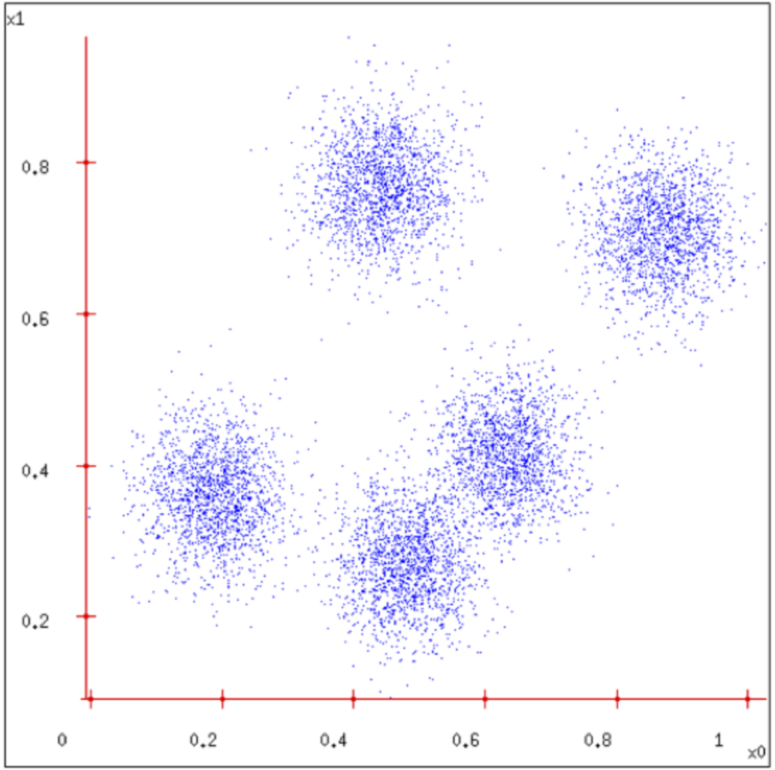
Johns Hopkins
Whiting School of Engineering

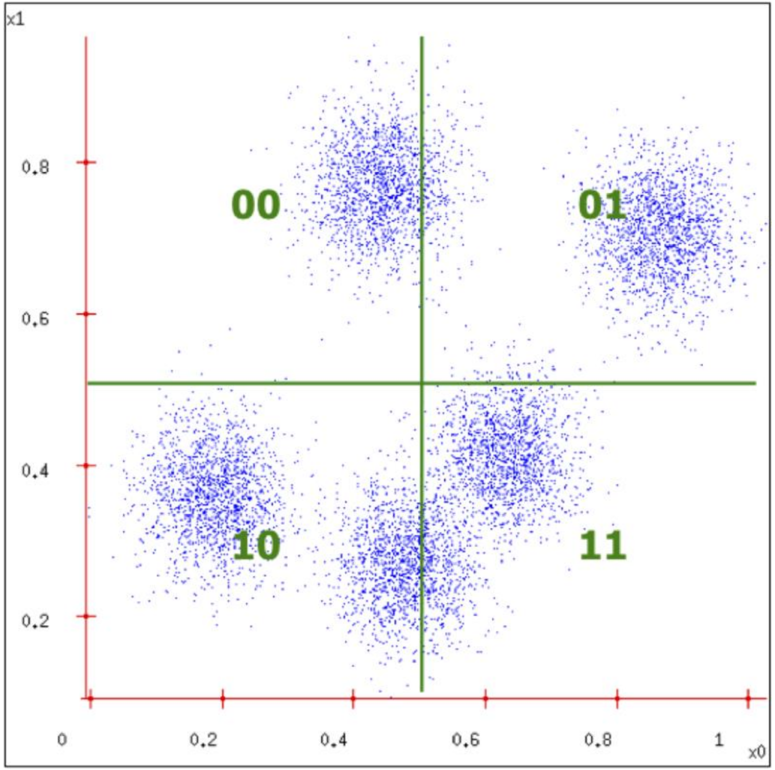
Introduction to Machine Learning

k-Means and k-medoids Clustering

Some Data



Grid Clustering



Expectation Maximization

A local search process where we estimate parameters and then adjust them to increase likelihood of correctness

Example – Find one of the roots of $x^5 - 3x^2 + 2x - 17 = 0$.

- Rewrite as $x = (3x^2 - 2x + 17)^{1/5}$.
- "Guess" that $x = 1$.
- Substitute into right hand side and recalculate, $x = 1.7826$.
- Repeat until convergence

$$1 \to 1.7826 \to 1.8716 \to 1.8845 \to 1.8864 \to 1.8866 \to 1.8867 \to \cdots$$

The proof sets up what we call a "contraction map."

EM for Clustering

- Bivariate Gaussians with equal variance
- *k*-means clustering

- **Assumptions**
  - We have *k* multi-variate Gaussian/spherical clusters.
  - Each cluster has an unknown mean vector: $\langle \boldsymbol{\mu}_1, \ldots \boldsymbol{\mu}_k \rangle$.
  - We do not know which Gaussian generated which data point.
- We can apply EM to find the cluster means.
- We can use the cluster means to assign the points.

k-Means Clustering

Example

- Let $k = 2$.
- Describe a generated instance as $y_i = \langle x_i, z_{i1}, z_{i2} \rangle$, where $z_{ij} = 1$ means x_i was generated by cluster j.
- Begin by guessing values for $h = \langle \mu_1, \mu_2 \rangle$ at random.

