Primal-Dual Algorithms for Clustering and Feature Allocation

Nathan Cordner

Boston University

1 October 2018

Introduction

Clustering Problem

Introduction

Clustering Problem

Introduction

Clustering Problem

Introduction

Clustering Problem

Introduction

Feature Allocation Problem

Introduction

Feature Allocation Problem

Introduction

Feature Allocation Problem

Introduction

Feature Allocation Problem

(Metric) Uncapacitated Facility Location

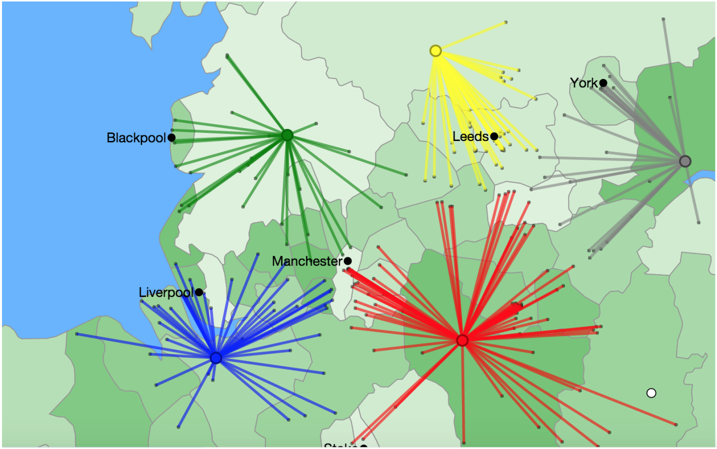


Figure: http://examples.gurobi.com/facility-location

(Metric) Uncapacitated Facility Location

Given a set of facilities F, and a set of clients C

(Metric) Uncapacitated Facility Location

Given a set of facilities F, and a set of clients C

- $c_{ij} =$ distance between facility i and client j

(Metric) Uncapacitated Facility Location

Given a set of facilities F, and a set of clients C

- $c_{ij} =$ distance between facility i and client j
- $f_i =$ cost of opening facility i

(Metric) Uncapacitated Facility Location

Given a set of facilities F, and a set of clients C

- c_{ij} = distance between facility i and client j
- f_i = cost of opening facility i
- x_{ij} = indicator for whether client j connects to facility i

(Metric) Uncapacitated Facility Location

Given a set of facilities F, and a set of clients C

- c_{ij} = distance between facility i and client j
- f_i = cost of opening facility i
- x_{ij} = indicator for whether client j connects to facility i
- y_i = indicator for whether facility i is open

(Metric) Uncapacitated Facility Location

Primal IP:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i \in F, j \in C} c_{ij} x_{ij} + \sum_{i \in F} f_i y_i \\
\text{subject to} \quad & \forall j \in C : \sum_{i \in F} x_{ij} \geq 1, \\
& \forall i \in F, j \in C : y_i - x_{ij} \geq 0, \\
& \forall i \in F, j \in C : x_{ij} \in \{0, 1\}, \\
& \forall i \in F : y_i \in \{0, 1\}.
\end{aligned}
$$

c_{ij} = distance, f_i = facility cost,
x_{ij} = client connection, y_i = facility open

(Metric) Uncapacitated Facility Location

Primal LP Relaxation:

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i \in F, j \in C} c_{ij} x_{ij} + \sum_{i \in F} f_i y_i \\ \text{subject to} & \forall j \in C : \displaystyle\sum_{i \in F} x_{ij} \geq 1, \\ & \forall i \in F, j \in C : y_i - x_{ij} \geq 0, \\ & \forall i \in F, j \in C : x_{ij} \geq 0, \\ & \forall i \in F : y_i \geq 0. \end{array}$$

c_{ij} = distance, f_i = facility cost,
x_{ij} = client connection, y_i = facility open

(Metric) Uncapacitated Facility Location

Dual LP:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{j \in C} \alpha_j \\
\text{subject to} \quad & \forall i \in F, j \in C : \alpha_j - \beta_{ij} \leq c_{ij}, \\
& \forall i \in F : \sum_{j \in C} \beta_{ij} \leq f_i, \\
& \forall j \in C : \alpha_j \geq 0, \\
& \forall i \in F, j \in C : \beta_{ij} \geq 0.
\end{aligned}
$$

