Data Driven Resource Allocation for Distributed Machine Learning

Venkata Krishna Pillutla
www.cs.cmu.edu/~vpillutl

Thesis Committee

- Nina Balcan, Chair
- Alex Smola
- Christos Faloutsos

Collaborators





Machine Learning is Changing the World

"A breakthrough in machine learning would be worth ten Microsofts"
(Bill Gates, Chairman, Microsoft)

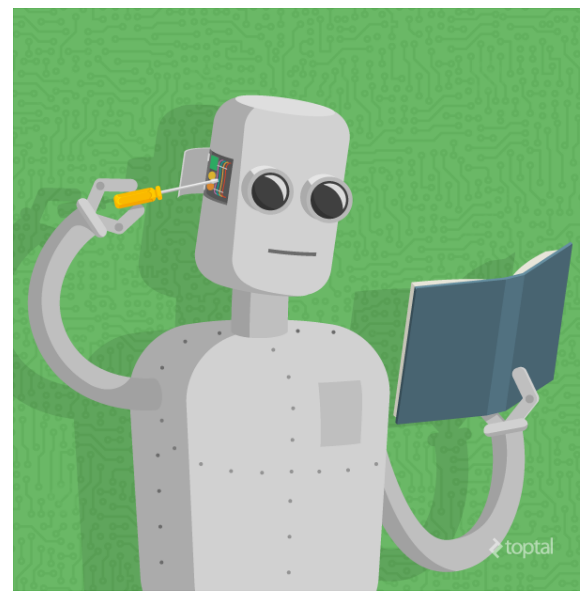
"Machine learning is the next Internet"
(Tony Tether, former director, DARPA)

"Machine learning is the hot new thing"
(John Hennessy, President, Stanford)

The World is Changing ML

Outbreak of the “Data Epidemic”

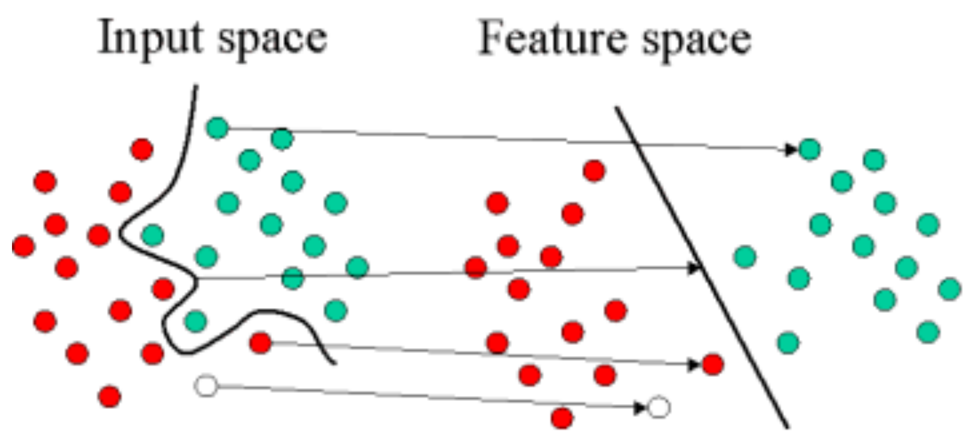
New Applications



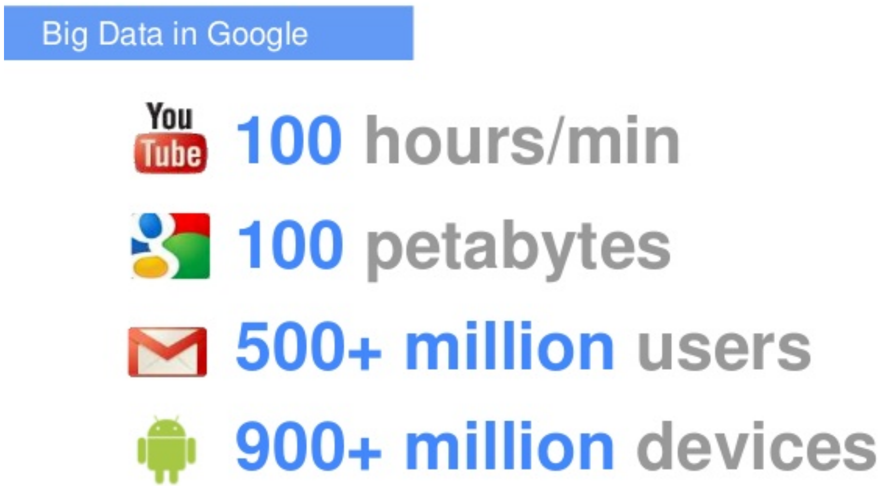
Introduction/Motivation

Machine Learning

- Traditional ML is centralized
- All the data is assumed to be on one machine



Distributed ML



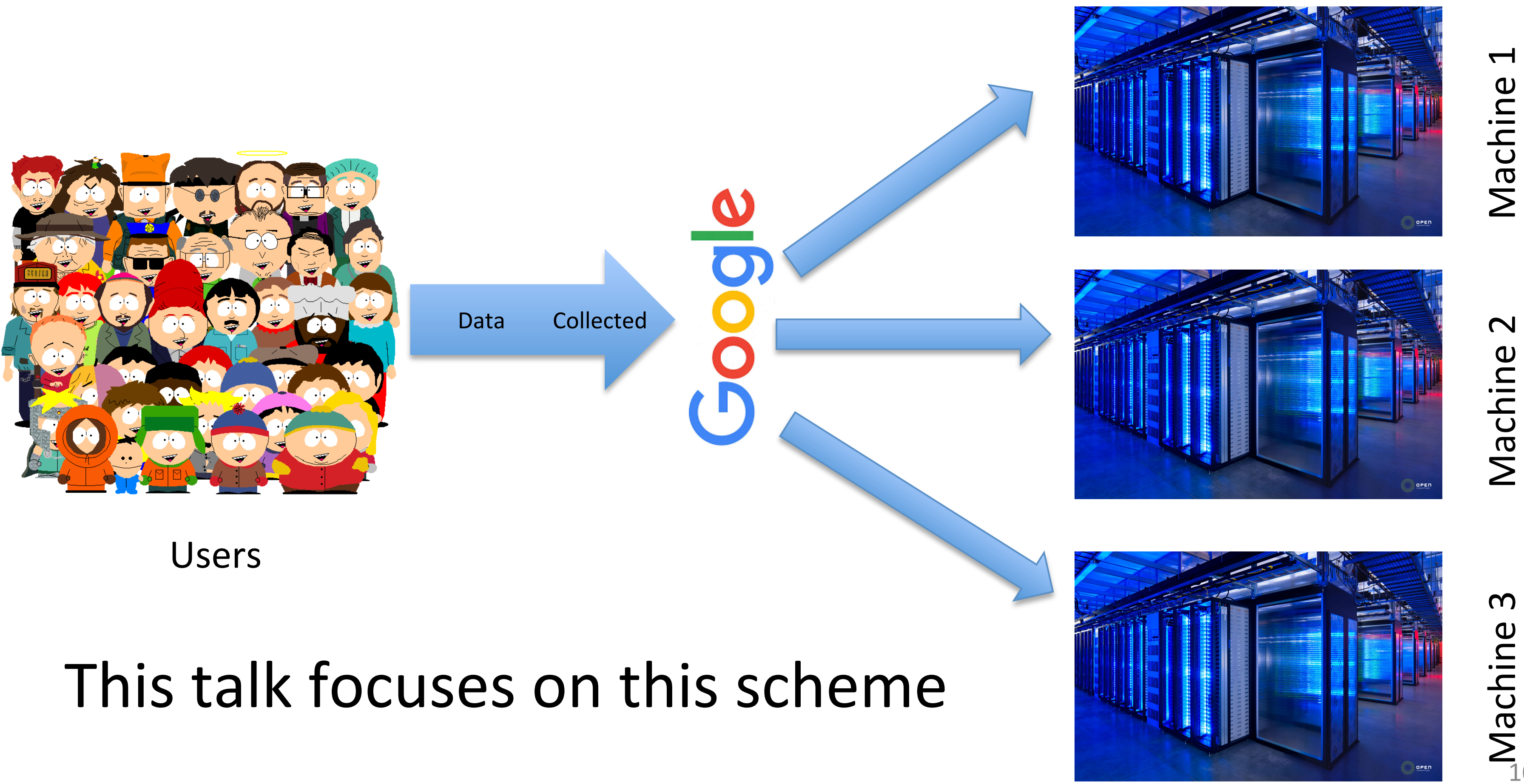
Distributed ML

Massive data is inherently distributed!