E Step: (Expectation)

$$E[z_{ij}] = \frac{P(x_i|\mu_j)}{\sum_{k=1}^{2} P(x_i|\mu_k)} = \frac{\exp\left[-\frac{1}{2\sigma}(x_i - \mu_j)^2\right]}{\sum_{k=1}^{2} \exp\left[-\frac{1}{2\sigma}(x_i - \mu_k)^2\right]}$$

M Step: (Maximization, $h' = \langle \mu_1', \mu_2' \rangle$)

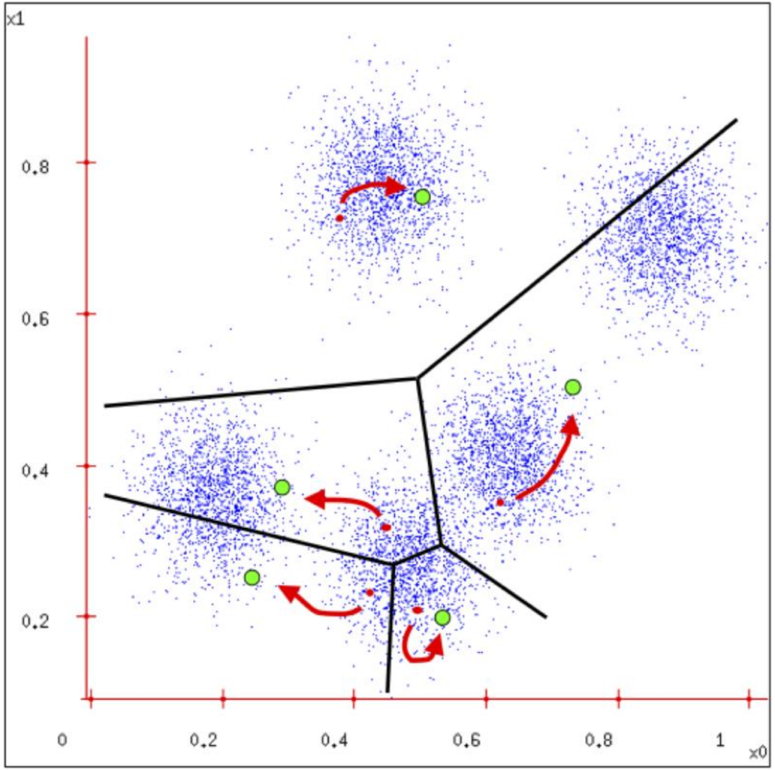
$$\mu_j = \frac{\sum_{i=1}^{m} E[z_{ij}] x_i}{\sum_{i=1}^{m} E[z_{ij}]}$$

k-Means Clustering

Algorithm 10.3 K-Means Clustering

```
function KMeans(𝒟, k)
    initialize μ_1, ..., μ_k randomly
    repeat
        for all x_i ∈ 𝒟 do
            c ← arg min_{μ_j} d(x_i, μ_j)        ▷ d() is the distance between x_i and μ_j.
            assign x_i to the cluster c
        end for
        recalculate all μ_j based on new clusters
    until no change in μ_1, ..., μ_k
    return μ_1, ..., μ_k
end function
```

Illustrating *k*-Means



Another Illustration of *k*-Means

1 2 3 5 6 7 10

$K = 2$

$$\underbrace{1\ 2\ 3\ 5}_{2.5} \qquad \underbrace{6\ 7\ 10}_{7.67}$$

$$\underbrace{1\ 2\ 3}_{2} \qquad \underbrace{3\ 6\ 7\ 10}_{7.5}$$

K-Medoids

- Representative object or member of a data set
- Different objective functions
  - *K*-means

$$J(\mathcal{C}) = \sum_{j=1}^{K} \sum_{x_i \in \mathcal{C}} \left(x_i - c_j\right)^2$$

  - *K*-medoids

$$J(\mathcal{C}) = \sum_{j=1}^{K} \sum_{x_i \in \mathcal{C}} \left(x_i - m_j\right)^2$$

Partitioning Around Medoids (PAM)

Algorithm 10.4 K-Medoids Clustering

function KMEDOIDS-PAM($\mathcal{D}, k$)
    select $\mathbf{m}_1, \ldots, \mathbf{m}_k$ randomly
    repeat
        for all $\mathbf{x}_i \in \mathcal{D}$ **do**
            $c \leftarrow \arg\min_{\mathbf{m}_j} d(\mathbf{x}_i, \mathbf{m}_j)$ $\triangleright$ $d()$ is the distance between $\mathbf{x}_i$ and $\mathbf{m}_j$.
            assign $\mathbf{x}_i$ to the cluster c
        end for
        $distortion_{k\text{-medoids}} = \sum_{j=1}^{k} \sum_{i \in ownedby(\mathbf{c}_j)} (\mathbf{x}_i - \mathbf{m}_j)^2$
        for all $\mathbf{m}_i \in \mathbf{m}$ **do**
            for all $\mathbf{x}_j \in \mathcal{D}$ where $\mathbf{x}_j \notin \mathbf{m}$ **do**
                swap $\mathbf{m}_i$ and $\mathbf{x}_j$
                $distortion'_{k\text{-medoids}} = \sum_{j=1}^{k} \sum_{i \in ownedby(\mathbf{c}_j)} (\mathbf{x}_i - \mathbf{m}_j)^2$
                if $distortion_{k\text{-medoids}} \leq distortion'_{k\text{-medoids}}$ **then**
                    swap back
                end if
            end for
        end for
    until no change in $\mathbf{m}_1, \ldots, \mathbf{m}_k$
    return $\mathbf{m}_1, \ldots, \mathbf{m}_k$
end function