(Metric) Uncapacitated Facility Location

Primal-Dual Approach [JV]: increase dual variables until constraints become tight

(Metric) Uncapacitated Facility Location

Primal-Dual Approach [JV]: increase dual variables until constraints become tight

- c_{ij} = cost of client j to use facility i

(Metric) Uncapacitated Facility Location

Primal-Dual Approach [JV]: increase dual variables until constraints become tight

- c_{ij} = cost of client j to use facility i
- β_{ij} = contribution of client j to opening facility i

(Metric) Uncapacitated Facility Location

Primal-Dual Approach [JV]: increase dual variables until constraints become tight

- $c_{ij} =$ cost of client j to use facility i
- $\beta_{ij} =$ contribution of client j to opening facility i

$\phi(j) = i$ denotes that client j is *connected* to facility i

(Metric) Uncapacitated Facility Location

Primal-Dual Approach [JV]: increase dual variables until constraints become tight

- c_{ij} = cost of client j to use facility i
- β_{ij} = contribution of client j to opening facility i

$\phi(j) = i$ denotes that client j is *connected* to facility i

Facility i is *paid for* when $\sum_{j:\phi(j)=i} \beta_{ij} = f_i$

(Metric) Uncapacitated Facility Location

Primal-Dual Approach [JV]: increase dual variables until constraints become tight

- c_{ij} = cost of client j to use facility i
- β_{ij} = contribution of client j to opening facility i

$\phi(j) = i$ denotes that client j is *connected* to facility i

Facility i is *paid for* when $\sum_{j:\phi(j)=i} \beta_{ij} = f_i$

Total price paid by client j: $\alpha_j = \beta_{\phi(j)j} + c_{\phi(j)j}$

(Metric) Uncapacitated Facility Location

Algorithm Outline: sort list of edges in increasing order

(Metric) Uncapacitated Facility Location

Algorithm Outline: sort list of edges in increasing order

If an edge (i, j) goes tight:

(Metric) Uncapacitated Facility Location

Algorithm Outline: sort list of edges in increasing order

If an edge (i, j) goes tight:

- If i is not paid for, then it gets one more contributor

(Metric) Uncapacitated Facility Location

Algorithm Outline: sort list of edges in increasing order

If an edge (i, j) goes tight:

- If i is not paid for, then it gets one more contributor
- Else, j connects to i and j is removed as a contributor to all other facilities

(Metric) Uncapacitated Facility Location

Algorithm Outline: sort list of edges in increasing order

If an edge (i, j) goes tight:

- If i is not paid for, then it gets one more contributor
- Else, j connects to i and j is removed as a contributor to all other facilities

If a facility is paid for:

(Metric) Uncapacitated Facility Location

Algorithm Outline: sort list of edges in increasing order

If an edge (i, j) goes tight:

- If i is not paid for, then it gets one more contributor
- Else, j connects to i and j is removed as a contributor to all other facilities

If a facility is paid for:

- Each contributing client is now declared connected and removed as contributor to all other facilities

(Metric) Uncapacitated Facility Location

Algorithm Outline: sort list of edges in increasing order

If an edge (i, j) goes tight:

- If i is not paid for, then it gets one more contributor
- Else, j connects to i and j is removed as a contributor to all other facilities

If a facility is paid for:

- Each contributing client is now declared connected and removed as contributor to all other facilities

Finish when all clients are connected

(Metric) Uncapacitated Facility Location

Running Time: $O(m \log m)$, where $m = |F| \cdot |C|$

(Metric) Uncapacitated Facility Location

Running Time: $O(m \log m)$, where $m = |F| \cdot |C|$

Approximation Bound:

$$\sum_{i \in F, j \in C} c_{ij} x_{ij} + \sum_{i \in F} f_i y_i \leq 3 \cdot \mathsf{OPT}$$