Also stored in a distributed manner. Eg: Yahoo! PNUTS

Distributed ML

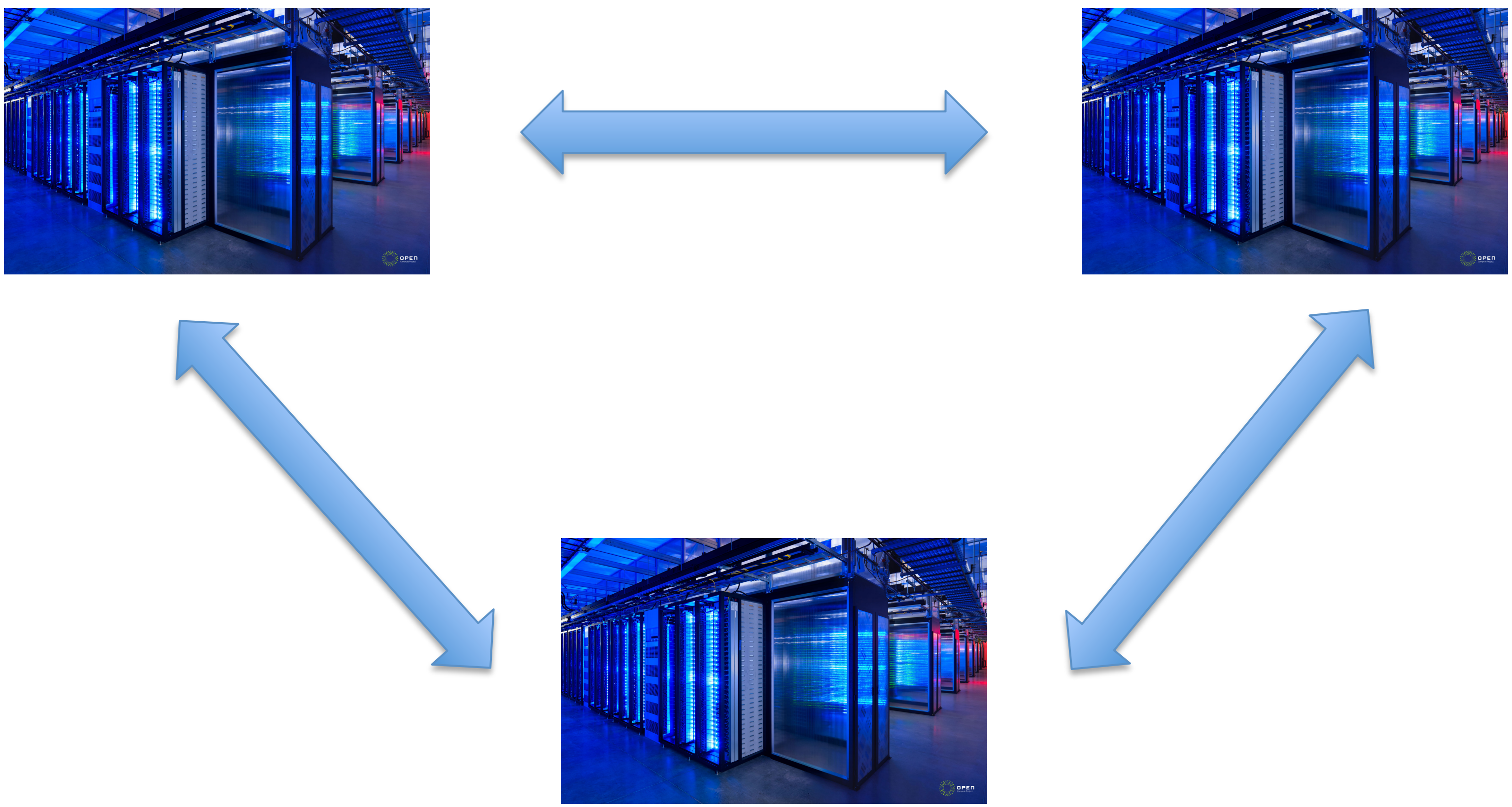
In other cases, massive data centrally collected



This talk focuses on this scheme

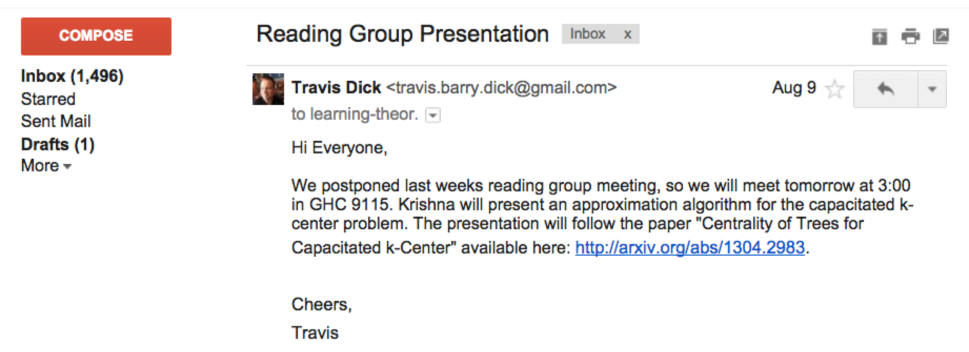
Distributed ML

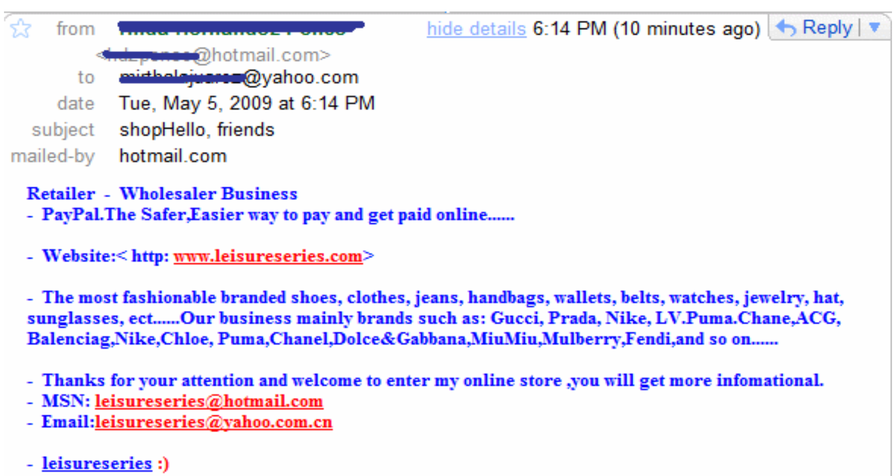
Communication: important resource (in addition to computation)