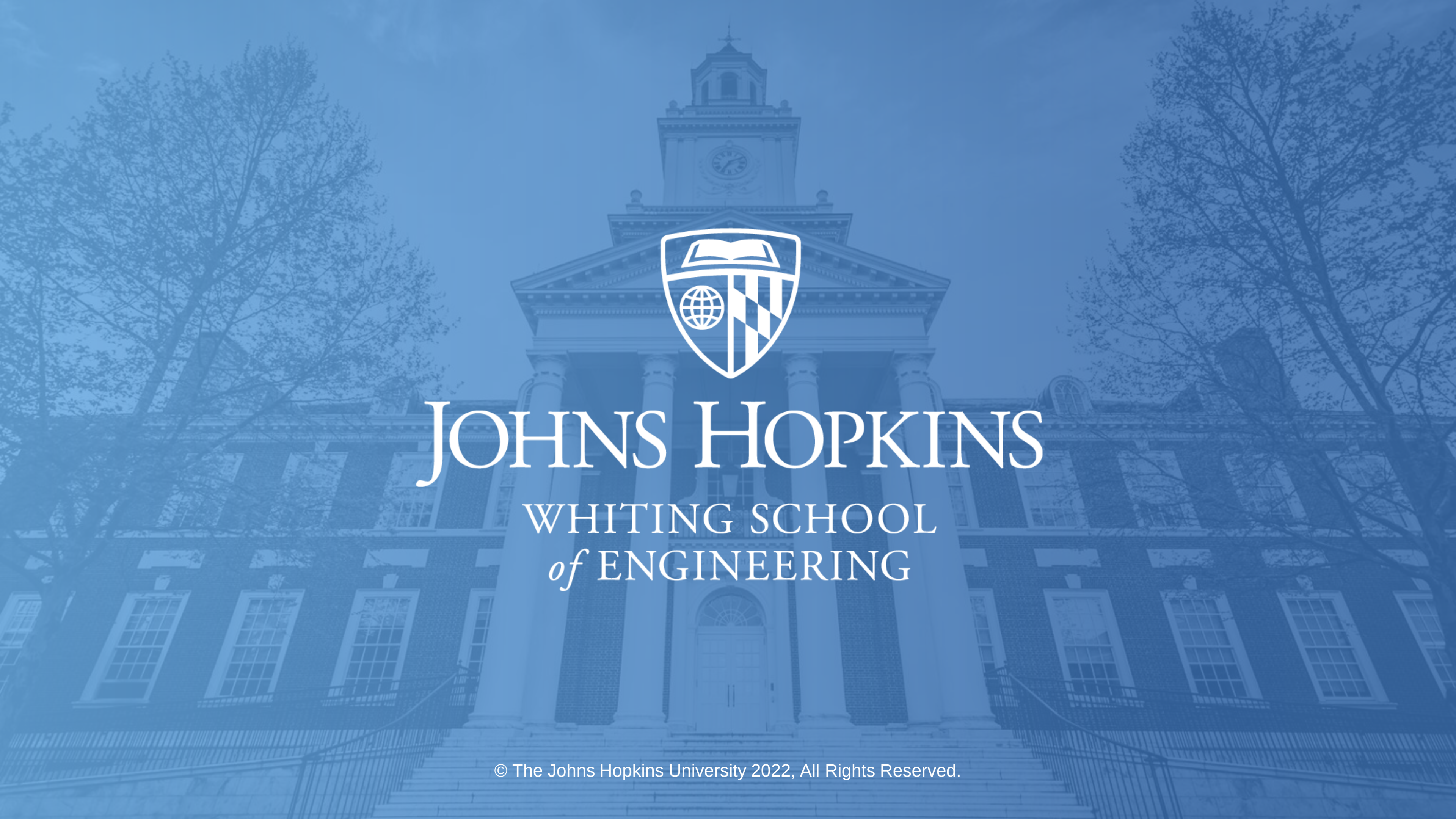
Fuzzy *c*-Means

- A method to create “soft” clusters, where f is a level of ”fuzzification” in the range $1 \ldots n$
  - $f = 1$ indicates “crisp” clusters
  - $f > 1$ indicates ”soft” clusters
- Objective function

$$J(\mathcal{C}) = \sum_{i=1}^{n} \sum_{j-1}^{c} w_j(x_i)^f \left\| c_j - x_i \right\|^2$$

$$w_j(x_i) = \frac{1}{\sum_{k=1}^{c} \left(\frac{\left\| c_j - x_i \right\|}{\left\| c_k - x_i \right\|} \right)^{2/(f-1)}}$$

$$c_j = \frac{\sum_{i=1}^{n} w_j(x_i)^f x_i}{\sum_{i=1}^{n} w_j(x_i)^f}$$



Johns Hopkins

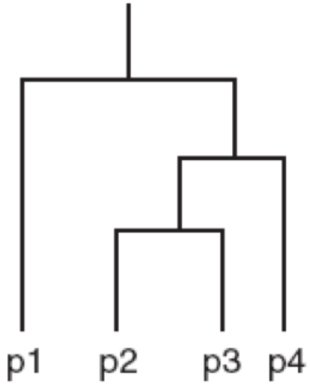
Whiting School of Engineering

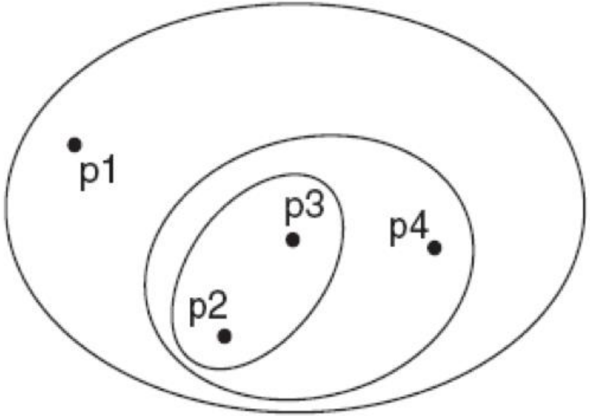
JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Introduction to Machine Learning