Primal-Dual Clustering

Now let $C = F$, and each $f_i = \lambda$

Primal-Dual Clustering

Now let $C = F$, and each $f_i = \lambda$

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i,j \in C} c_{ij} x_{ij} + \lambda \sum_{i \in C} y_i \\
\text{subject to} \quad & \forall j \in C : \sum_{i \in C} x_{ij} \geq 1, \\
& \forall i, j \in C : y_i - x_{ij} \geq 0, \\
& \forall i, j \in C : x_{ij} \in \{0, 1\}, \\
& \forall i \in C : y_i \in \{0, 1\}.
\end{aligned}
$$

Primal-Dual Clustering

Now let $C = F$, and each $f_i = \lambda$

$$\begin{aligned} \text{minimize} \quad & \sum_{i,j \in C} c_{ij} x_{ij} + \lambda \sum_{i \in C} y_i \\ \text{subject to} \quad & \forall j \in C : \sum_{i \in C} x_{ij} \geq 1, \\ & \forall i, j \in C : y_i - x_{ij} \geq 0, \\ & \forall i, j \in C : x_{ij} \in \{0, 1\}, \\ & \forall i \in C : y_i \in \{0, 1\}. \end{aligned}$$

LP relaxation and dual programs are similar

Primal-Dual Clustering

Algorithm is just Facility Location in the special case

Primal-Dual Clustering

Algorithm is just Facility Location in the special case

- At worst a 3-approximation algorithm

Primal-Dual Clustering

Algorithm is just Facility Location in the special case

- At worst a 3-approximation algorithm
- As λ gets large, results converge to OPT

Primal-Dual Clustering

Algorithm is just Facility Location in the special case

- At worst a 3-approximation algorithm
- As λ gets large, results converge to OPT

Running time is $O(n^2 \log n)$ where $n = |C|$

Primal-Dual Clustering

Algorithm is just Facility Location in the special case

- At worst a 3-approximation algorithm
- As λ gets large, results converge to OPT

Running time is $O(n^2 \log n)$ where $n = |C|$

- Can run quicker for smaller values of λ

Primal-Dual Clustering

Algorithm is just Facility Location in the special case

- At worst a 3-approximation algorithm
- As λ gets large, results converge to OPT

Running time is $O(n^2 \log n)$ where $n = |C|$

- Can run quicker for smaller values of λ

Compare with K-means

Primal-Dual Clustering

Algorithm is just Facility Location in the special case

- At worst a 3-approximation algorithm
- As λ gets large, results converge to OPT

Running time is $O(n^2 \log n)$ where $n = |C|$

- Can run quicker for smaller values of λ

Compare with K-means

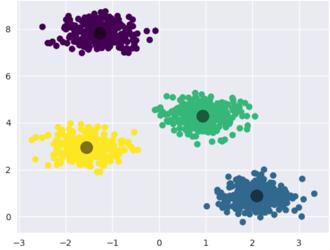
- PD approach takes a little longer, but can give better results