Typical Example: Learning Task

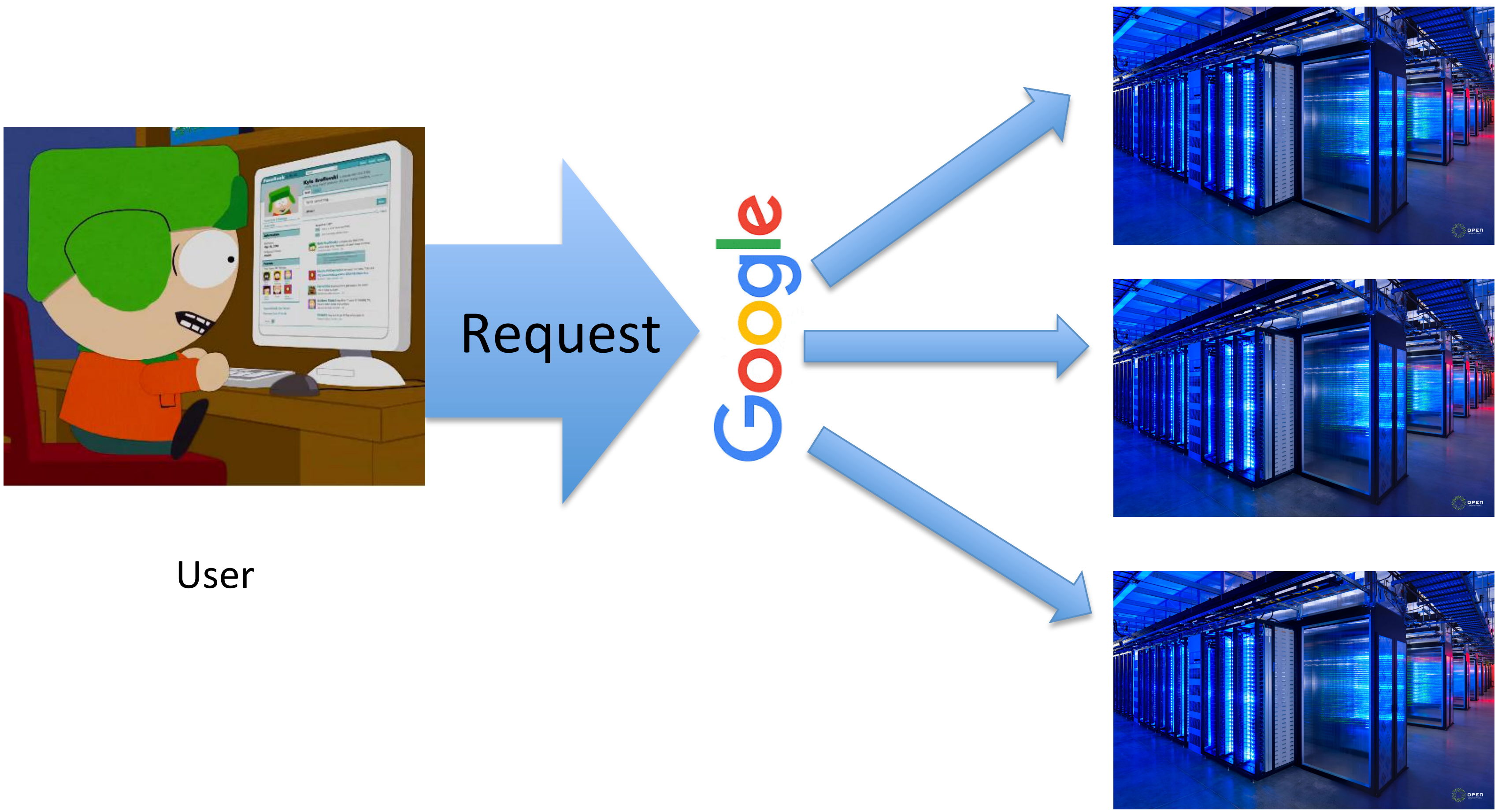
- Spam vs Not Spam





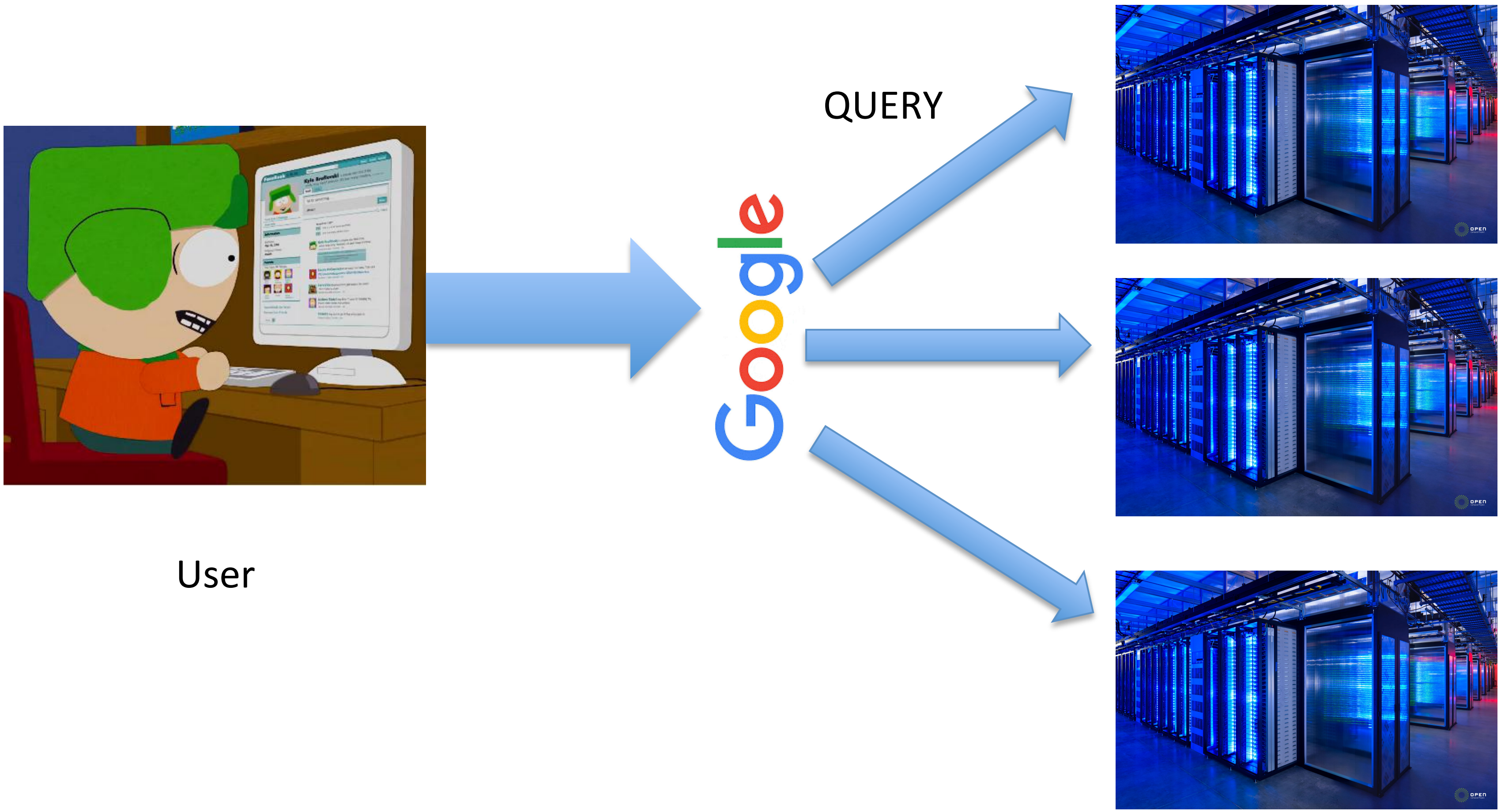
Another Example: Learning Task

Online advertising: Speed is of essence



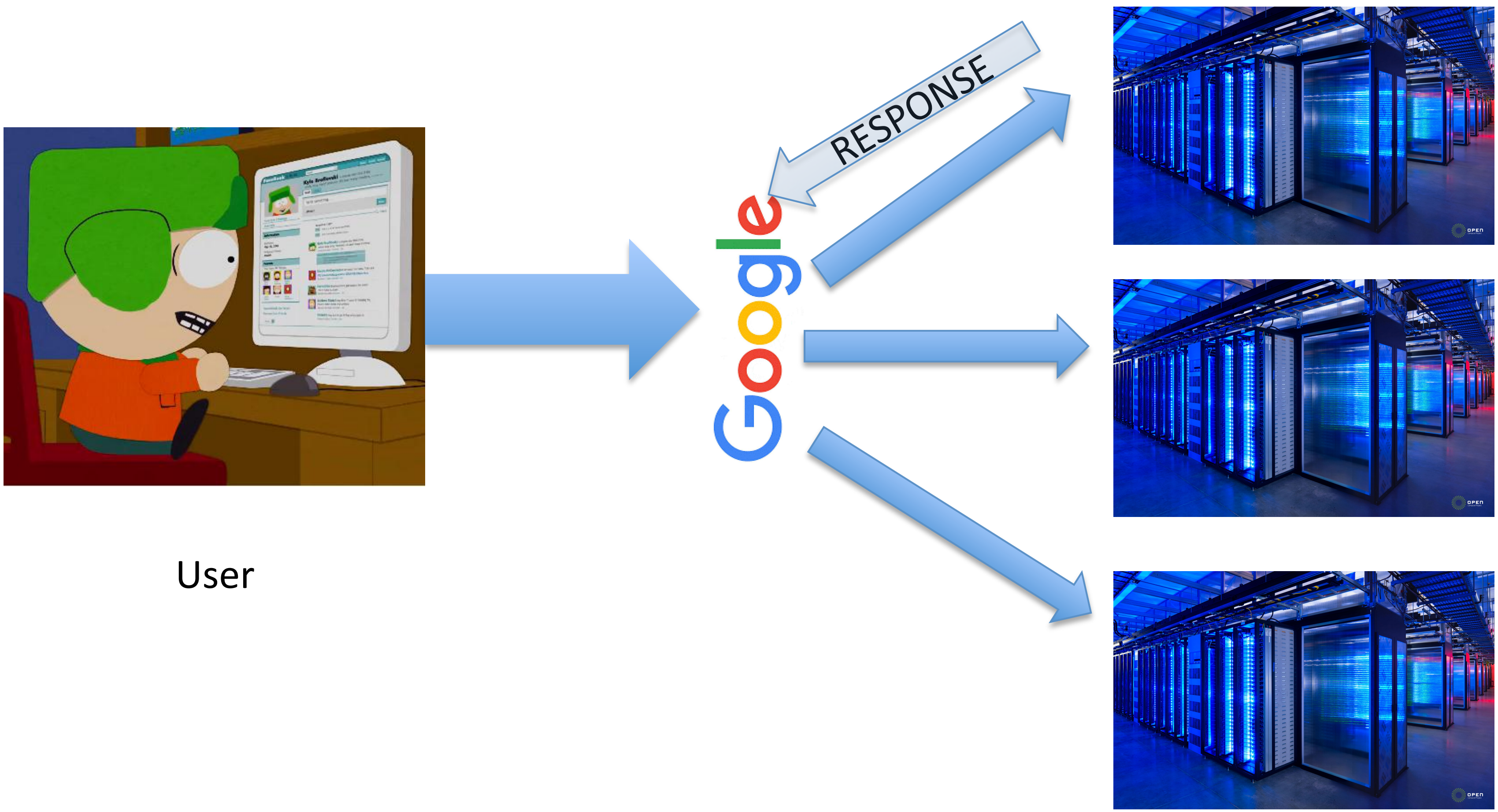
Another Example: Learning Task

Online advertising: Speed is of essence



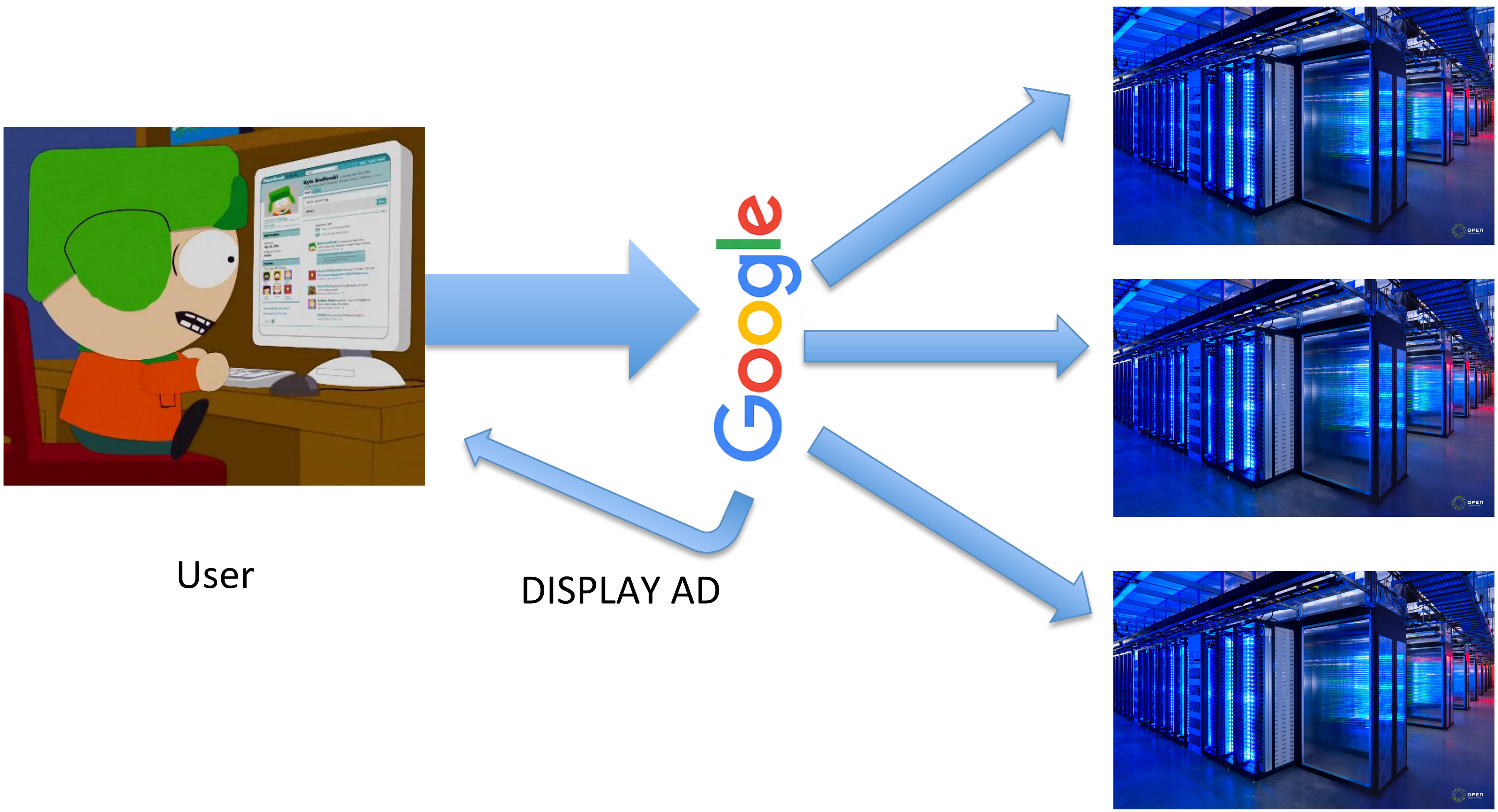
Another Example: Learning Task

Online advertising: Speed is of essence

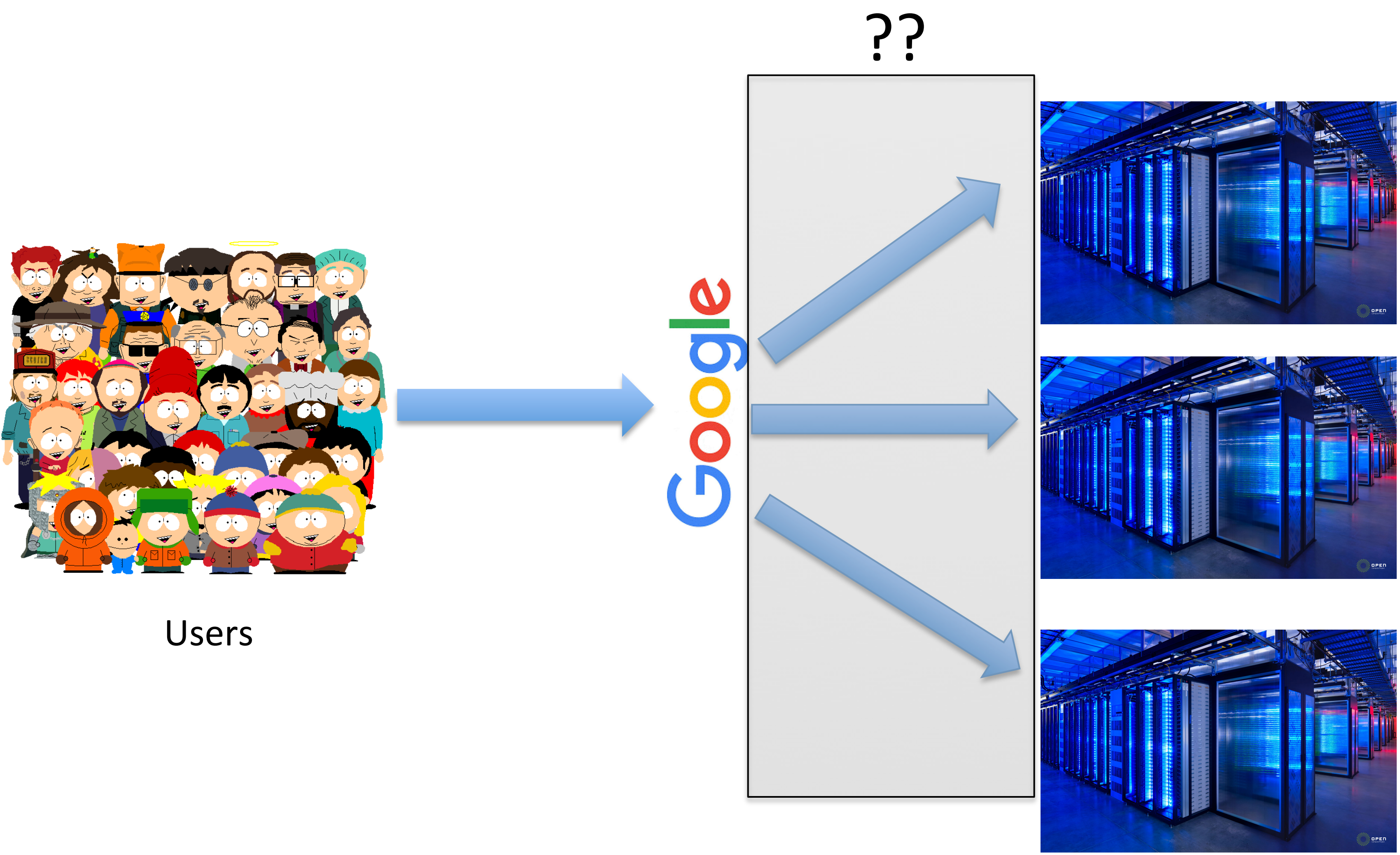


