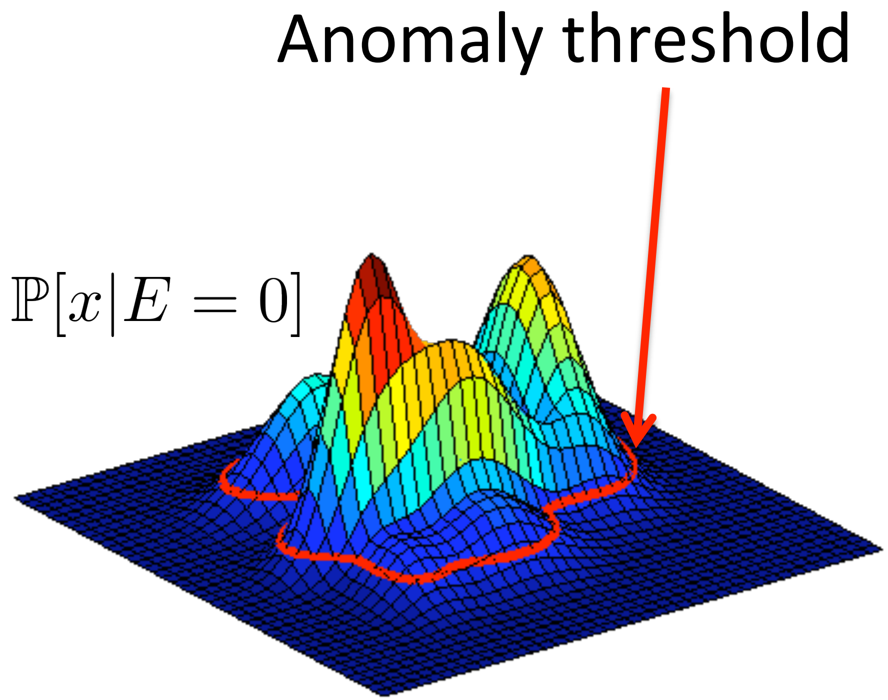
Analyzing data on the phone



- Removing gravity

Analyzing data on the phone

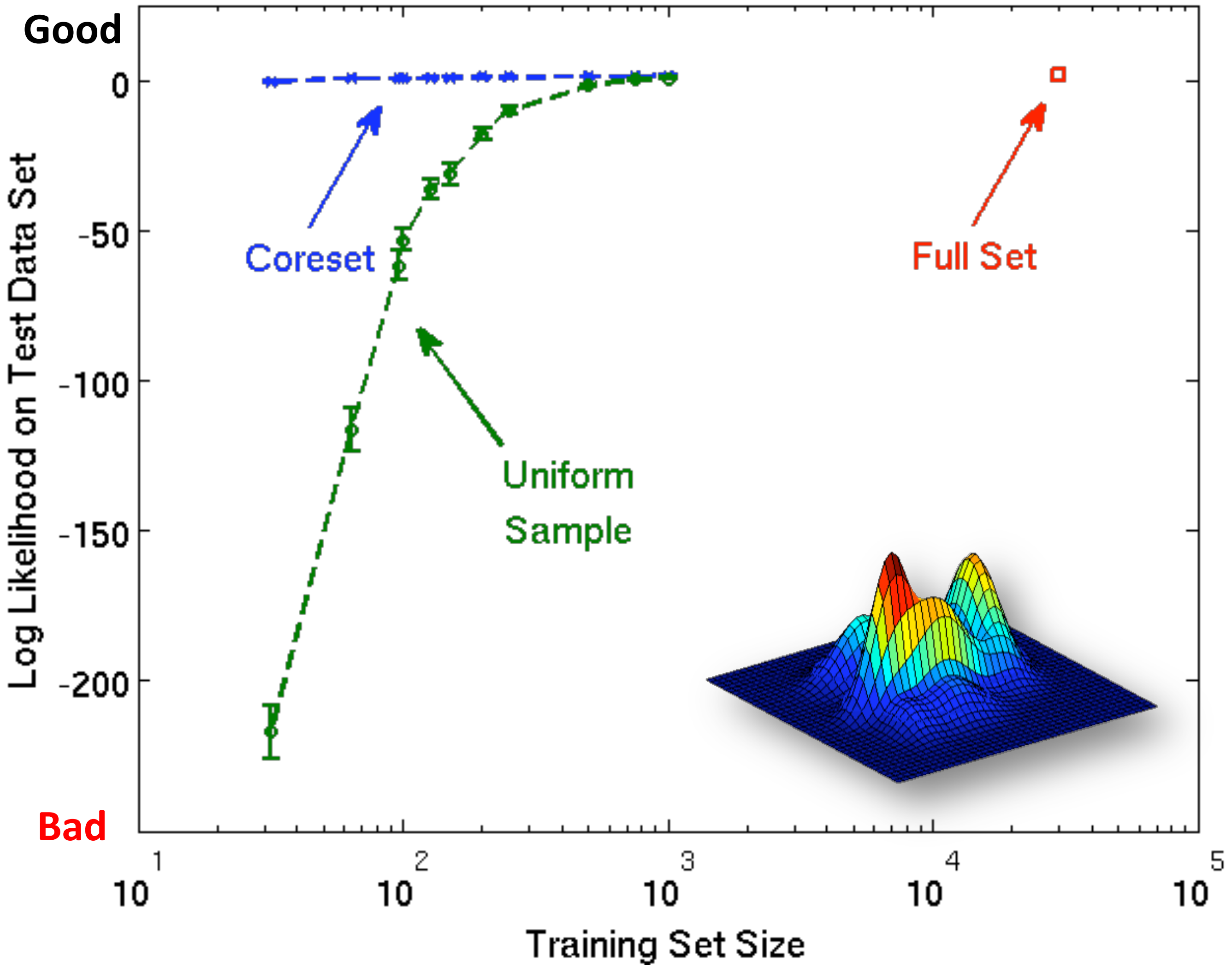




- Calculate "fingerprints" of accelerometer data (frequency spectra, moments, …)
- Learn (online) statistical models of normal behavior