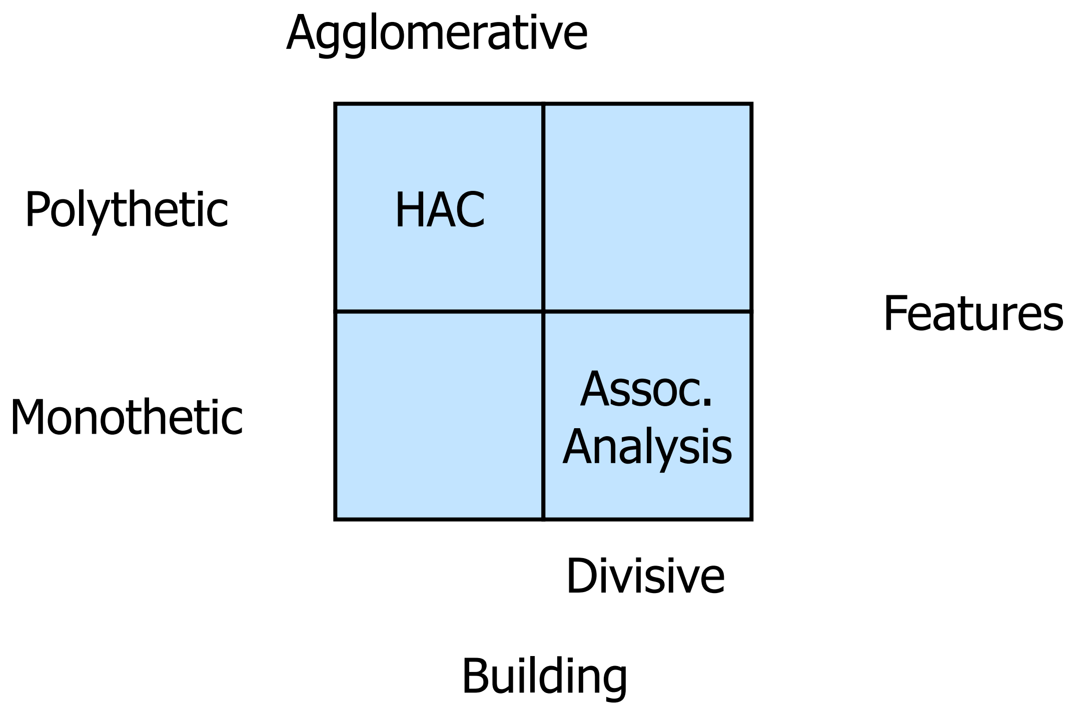
Hierarchical Agglomerative Clustering

Dendrogram

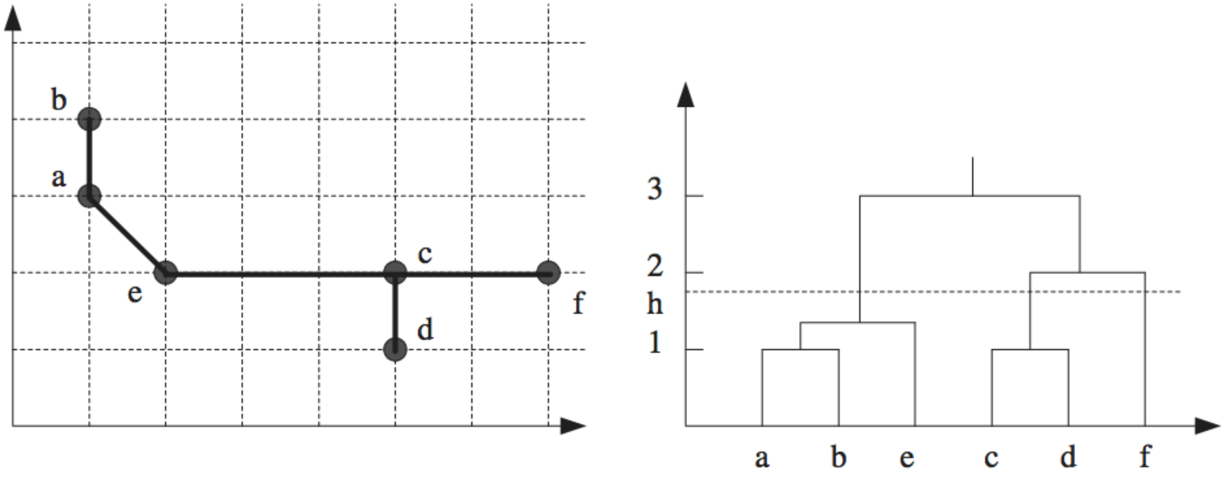




Agglomerative vs Divisive



Hierarchical Agglomerative Clustering



Association Analysis

- Recall

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

- Binary → Normalized
- F_{ij} is the j^{th} attribute of the i^{th} data point
- F_{ik} is the k^{th} attribute of the i^{th} data point

$$a_{jk} = \sum\nolimits_{x_i} F_{ij} \times F_{ik}$$

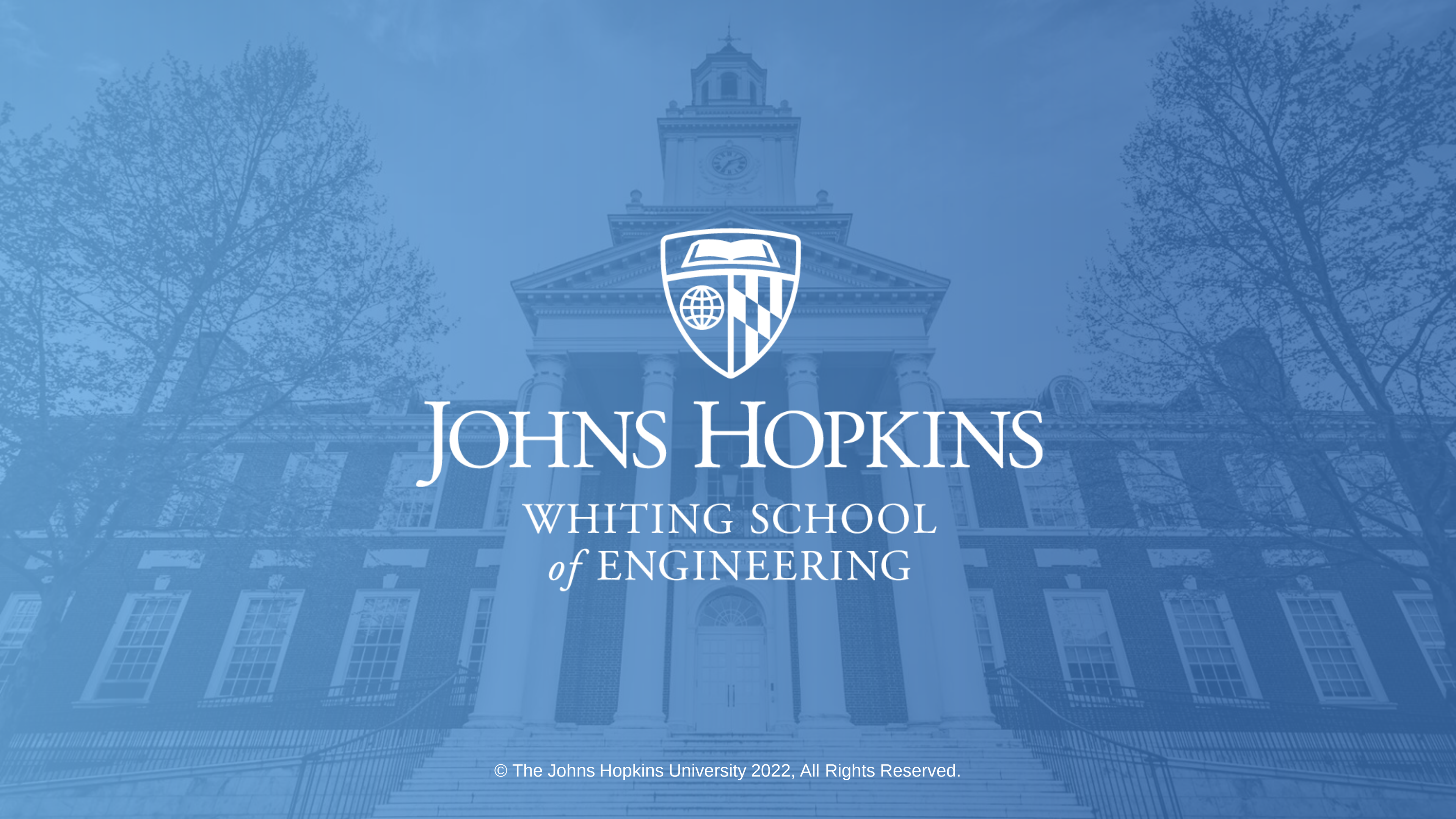
$$b_{jk} = \sum\nolimits_{x_i} (1 - F_{ij}) \times F_{ik}$$