Another Example: Learning Task

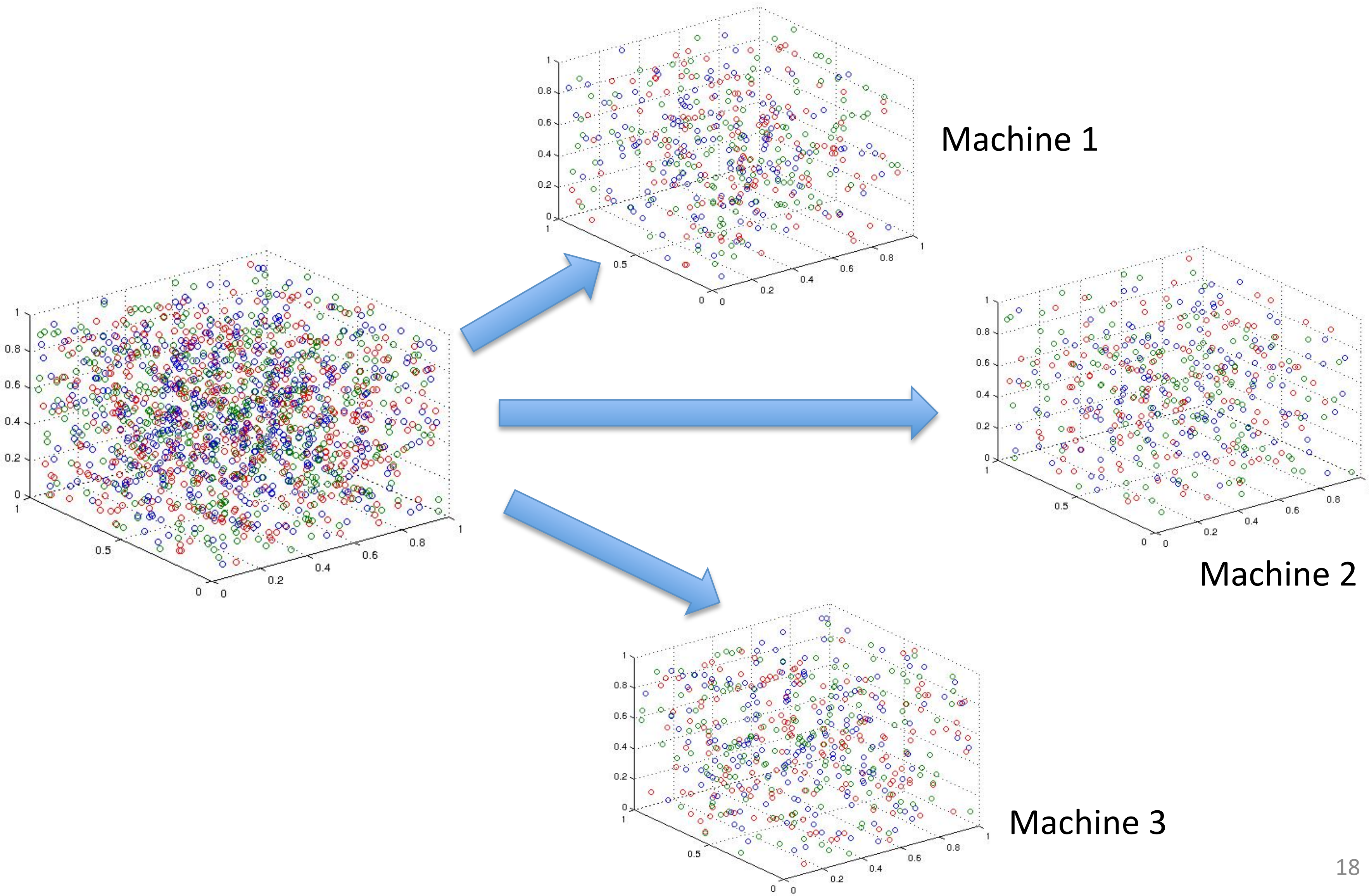
Online advertising: Speed is of essence



How to partition the data?

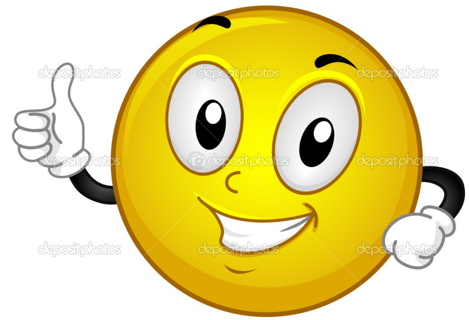


Random Partitioning



Random Partitioning

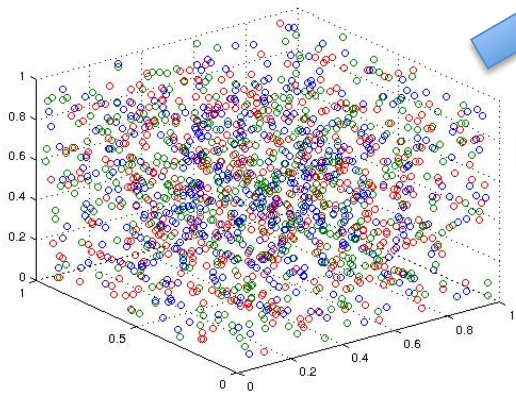
- Advantages
  - Easy to implement
  - Clean theory

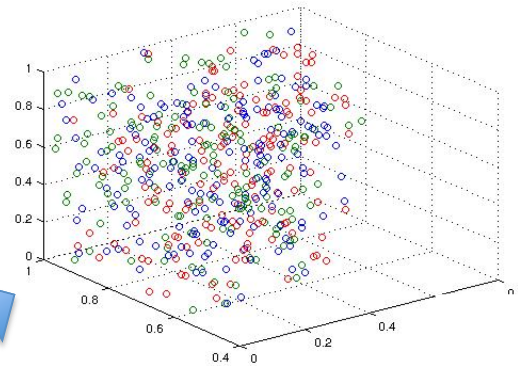


- Disadvantages
  - Statistically sub-optimal

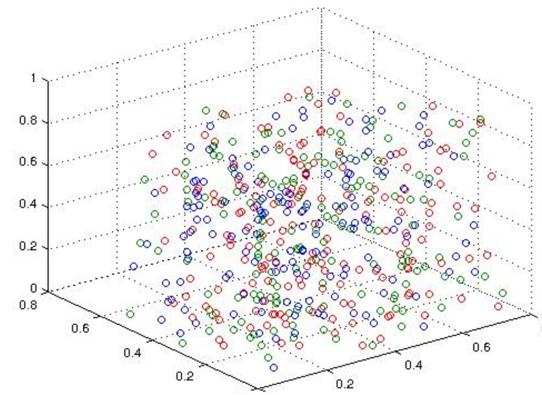
- Can we do better?