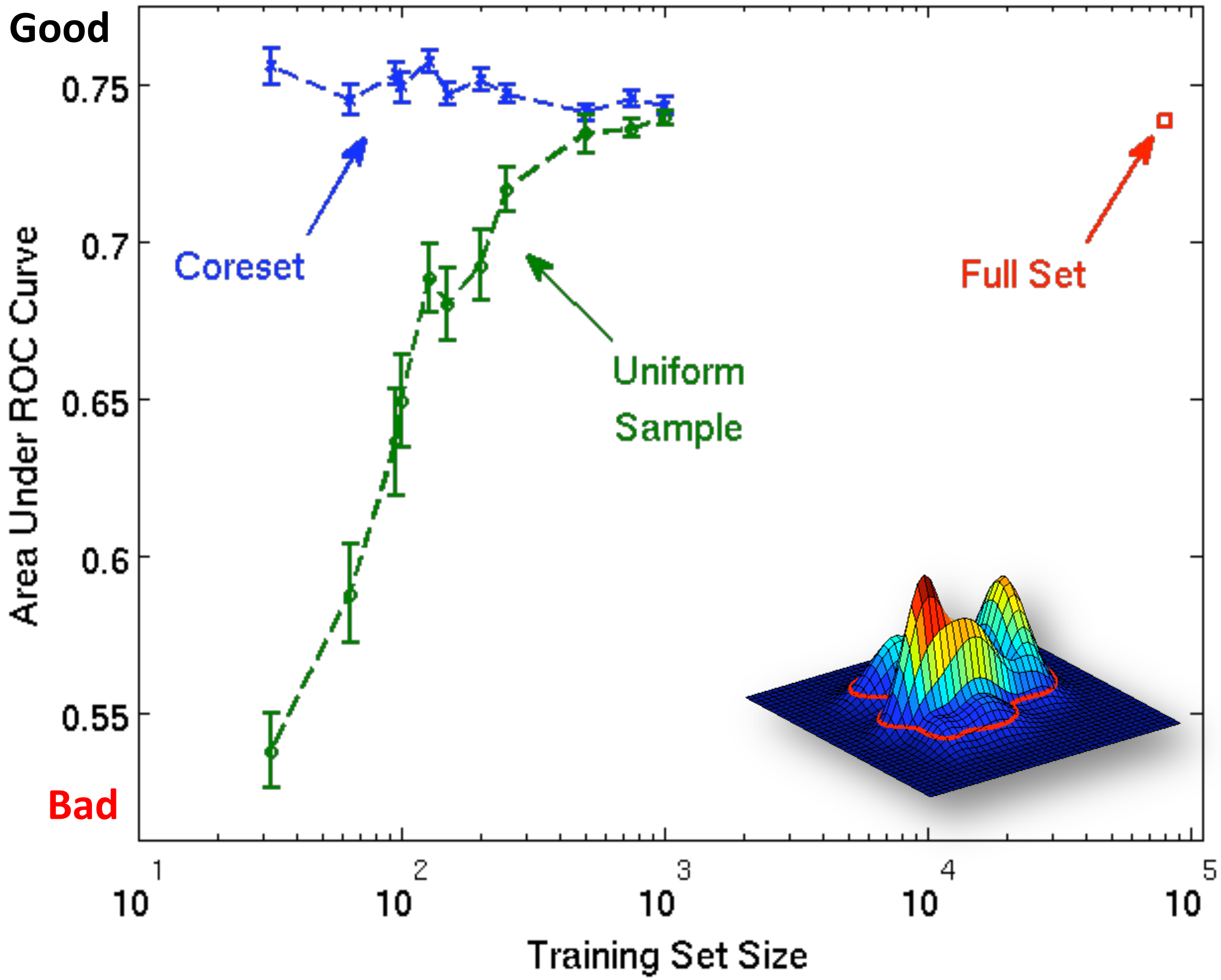
Learning User Acceleration

17-dimensional acceleration feature vectors



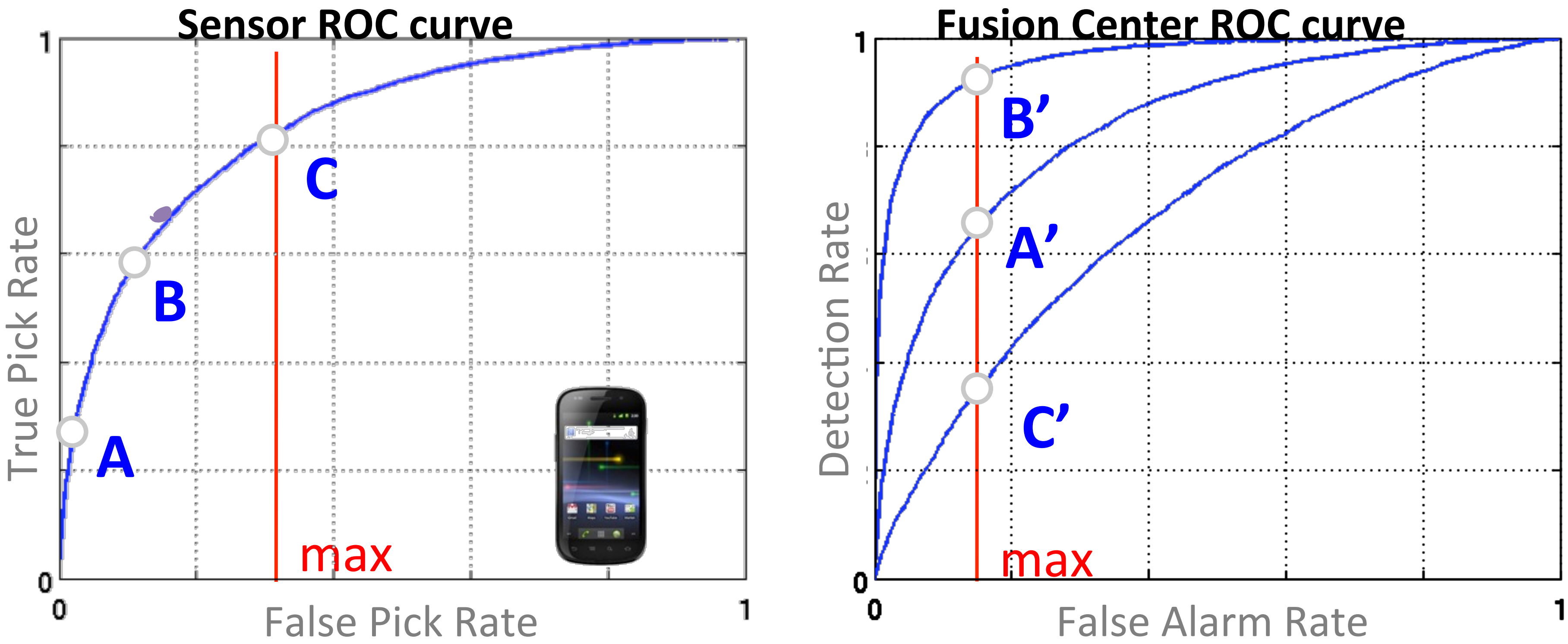
Seismic Anomaly Detection