$$c_{jk} = \sum\nolimits_{x_i} F_{ij} \times (1 - F_{ik})$$

$$d_{jk} = \sum\nolimits_{x_i} (1 - F_{ij}) \times (1 - F_{ik})$$

$$\chi^2_{jk} = \frac{(ad - bc)^2}{(a-b)(a-c)(b-d)(c-d)}$$

$$F_{split} = \operatorname{argmax} \sum\nolimits_{k=1}^{d} \chi^2_{jk}$$

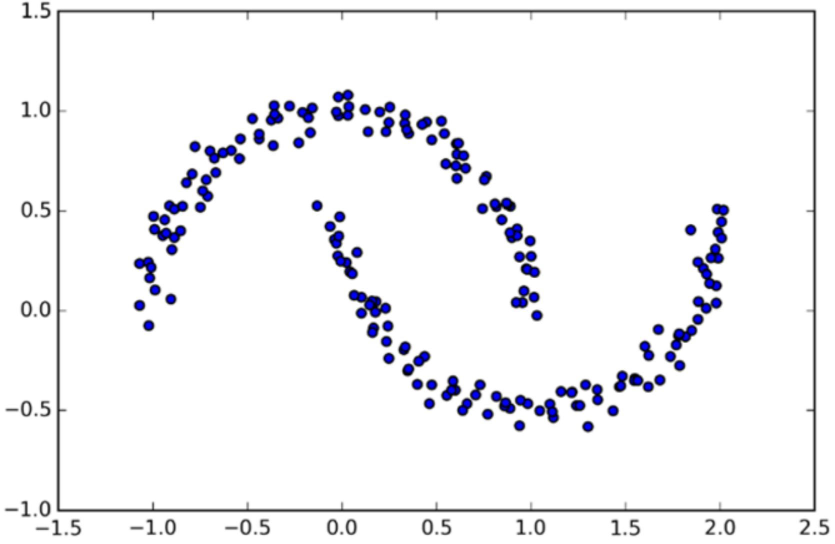

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Introduction to Machine Learning