Our idea: Data dependent partitioning

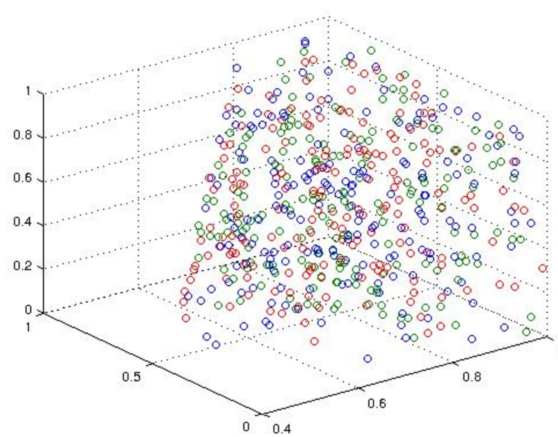




Machine 1



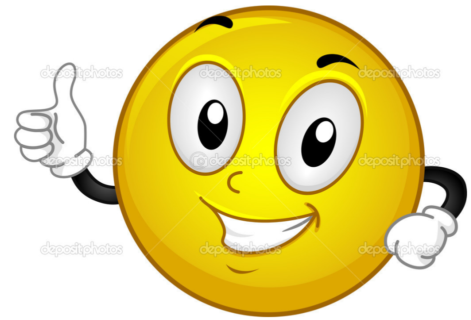
Machine 2



Machine 3

Vapnik:
Locally Simple but
Globally Complex!

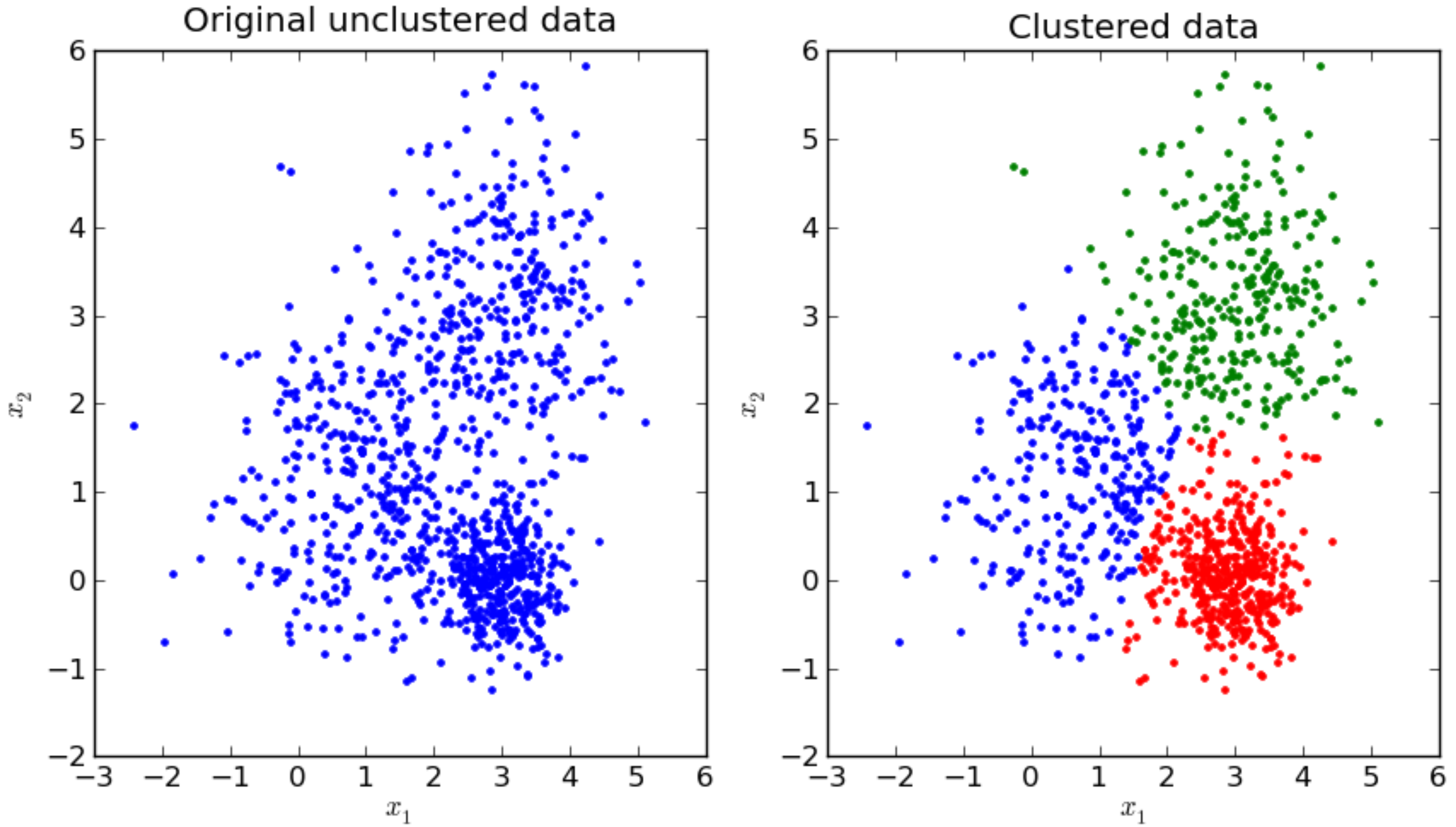
Pros and Cons



- Advantages
  - Distributed
  - More expressive concept class!
  - Better performance at same communication
- Possible Concern
  - More expressive dispatch rule is required ✓