GMM used for anomaly detection



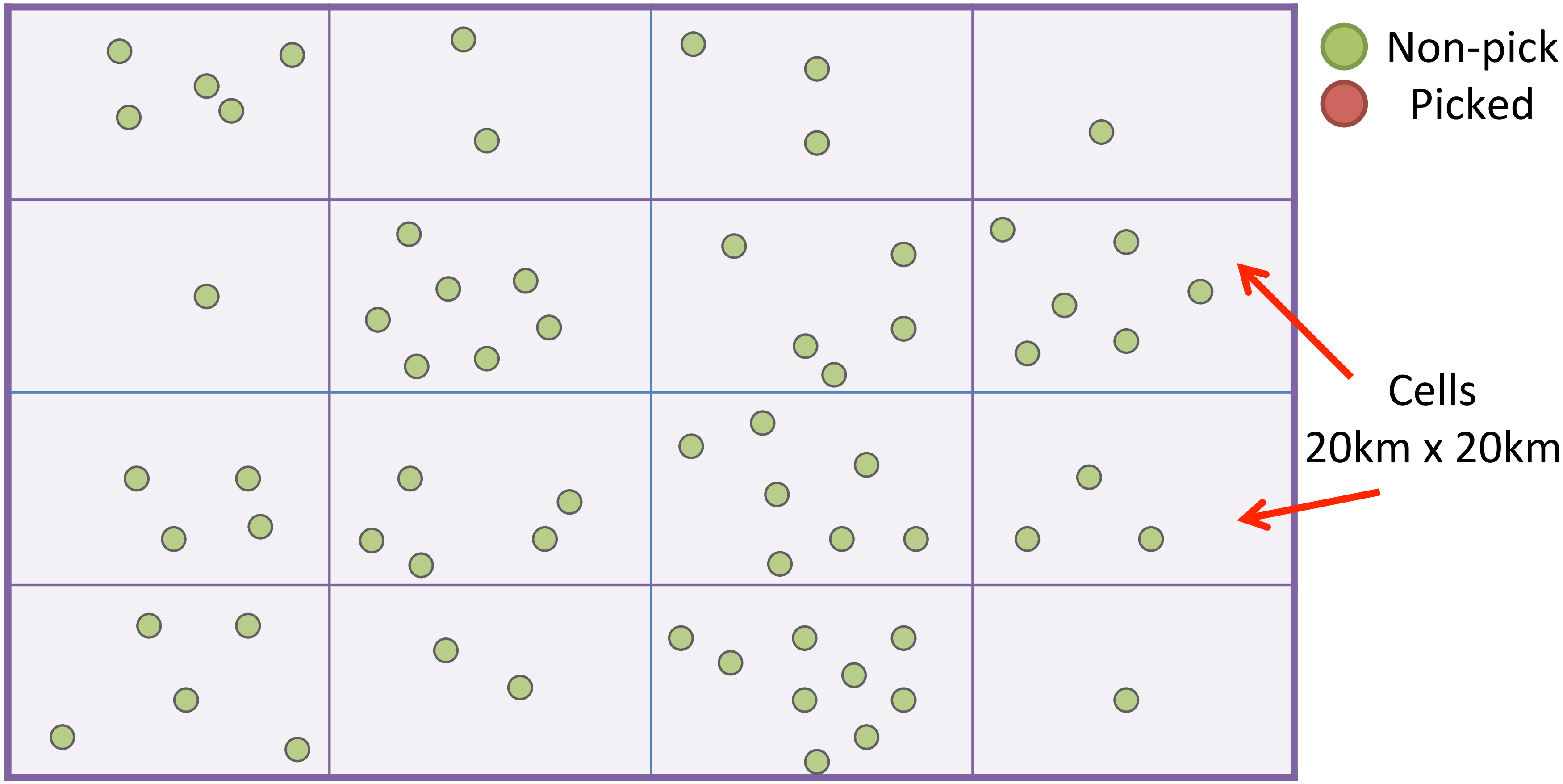
Joint Threshold Optimization

Maximize detection performance, under constraints on sensor messages and system false alarm rate