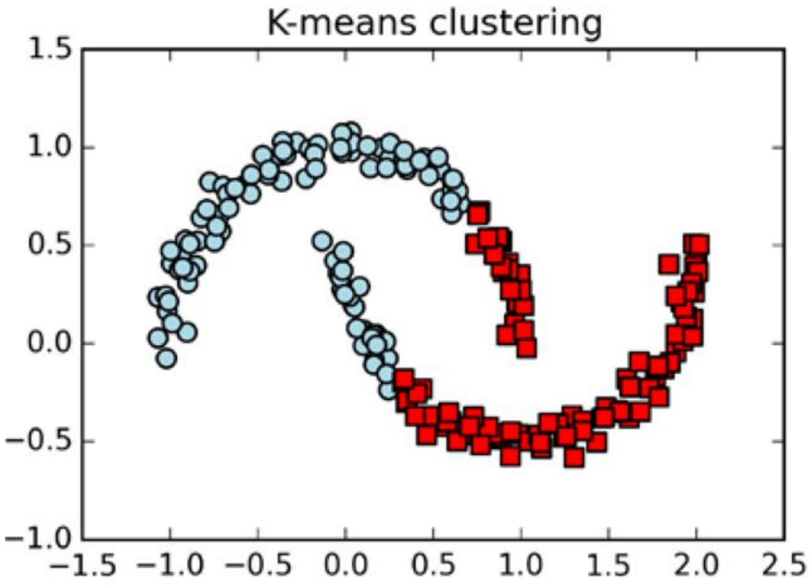
Density-based Clustering

Johns Hopkins
Whiting School of Engineering

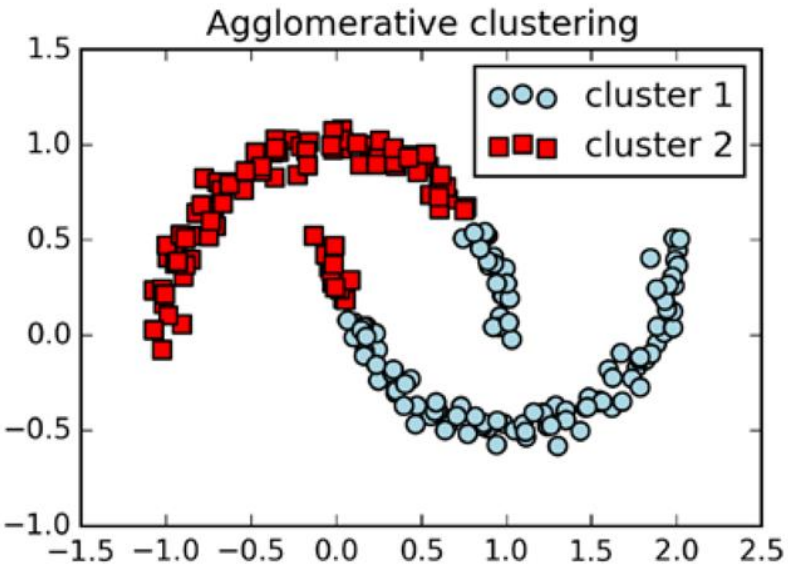
A Motivating Example



K-Means

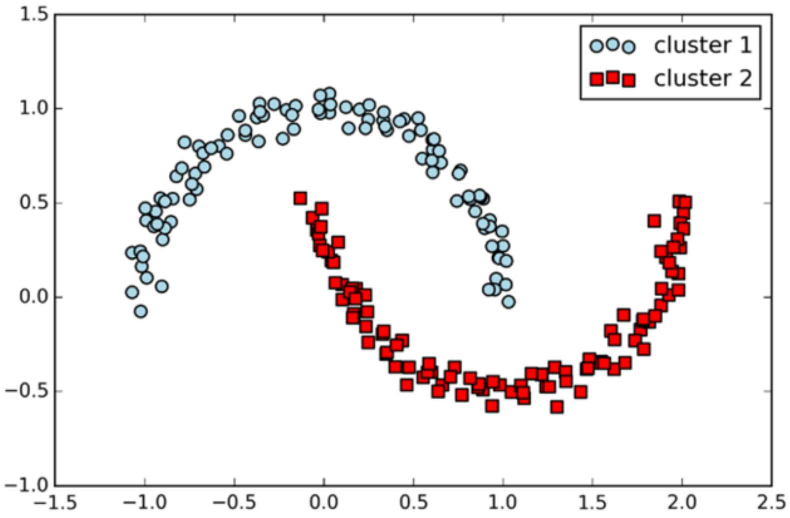


HAC

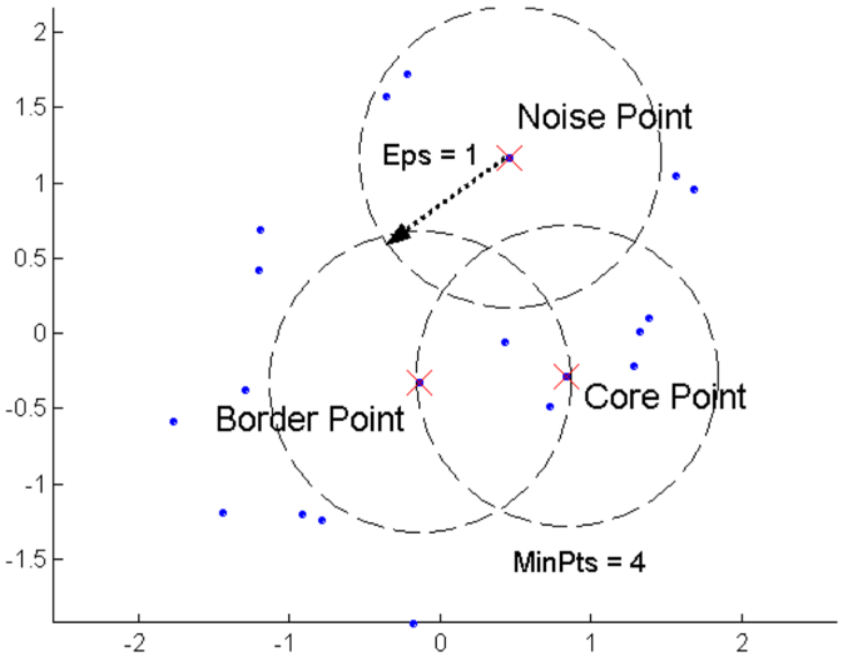


Agglomerative clustering
cluster 1
cluster 2
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5

Density-based Clustering



Point Categorization



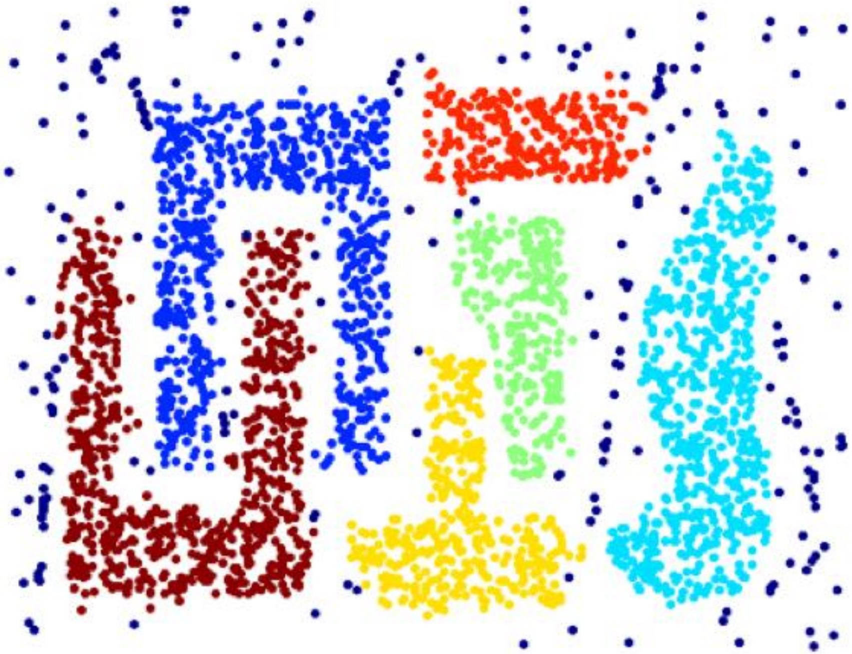
DBSCAN

Algorithm 2 DB-Scan

```
function DB-SCAN(𝒟)
    currClustLbl ← 1
    for all p ∈ Core do do
        if clustLbl[p] = "Unknown" then
            currClustLbl ← currClustLbl + 1
            clustLbl[p] ← currClustLbl
        end if
        for all p′ ∈ θ-neighborhood do
            if clustLbl[p′] = "Unknown" then
                clustLbl[p′] ← currClustLbl
            end if
        end for
    end for
    return clustLbl
end function
```