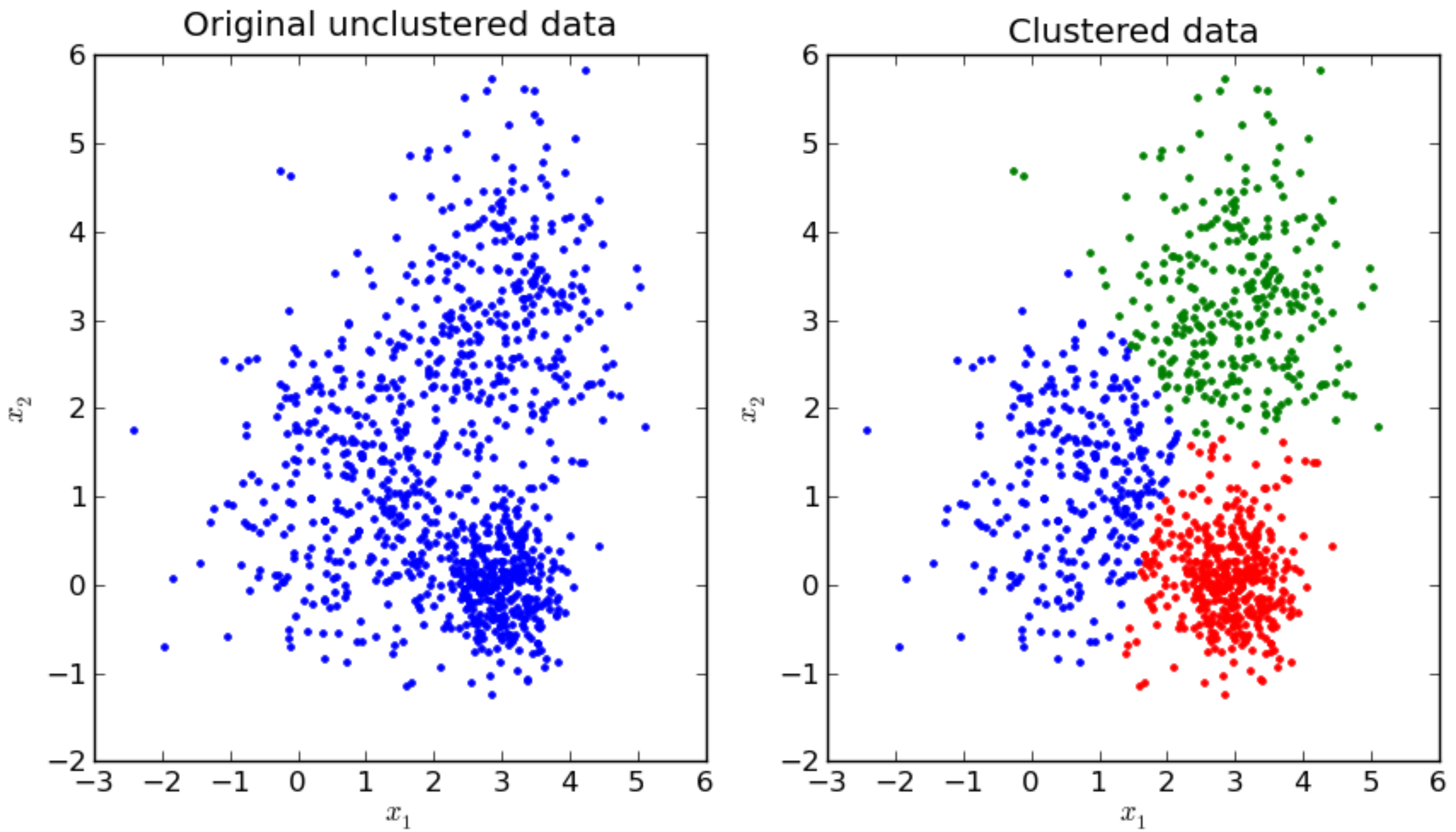
Data Dependent Partitioning

- How? Clustering



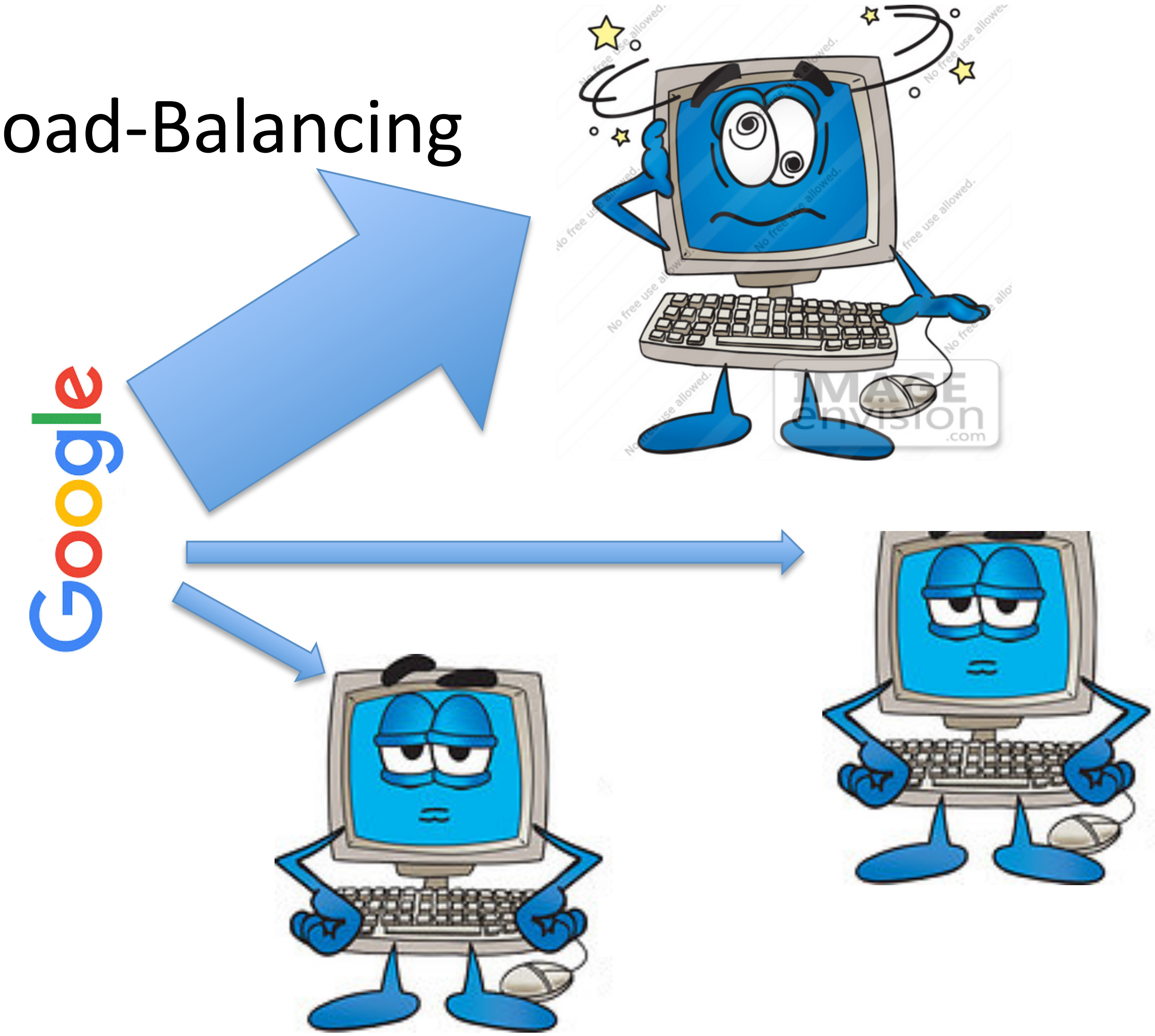
Data Dependent Partitioning

- For efficiency, cluster an initial sample



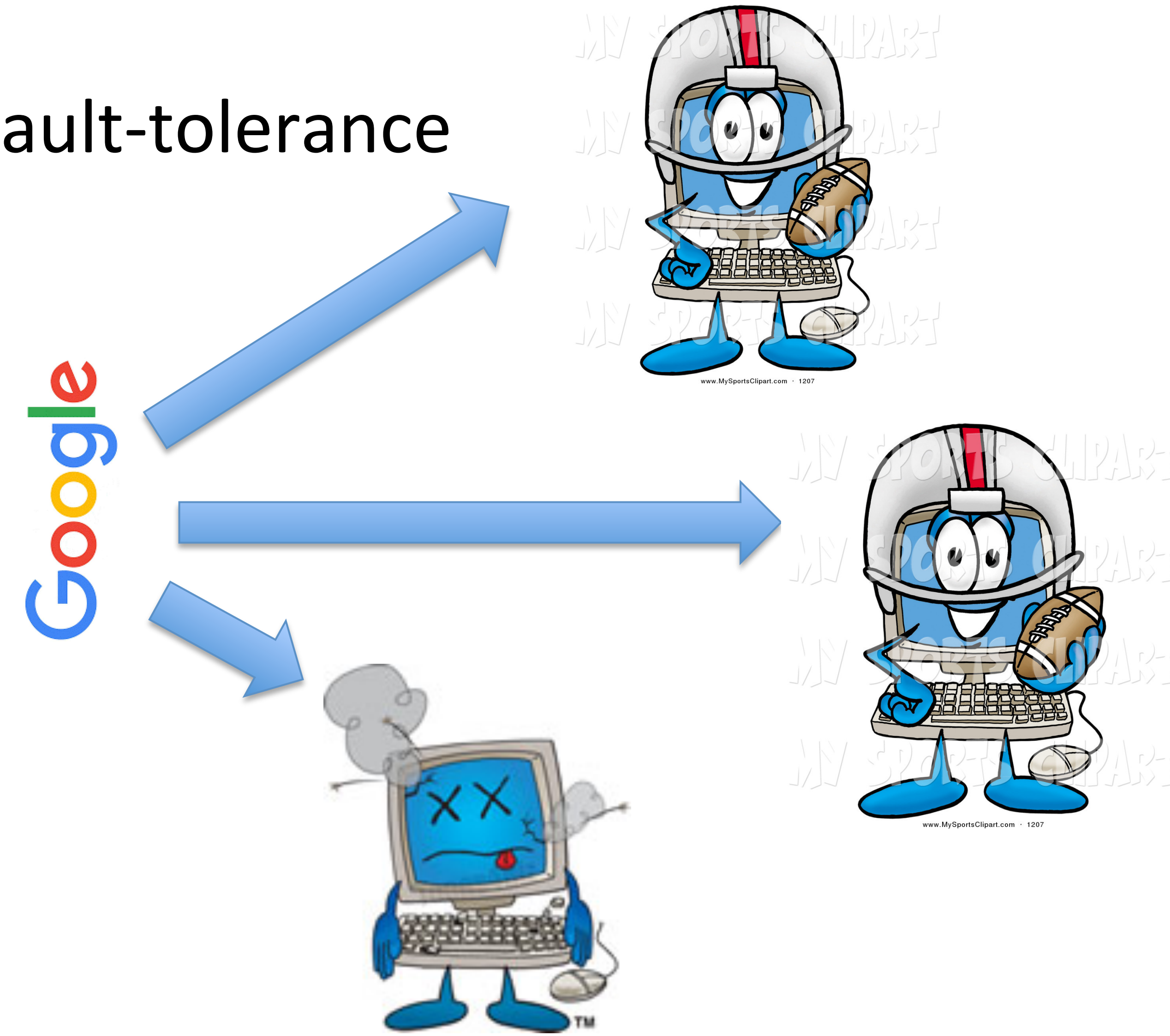
Requirements I

- Load-Balancing



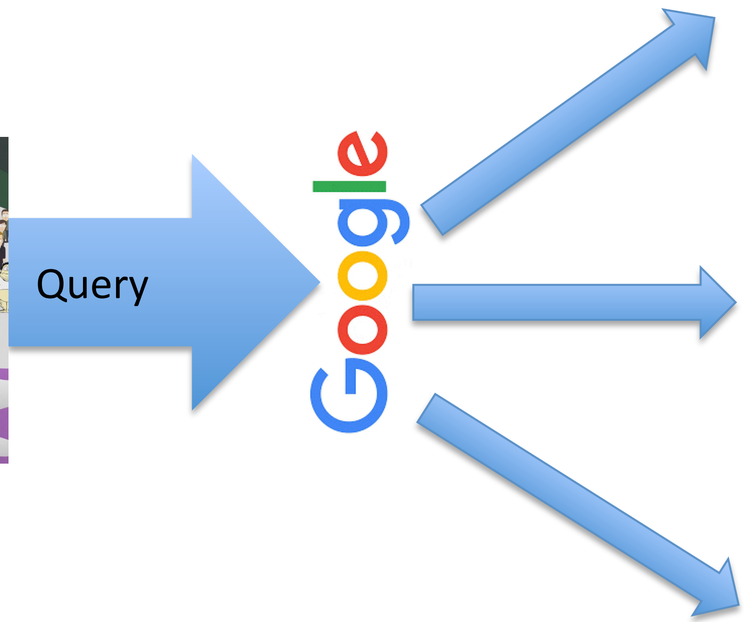
Requirements II

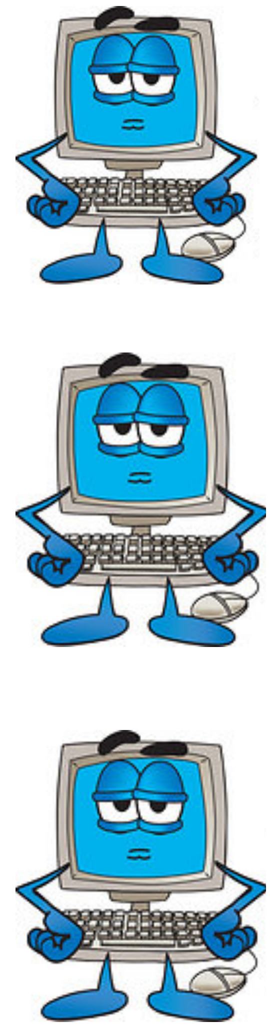
- Fault-tolerance



Requirements III

- Efficient dispatch during deployment





Users
(waiting for a real-time response)

Contributions*

- Balanced Clustering with Fault Tolerance
  - NP-hard
  - Approximation algorithm with strong guarantees
- Nearest Neighbor Dispatch
  - Efficient, Online Dispatch
  - Provably good
- Experiments
  - Classification accuracy after data dependent partitioning
  - Scalability

*Joint work with: Travis Dick, Mu Li, Colin White, Maria-Florina Balcan, Alex Smola
Under submission at AISTATS 2016

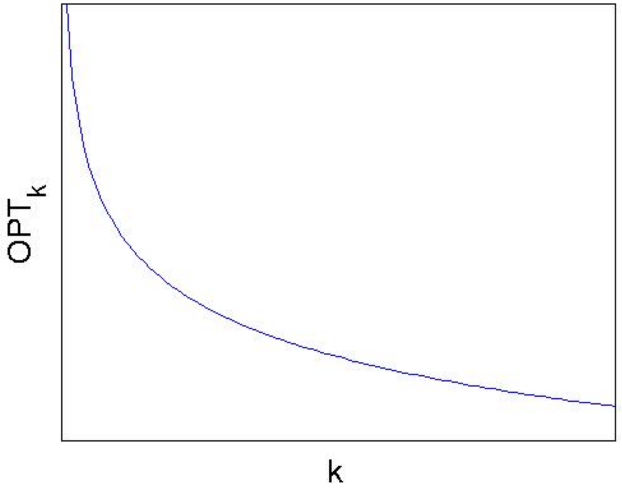
Balanced Clustering with Fault Tolerance