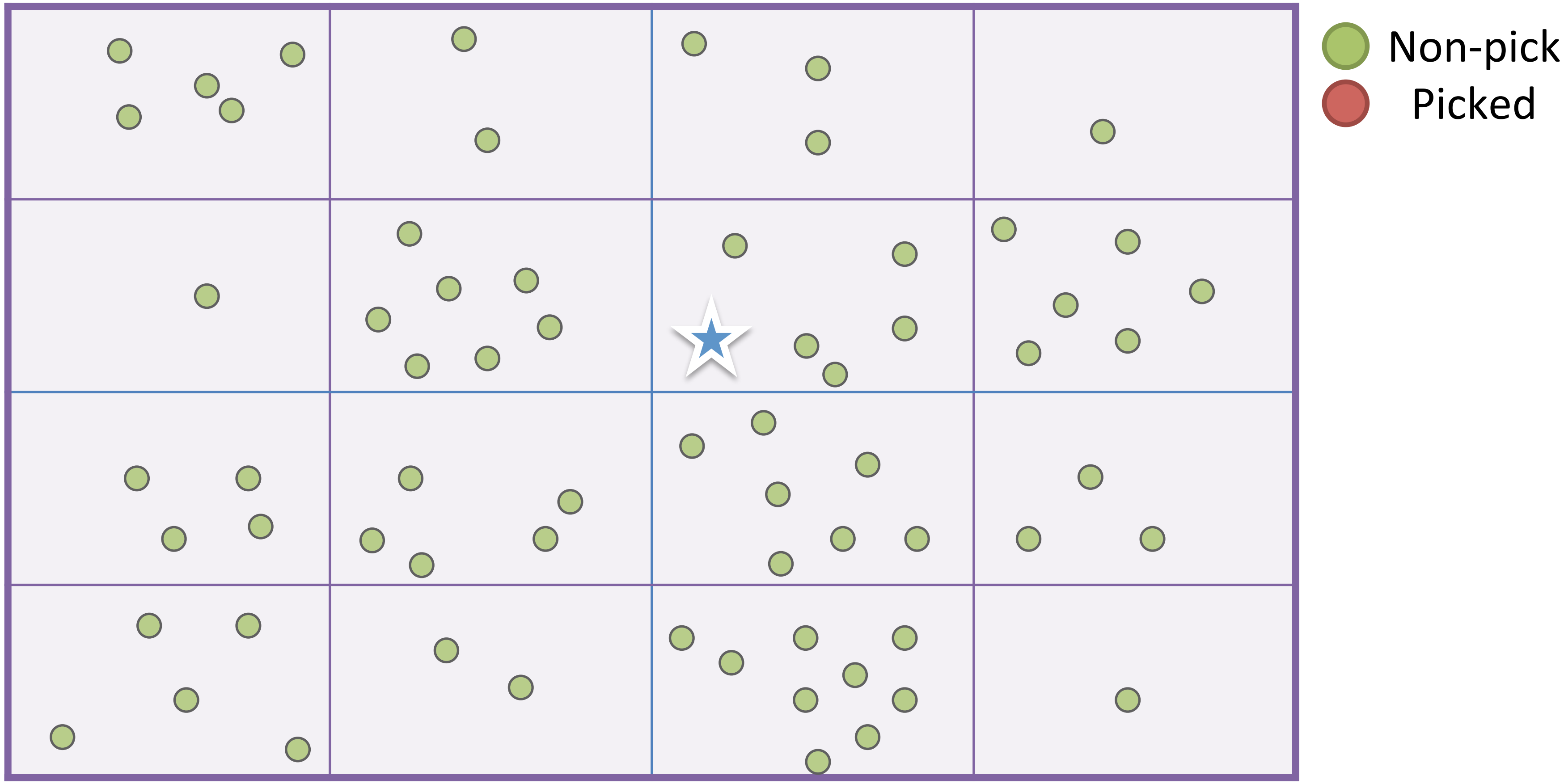
Sensor and Fusion Center thresholds are optimized, e.g. by grid search, subject to constraints