Another Example

Another Example



JOHNS HOPKINS

WHITING SCHOOL *of* ENGINEERING

Introduction to Machine Learning

Spectral Clustering

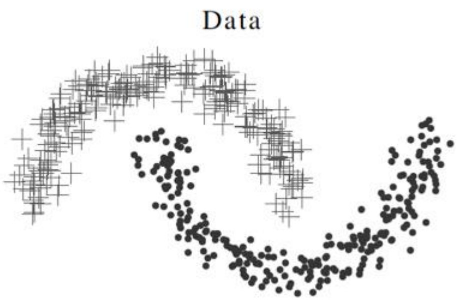
Laplacian Matrix

- **Begin with a similarity matrix,** $\mathbf{M} = \begin{bmatrix} \delta(x_1, x_1) & \cdots & \delta(x_1, x_n) \\ \vdots & \ddots & \vdots \\ \delta(x_n . x_1) & \cdots & \delta(x_n . x_n) \end{bmatrix}$
- The ϵ-neighborhood graph limits points for which $\delta(x_i, x_j)$ is calculated to require that $\delta(x_i, x_j) \leq \epsilon$.
- The k-nearest neighbor graph limits points corresponding to, as the name suggests, the k nearest neighbors of each point.
- These approaches lead to "reduced" similarity matrices, $\mathbf{M}_{\text{reduced}}$.
- The graph Laplacian begins with a diagonal "degree" matrix, $\boldsymbol{\Delta} = \begin{bmatrix} \deg(x_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \deg(x_n) \end{bmatrix}$, where $\deg(x_i) = \sum_j \delta(x_i, x_j)$.
- Then the graph Laplacian is $\mathbf{L} = \boldsymbol{\Delta} - \mathbf{M}_{\text{reduced}}$.

Un-normalized Spectral Clustering

1. Construct $\mathbf{M}_{\text{reduced}}$.
2. Construct $\mathbf{L} = \mathbf{\Delta} - \mathbf{M}_{\text{reduced}}$.
3. Find the first k non-zero eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ of $\mathbf{L}$.
4. Construct matrix $\mathbf{U}$ from $\mathbf{u}_1, \ldots, \mathbf{u}_k$.
5. Cluster the *rows* of $\mathbf{U}$ with k-means.
6. Return the row indices grouped by the clusters.

Example Clusters



Normalized (Sym) Spectral Clustering

Two normalized graph Laplacians

$$\mathbf{L}_{\text{sym}} = \mathbf{\Delta}^{-1/2}\mathbf{L}\mathbf{\Delta}^{-1/2} = \mathbf{I} - \mathbf{\Delta}^{-1/2}\mathbf{U}\mathbf{\Delta}^{-1/2}$$

$$\mathbf{L}_{\text{rw}} = \mathbf{\Delta}^{-1/2}\mathbf{L} = \mathbf{I} - \mathbf{\Delta}^{-1/2}\mathbf{U}$$

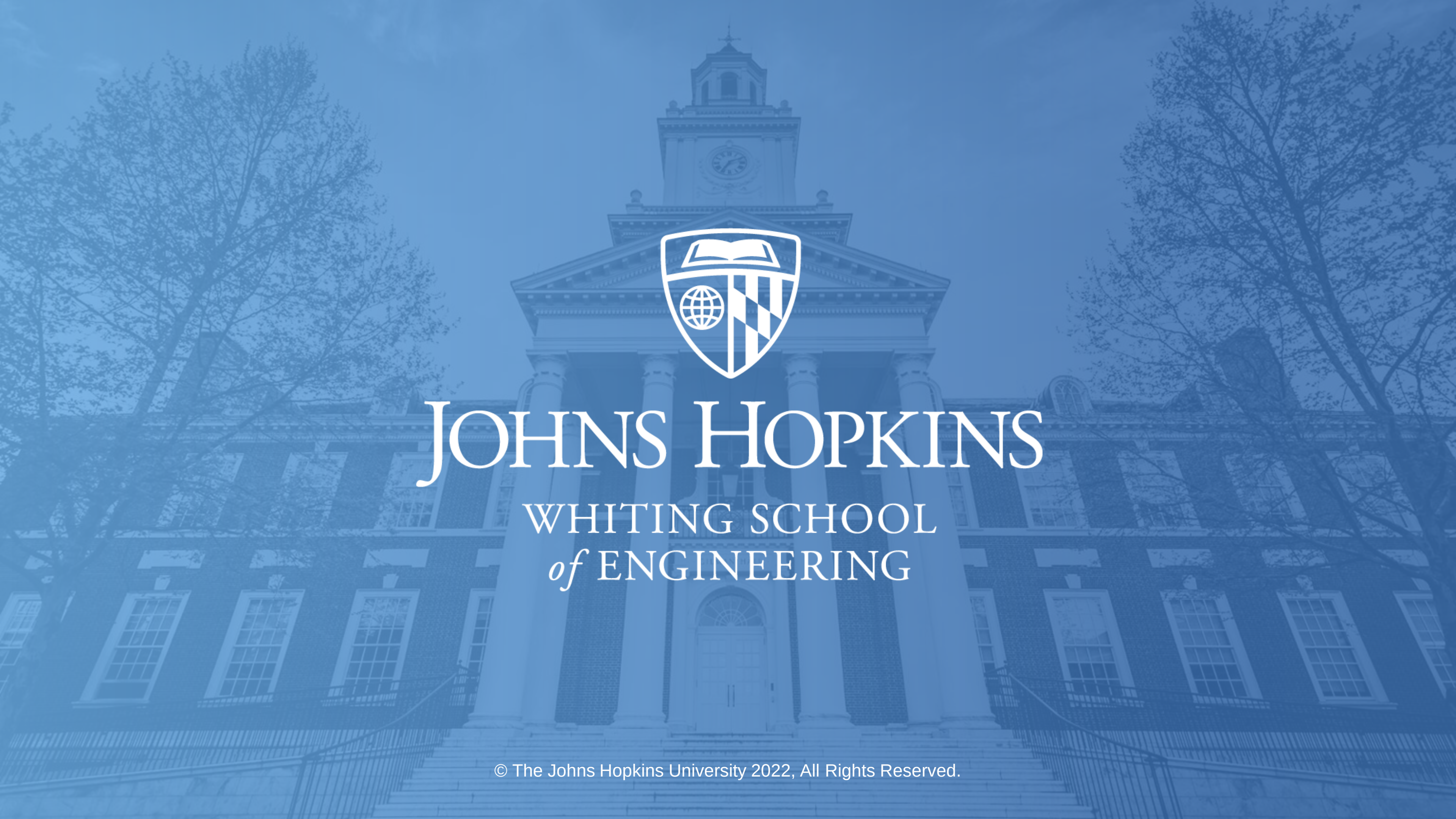
1. Construct $\mathbf{M}_{\text{reduced}}$.
2. Construct $\mathbf{L} = \mathbf{\Delta} - \mathbf{M}_{\text{reduced}}$.
3. Construct $\mathbf{L}_{\text{sym}} = \mathbf{\Delta}^{-1/2}\mathbf{L}\mathbf{\Delta}^{-1/2}$.
4. Find the first k non-zero eigenvectors $\mathbf{u}_1, \dots, \mathbf{u}_k$ of $\mathbf{L}_{\text{sym}}$.
5. Construct matrix $\mathbf{U}$ from $\mathbf{u}_1, \dots, \mathbf{u}_k$.
6. Normalize the rows of $\mathbf{U}$, $\forall i \leq n, \sum_j u_{ij}^2 = 1$.
7. Cluster the *rows* of $\mathbf{U}$ with k-means.
8. Return the row indices grouped by the clusters.