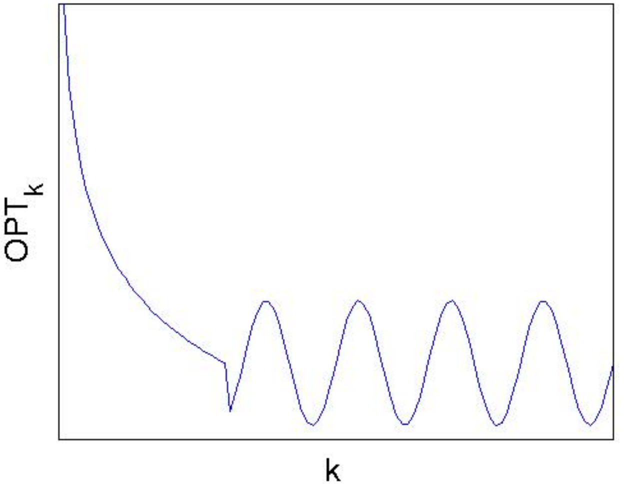
Requirements

• Load balancing: Upper bound on cluster size: *L* fraction	Well studied [KS, ABC+, ABG+]
• Load balancing: Lower bound on cluster size: *l* fraction	Not studied; very tricky
• Fault tolerance: *p* replication	

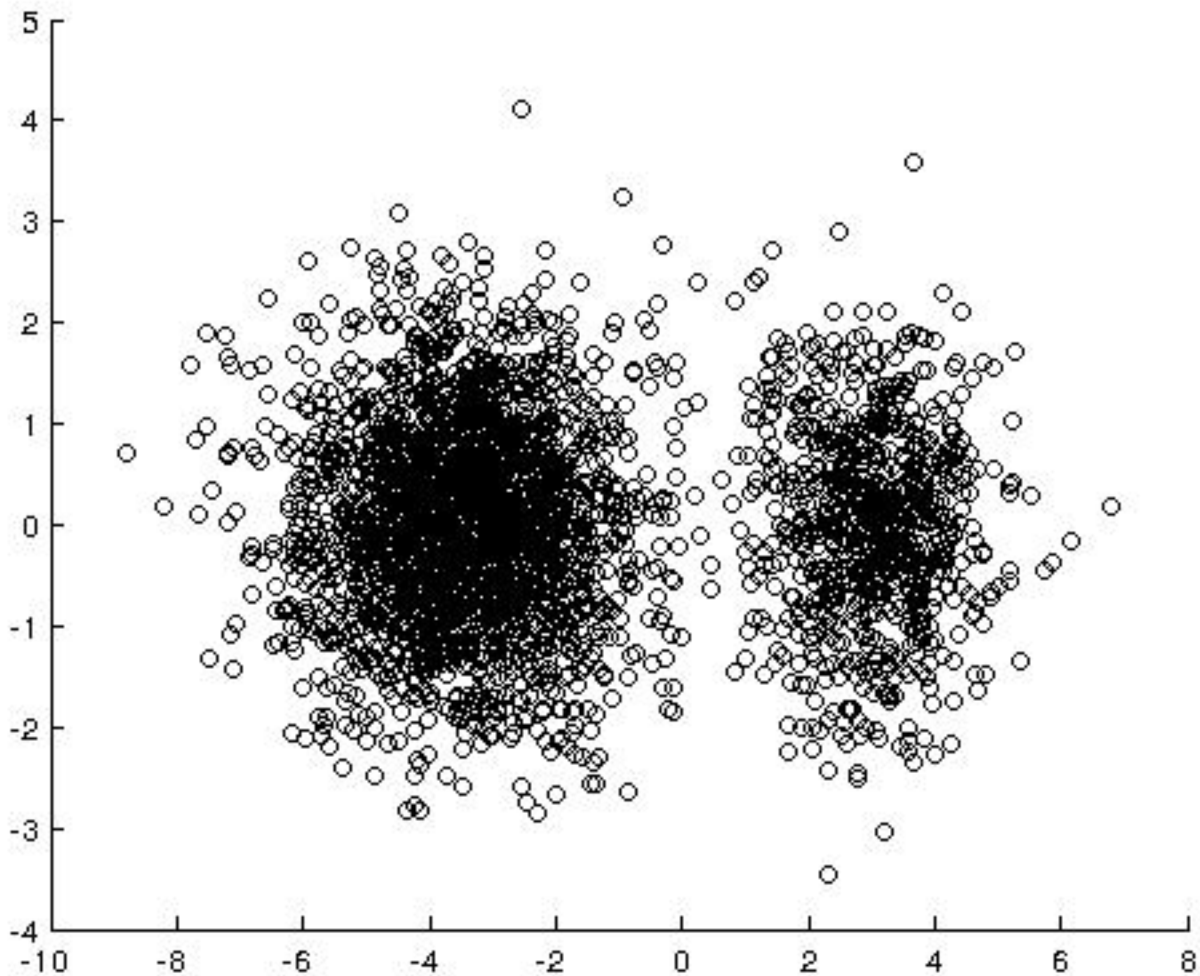
Lower bounds are tricky

- Typically: OPT_k decreases as k increases
- With lower bounds:
  - Arbitrary number of local maxima [DLP+]

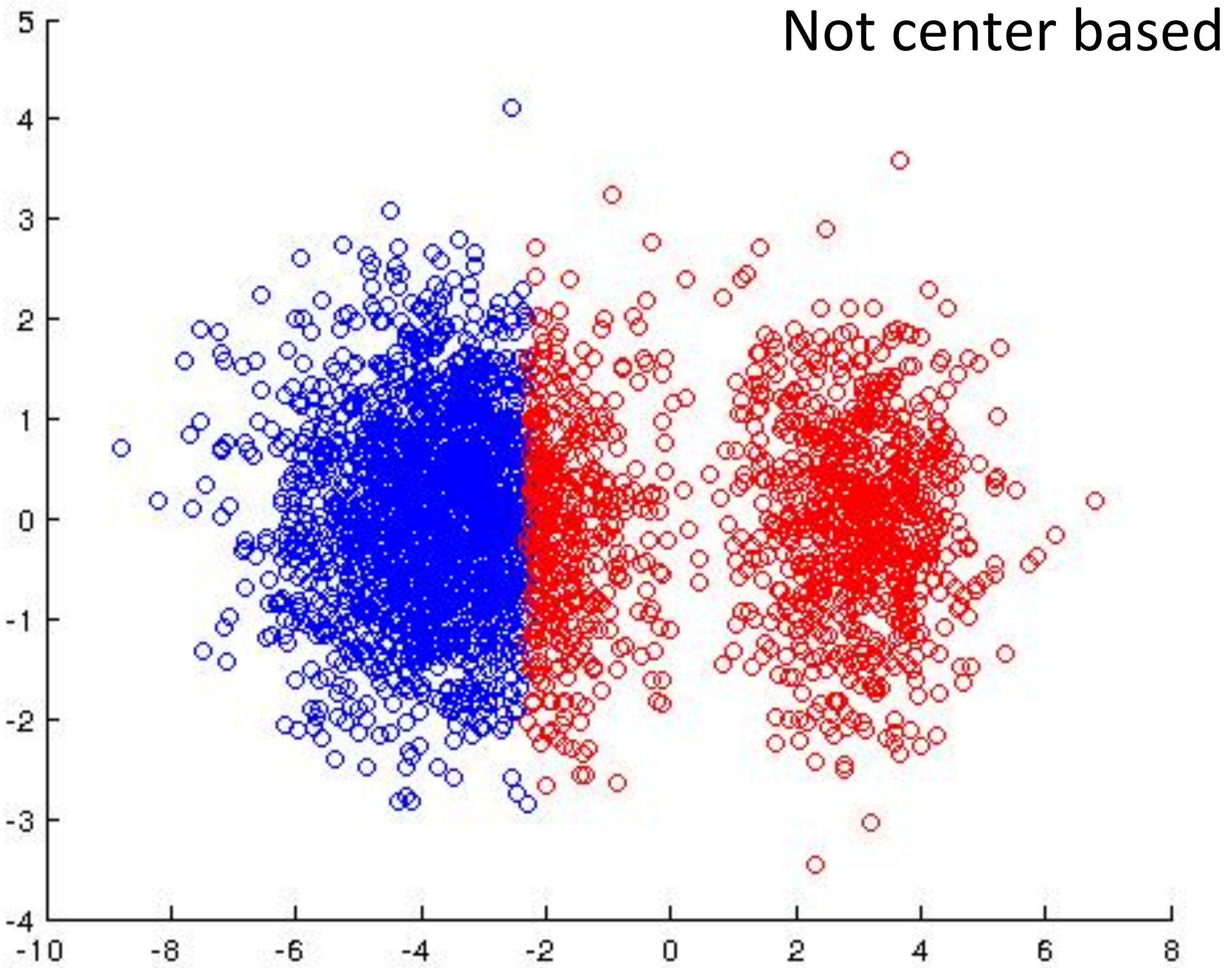




Handling Size Constraints



Handling Size Constraints



Algorithm Overview

- Notation:
  - y_i : point *i* is a center: opening
  - x_{ij} : point *i* is the center corresponding to *j*: assignments
  - *V* : set of points
- Works for any metric space $(\mathcal{X}, d)$