Decentralized anomaly detection



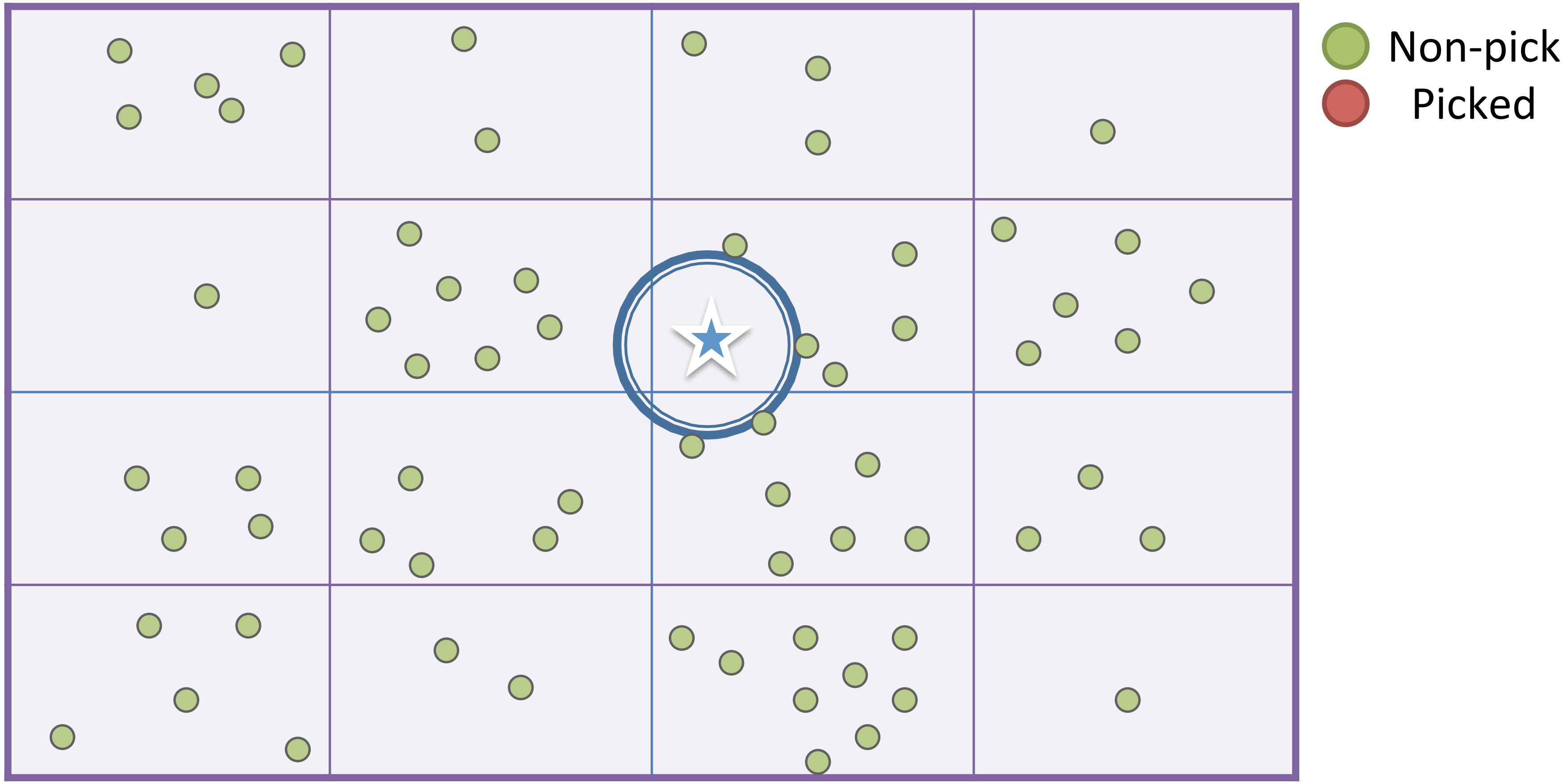
Each cell performs decentralized anomaly detection

Decentralized anomaly detection



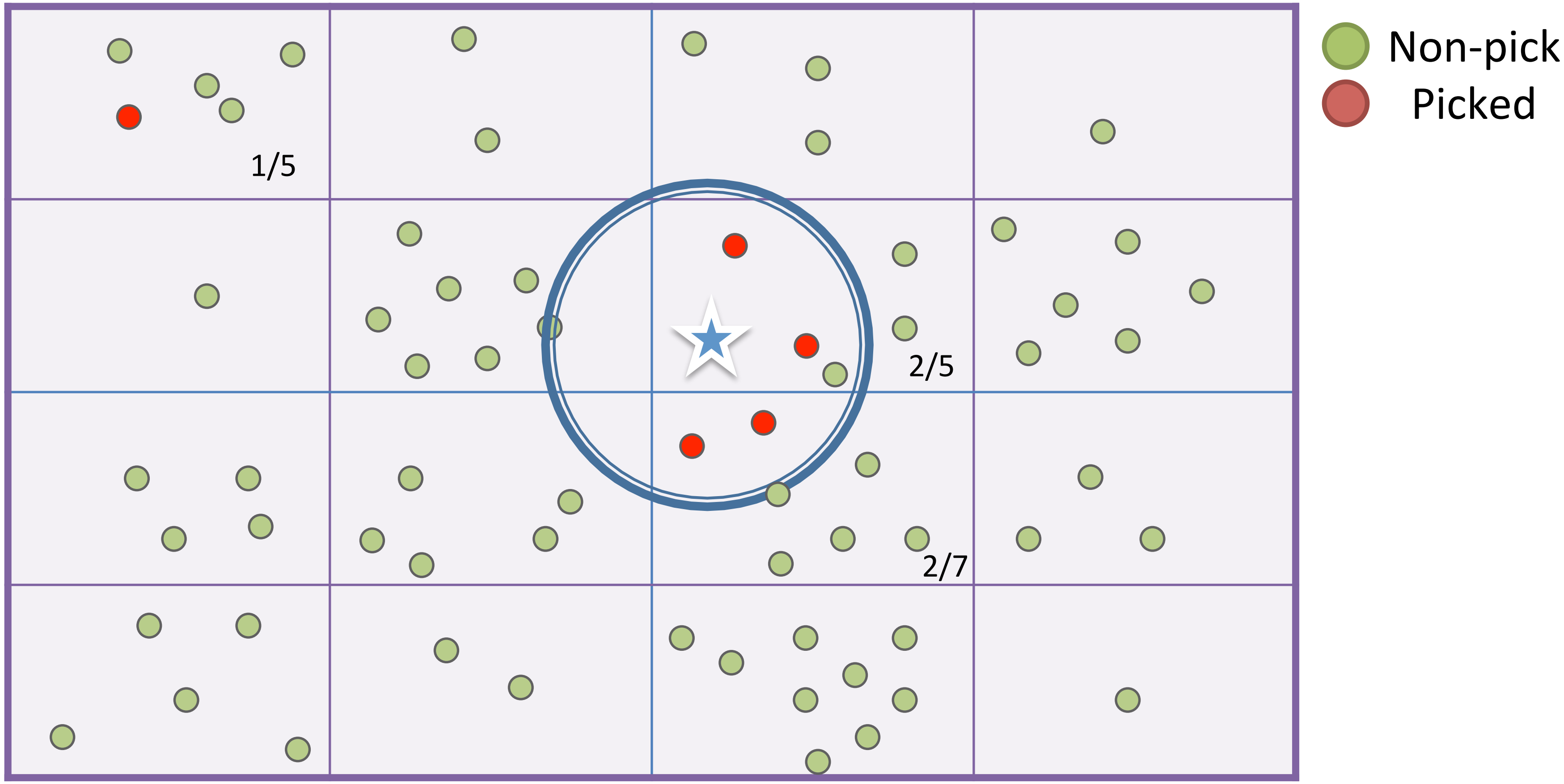
Each cell performs decentralized anomaly detection