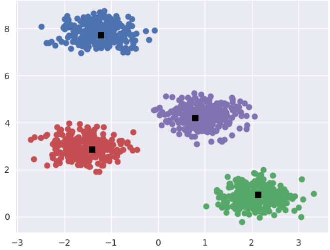
Primal-Dual Clustering

Examples: 1200 points

Primal-Dual Clustering

Examples: 1200 points

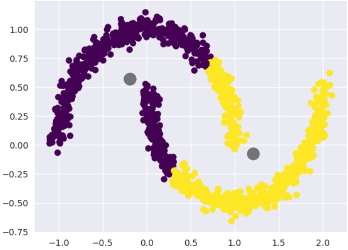


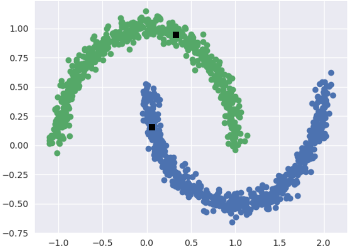


K-means: 0.06 s, $k = 4$ / PD: 0.77 s, $\lambda = 7$

Primal-Dual Clustering

Examples: 1200 points

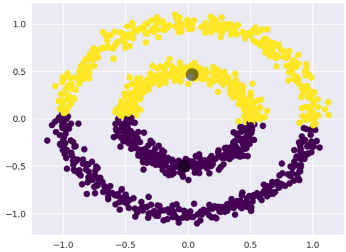


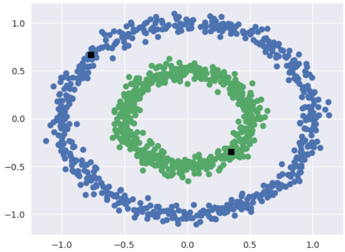


K-means: 0.04 s, $k = 2$ / PD: 0.59 s, $\lambda = 3$

Primal-Dual Clustering

Examples: 1200 points

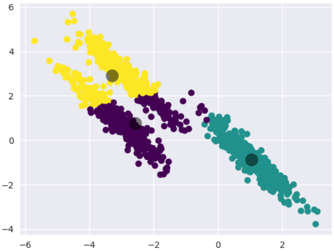


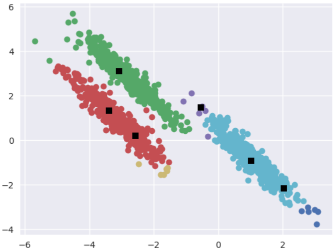


K-means: 0.04 s, $k = 2$ / PD: 0.54 s, $\lambda = 2$

Primal-Dual Clustering

Examples: 1200 points

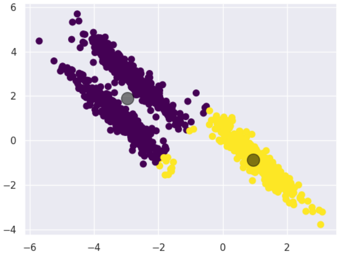


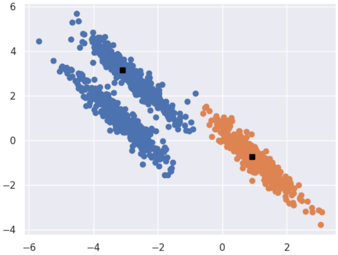


K-means: 0.04 s, $k = 3$ / PD: 0.83 s, $\lambda = 5$

Primal-Dual Clustering

Examples: 1200 points





K-means: 0.12 s, $k = 2$ / PD: 1.15 s, $\lambda = 10$

Primal-Dual Feature Allocation

Similar to clustering, with relaxed constraint for x_{ij}

Primal-Dual Feature Allocation

Similar to clustering, with relaxed constraint for x_{ij}

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i,j \in C} c_{ij} x_{ij} + \lambda \sum_{i \in C} y_i \\
\text{subject to} \quad & \forall j \in C : \sum_{i \in C} x_{ij} \geq 1, \\
& \forall i, j \in C : y_i - x_{ij} \geq 0, \\
& \forall i, j \in C : x_{ij} \geq 0, \\
& \forall i \in C : y_i \in \{0, 1\}.
\end{aligned}
$$

Primal-Dual Feature Allocation

Similar to clustering, with relaxed constraint for x_{ij}

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i,j \in C} c_{ij} x_{ij} + \lambda \sum_{i \in C} y_i \\
\text{subject to} \quad & \forall j \in C : \sum_{i \in C} x_{ij} \geq 1, \\
& \forall i, j \in C : y_i - x_{ij} \geq 0, \\
& \forall i, j \in C : x_{ij} \geq 0, \\
& \forall i \in C : y_i \in \{0, 1\}.
\end{aligned}
$$

LP relaxation and dual programs are the same

Primal-Dual Feature Allocation

Algorithm Outline:

Primal-Dual Feature Allocation

Algorithm Outline:

- same as clustering, except we do not remove any contributions from clients when they connect to facilities

Primal-Dual Feature Allocation

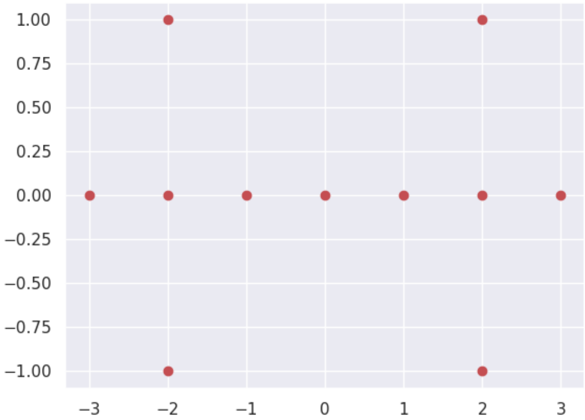
Algorithm Outline:

- same as clustering, except we do not remove any contributions from clients when they connect to facilities

A client may contribute to more than one facility, and thus have multiple features

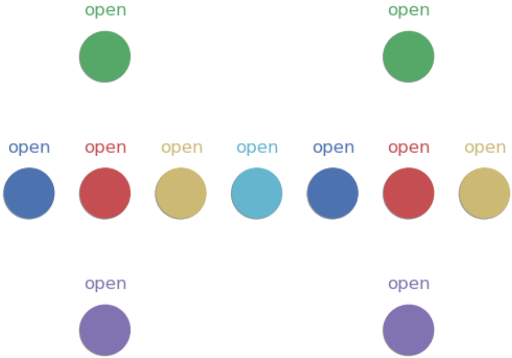
Primal-Dual Feature Allocation

Example:



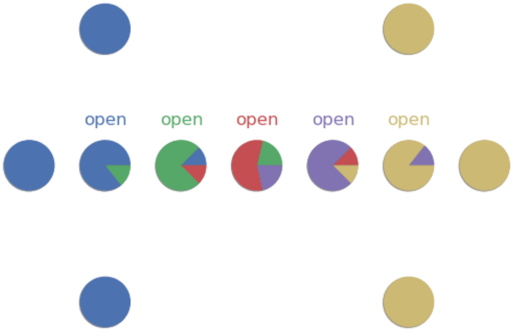
Primal-Dual Feature Allocation

Example: $\lambda = 1$



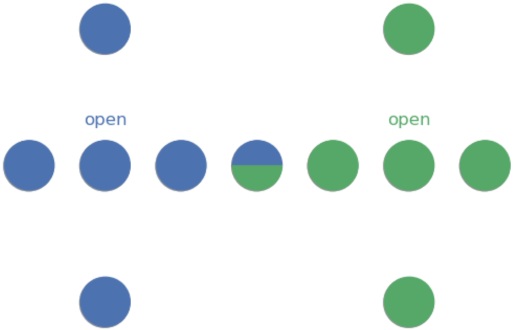
Primal-Dual Feature Allocation

Example: $\lambda = 2$



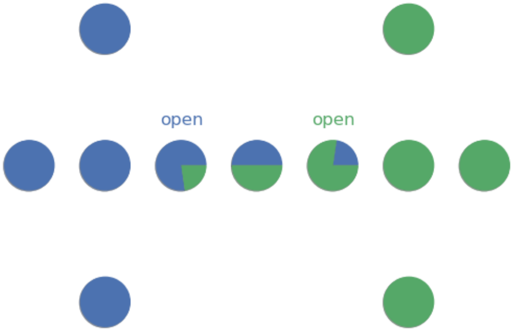
Primal-Dual Feature Allocation

Example: $\lambda = 7$



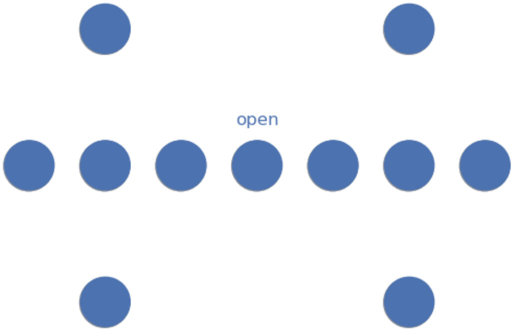
Primal-Dual Feature Allocation

Example: $\lambda = 11$



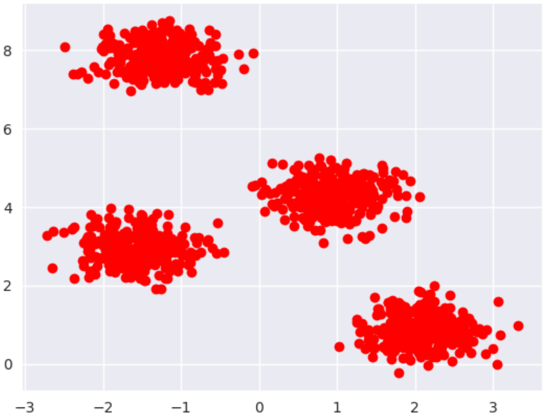
Primal-Dual Feature Allocation

Example: $\lambda = 15$



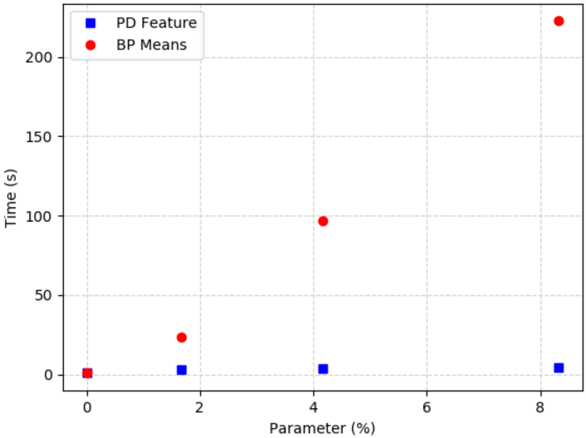
Primal-Dual Feature Allocation

Compare with BP Means [BKJ]:



Primal-Dual Feature Allocation

Compare with BP Means [BKJ]:



Primal-Dual Feature Allocation

Analysis:

Primal-Dual Feature Allocation

Analysis:

- Theoretical run time is still $O(n^2 \log n)$, though can run quicker for smaller values of λ

Primal-Dual Feature Allocation

Analysis:

- Theoretical run time is still $O(n^2 \log n)$, though can run quicker for smaller values of λ
- As λ gets large, the algorithm's result equals OPT

Primal-Dual Feature Allocation

Analysis:

- Theoretical run time is still $O(n^2 \log n)$, though can run quicker for smaller values of λ
- As λ gets large, the algorithm's result equals OPT
- Worst case approximation bound: ongoing work

Leaving Out λ

We require $\lambda \geq 0$

Leaving Out λ

We require $\lambda \geq 0$

Theorem: The smallest value of λ that allows all edges to go tight is $\lambda^* = \max_i \left(n \cdot \max_j (c_{ij}) - \sum_j c_{ij} \right)$

Leaving Out λ

We require $\lambda \geq 0$