Normalized (RW) Spectral Clustering

Two normalized graph Laplacians

$$\mathbf{L}_{\mathrm{sym}} = \boldsymbol{\Delta}^{-1/2}\mathbf{L}\boldsymbol{\Delta}^{-1/2} = \mathbf{I} - \boldsymbol{\Delta}^{-1/2}\mathbf{U}\boldsymbol{\Delta}^{-1/2}$$
$$\mathbf{L}_{\mathrm{rw}} = \boldsymbol{\Delta}^{-1/2}\mathbf{L} = \mathbf{I} - \boldsymbol{\Delta}^{-1/2}\mathbf{U}$$

1. Construct $\mathbf{M}_{\mathrm{reduced}}$.
2. Construct $\mathbf{L} = \boldsymbol{\Delta} - \mathbf{M}_{\mathrm{reduced}}$.
3. Construct $\mathbf{L}_{\mathrm{rw}} = \boldsymbol{\Delta}^{-1/2}\mathbf{L}$.
4. Find the first k non-zero eigenvectors $\mathbf{u}_1, \dots, \mathbf{u}_k$ of $\mathbf{L}_{\mathrm{sym}}$.
5. Construct matrix $\mathbf{U}$ from $\mathbf{u}_1, \dots, \mathbf{u}_k$.
6. Cluster the *rows* of $\mathbf{U}$ with k-means.
7. Return the row indices grouped by the clusters.



JOHNS HOPKINS

WHITING SCHOOL of ENGINEERING

Introduction to Machine Learning

PAC Learning

Sample Complexity

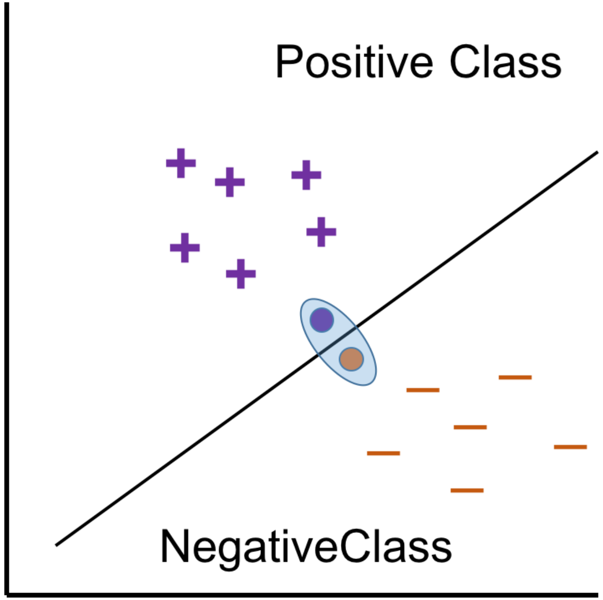
Computational Learning Theory (COLT)

1. What general laws constrain learning?
2. What types of learning problems can be solved in reasonable time/space?
3. When can we trust the output of a learned hypothesis, and by how much can be trust it?

Types of Supervised Learning

- Teacher-annotated learning
    - Probability distribution
    - Inductive learning principle/hypothesis
- Active learning
    - Examples → oracle
    - Oracle provides the label
- Helpful teacher
    - Teacher picks the examples
    - Principle to determine minimum number of examples needed

Learning with a Helpful Teacher



Back to Sample Complexity

- The sample complexity of a learning problem seeks to determine, for a concept,
  - The number of training examples needed to learn the concept
  - Ideally minimize the amount of training required under the associated learning model
- The notion of sample complexity was first posed by Leslie Valiant.

  L. G. Valiant, “A Theory of the Learnable,” *Communications of the ACM*, Volume 27, Issue 11, November 1984, pp. 1134–1142.

- **Def:** A concept *C* is said to be *PAC-Learnable* by learner *L* using hypothesis space *H* if, for all $c \in C$, distributions *D* over *X* of length *n*, ε such that $0 < \varepsilon < 0.5$, and δ such that $0 < \delta < 0.5$, learner *L* will output hypothesis $h \in H$ with probability at least $(1 - \delta)$ such that the true error $error(h) \leq \varepsilon$ in time polynomial in $\frac{1}{\varepsilon}, \frac{1}{\delta}, n$ and $size(c)$.

PAC Learning

- Term first coined by Dana Angluin.
- Many key theorems proven by David Haussler.

$$m \geq \frac{1}{\epsilon}\left(\ln|H| + \ln\frac{1}{\delta}\right)$$

- Example

$$y = f(x_1, \ldots, x_n) = x'_1 \wedge \cdots \wedge x'_k,$$

  - Active learner: n examples
  - Helpful teacher: k examples
  - Teacher annotated: $\frac{1}{\epsilon}\left(\ln n + \ln\frac{1}{\delta}\right)$ examples

JOHNS HOPKINS

WHITING SCHOOL of ENGINEERING