Decentralized anomaly detection



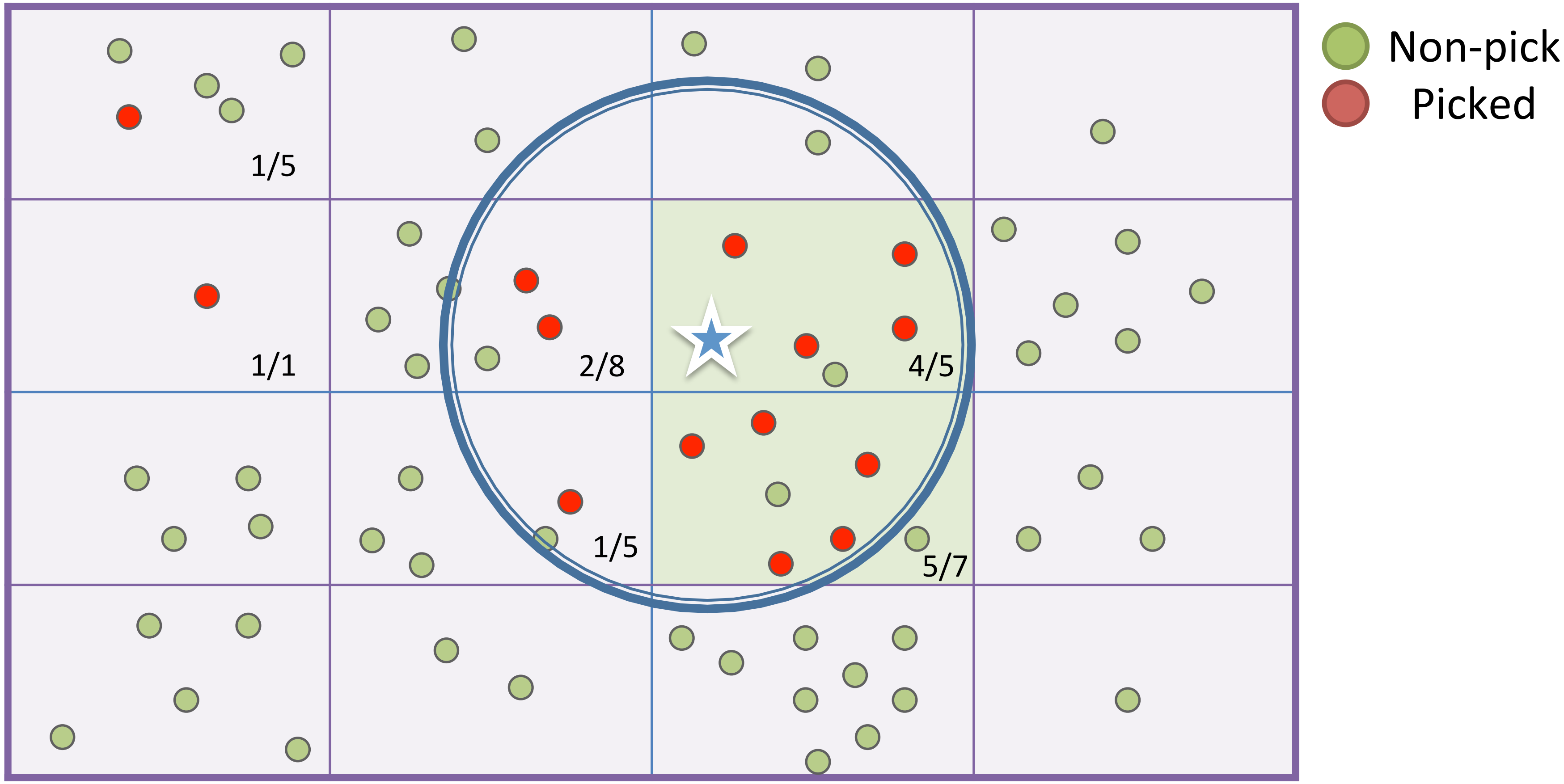
Each cell performs decentralized anomaly detection