Theorem: The smallest value of λ that allows all edges to go tight is $\lambda^* = \max_i \left(n \cdot \max_j (c_{ij}) - \sum_j c_{ij} \right)$

Theorem: For $\lambda \geq \lambda_* = \min_i \left(n \cdot \max_j (c_{ij}) - \sum_j c_{ij} \right)$, OPT $= \min_i \left(\sum_j c_{ij} \right) + \lambda$

Leaving Out λ

Running PD algorithms for multiple values of λ:

Leaving Out λ

Running PD algorithms for multiple values of λ:

- Before the first facility gets paid for, the algorithm's computations are the same for each value of λ

Leaving Out λ

Running PD algorithms for multiple values of λ:

- Before the first facility gets paid for, the algorithm's computations are the same for each value of λ
- Branch computations for a particular λ when it reaches its first paid facility event

Leaving Out λ

Running PD algorithms for multiple values of λ:

- Before the first facility gets paid for, the algorithm's computations are the same for each value of λ
- Branch computations for a particular λ when it reaches its first paid facility event

Running PD algorithms without first choosing λ:

Leaving Out λ

Running PD algorithms for multiple values of λ:

- Before the first facility gets paid for, the algorithm's computations are the same for each value of λ
- Branch computations for a particular λ when it reaches its first paid facility event

Running PD algorithms without first choosing λ:

- Algorithm chooses values of $\lambda \in [0, \lambda^*]$ to test

Leaving Out λ

Same strategies work for running both clustering and feature allocation together

Leaving Out λ

Same strategies work for running both clustering and feature allocation together

- Keep the best looking result?

Leaving Out λ

Same strategies work for running both clustering and feature allocation together

- Keep the best looking result?
- Run clustering and feature allocation together, without λ!