Decentralized anomaly detection



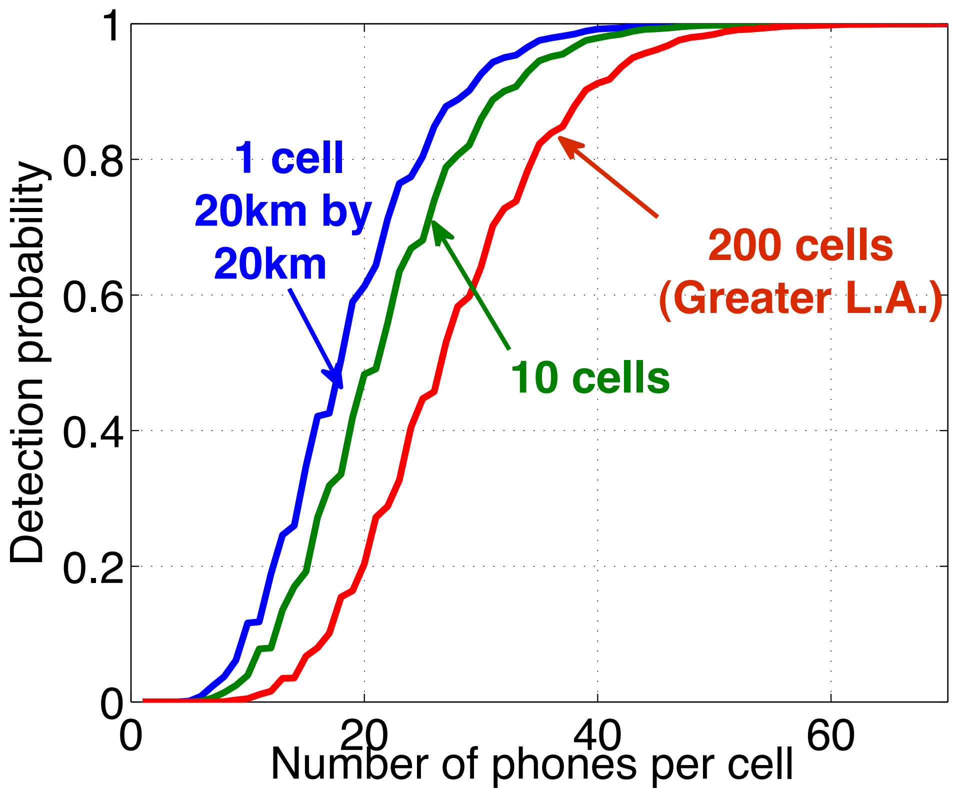
Each cell performs decentralized anomaly detection

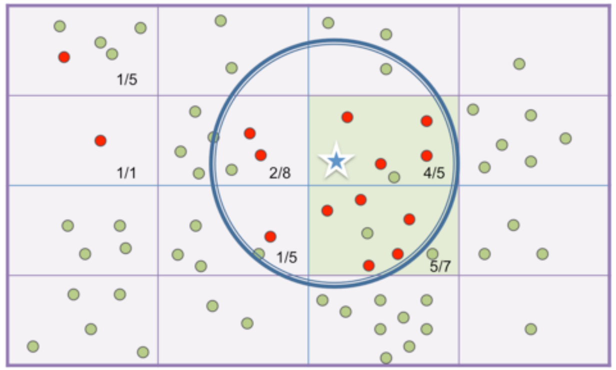
Decentralized anomaly detection



- Each cell performs decentralized anomaly detection

Detection performance





What density of phones do we need to ensure < 1 false alarm per year?

Larger area protected
➔ More false positives
➔ higher phone density needed

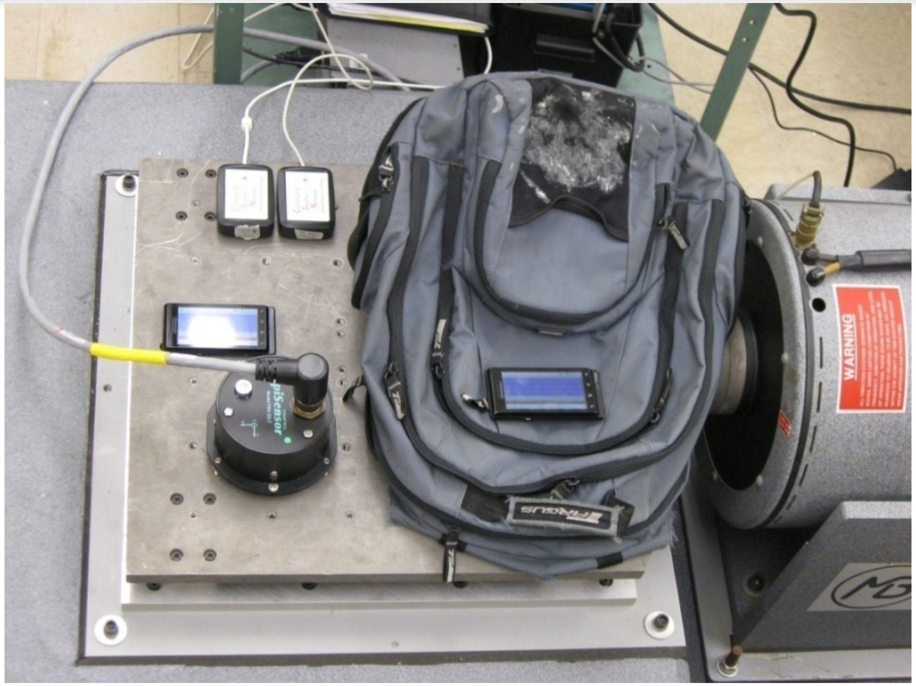
<u>Preliminary estimate:</u>

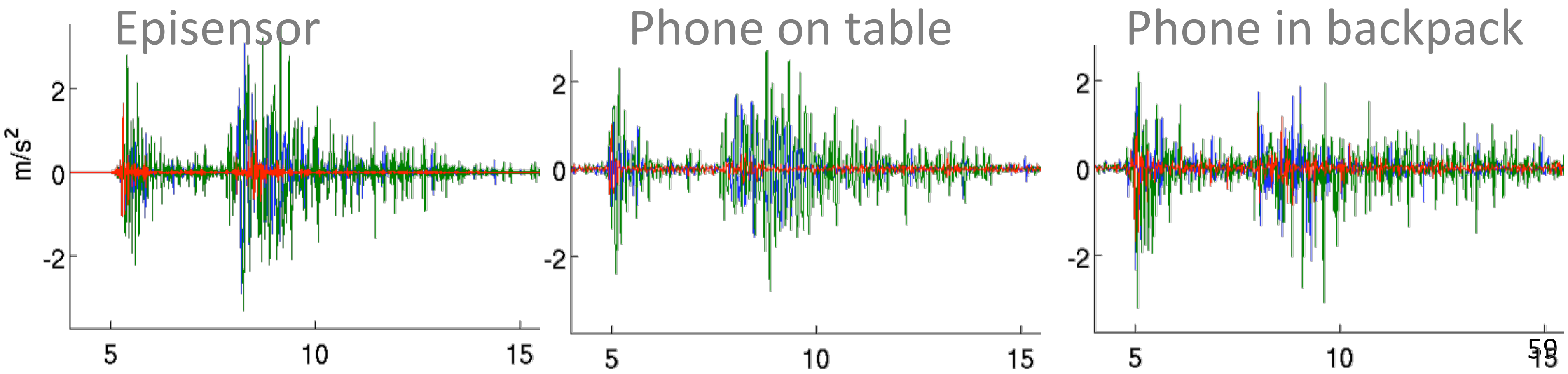
Need ~10k-20k active phones for Greater L.A. area to detect event of magnitude 5 or higher

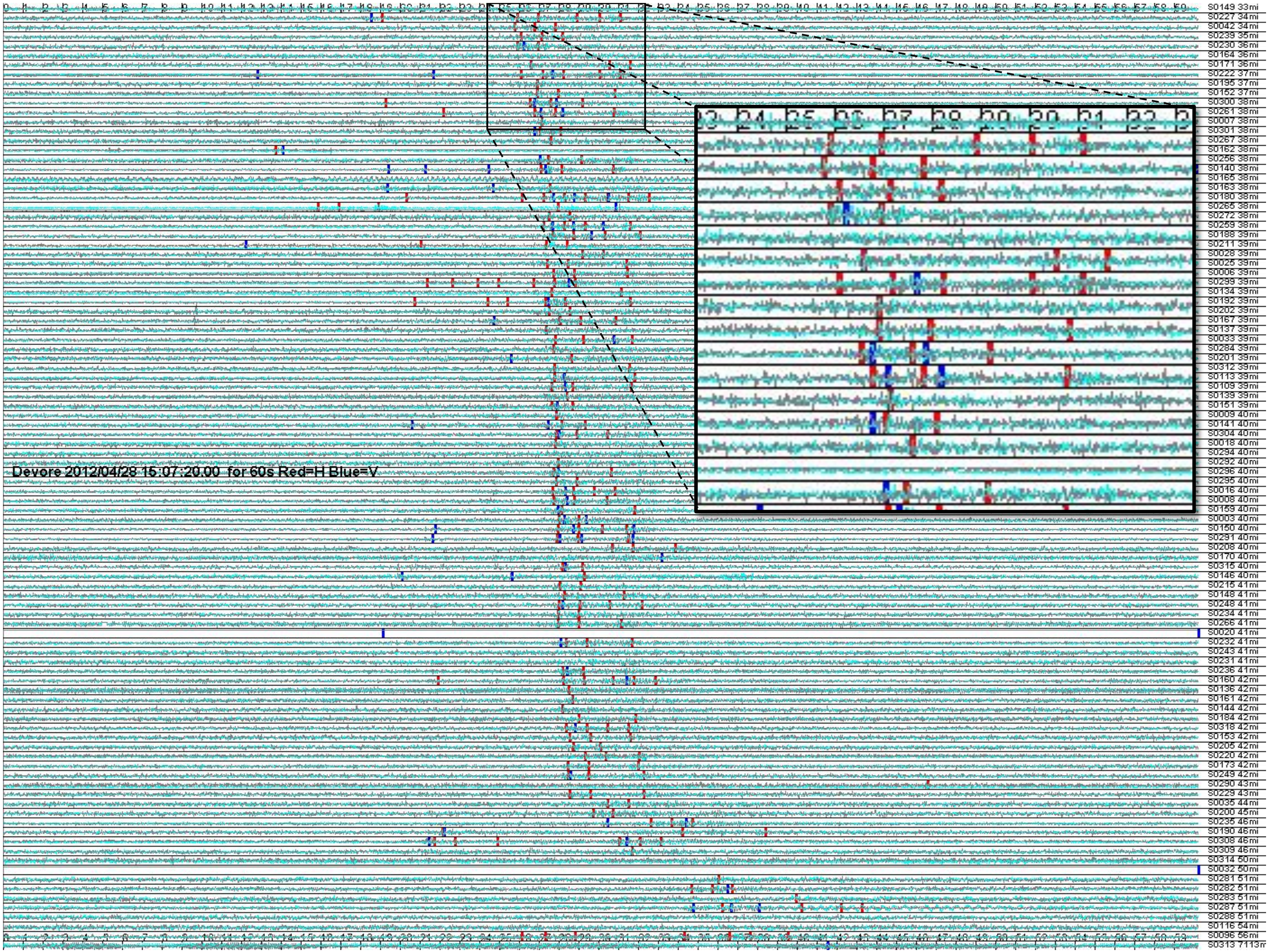
Shake Table Validation

Empirically compared sensors and tested pick algorithm on historic M6-8 quakes.

All 6 events triggered picks from the phones






Devore 2012/04/28 15:07:20.00 for 60s Red=H Blue=V

Lessons learned: From batch to online

- Batch algorithms (SVM, k-means, EM, …) infeasible for large data sets
- Key property that allows scaling: Loss function (hinge loss, quantization error, …) *decomposes additively* over data points
- Simple trick to get online algorithms: update parameters after processing each data point (or small subset)
- For supervised learning, loss functions are convex
  ➔ online convex programming guaranteed to converge
- For unsupervised learning, loss typically non-convex
  ➔ online k-means/EM only converge to local optimum
  ➔ want to "summarize" (compress) data set to do better

Acknowledgments

- The slides are partly based on material By Chris Bishop, Andrew Moore and Danny Feldman