LP Relaxation

K-median: $c_{i,j} = d(i,j)$

K-means: $c_{i,j} = d(i,j)^2$

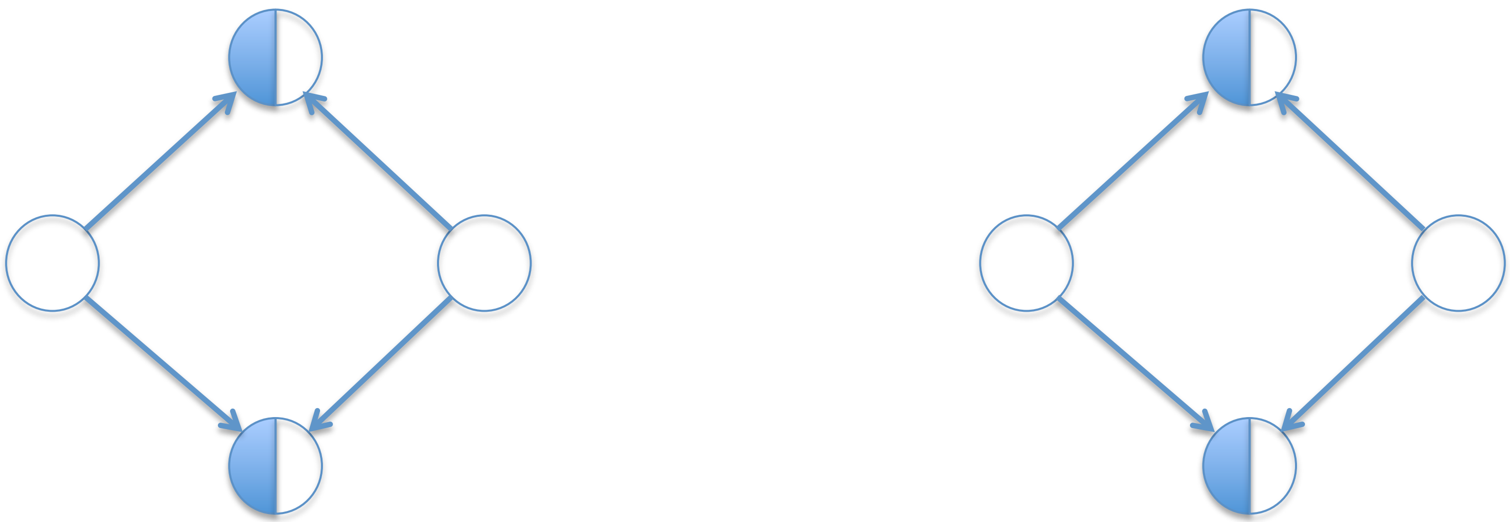
y_i : opening

x_{ij} : assignment

$$
\begin{aligned}
\min \quad & \sum_{i,j \in V} c_{ij} x_{ij} & \\
\text{subject to:} \quad & \sum_{i \in V} x_{ij} = p, & \forall j \in V \\
& \ell y_i \leq \sum_{j \in V} \frac{x_{ij}}{n} \leq L y_i, & \forall i \in V \\
& \sum_{i \in V} y_i \leq k; & \\
& 0 \leq x_{ij} \leq y_i \leq 1, & \forall i, j \in V.
\end{aligned}
$$

LP Relaxation

- May open *2k* half centers- requires rounding

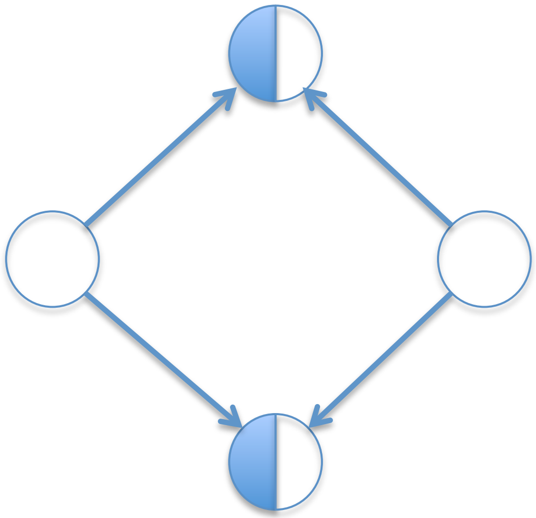


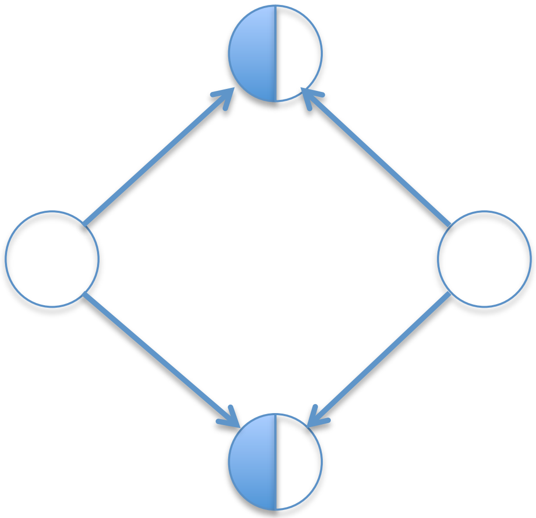
Algorithm Overview

- Step 1 : Solve LP
- Step 2: Round opening
  - Greedy Coarse Clustering to get $\leq k$ coarse clusters: Monarch Procedure
  - Round centers locally within each coarse cluster
- Step 3: Round assignments
  - Round assignments globally with min-cost flow

Algorithm Overview: Step 1

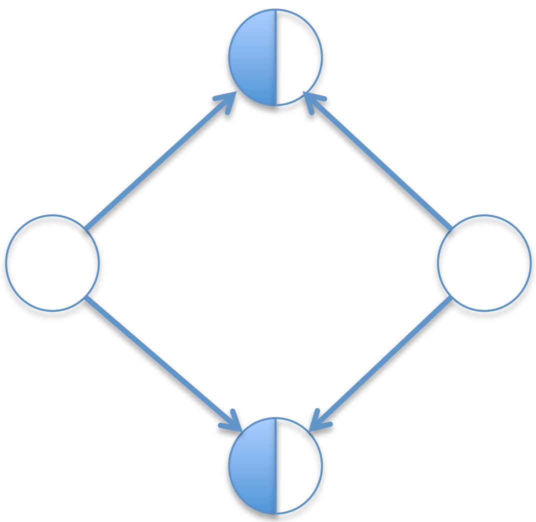
- Solve LP
- Example: 8 points

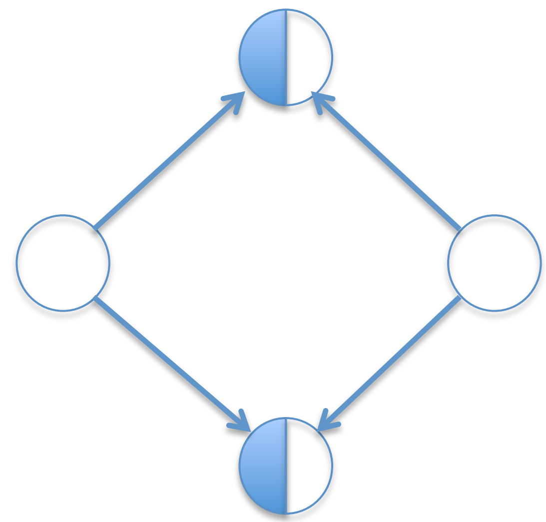




Algorithm Overview: Step 2

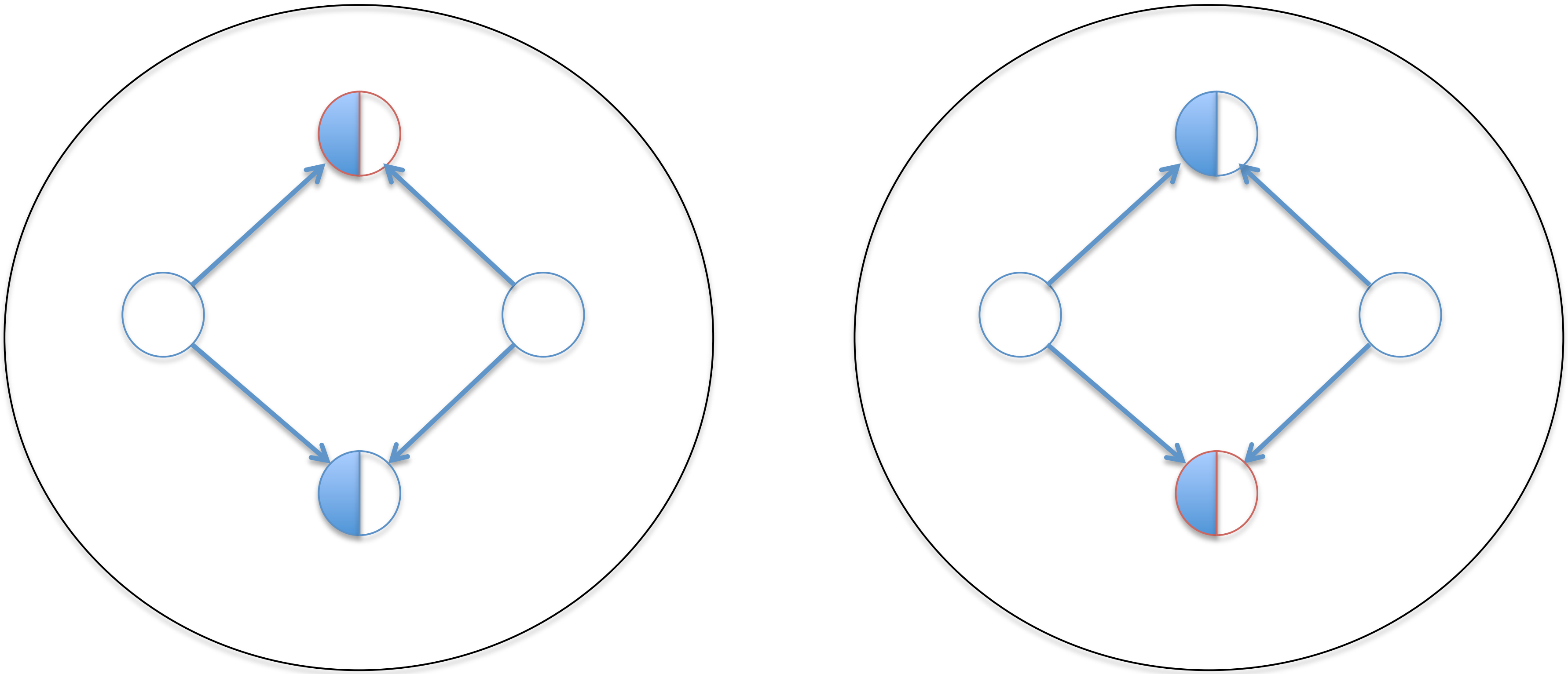
- Perform coarse clustering: Monarch procedure
- Greedy
- Good guarantees





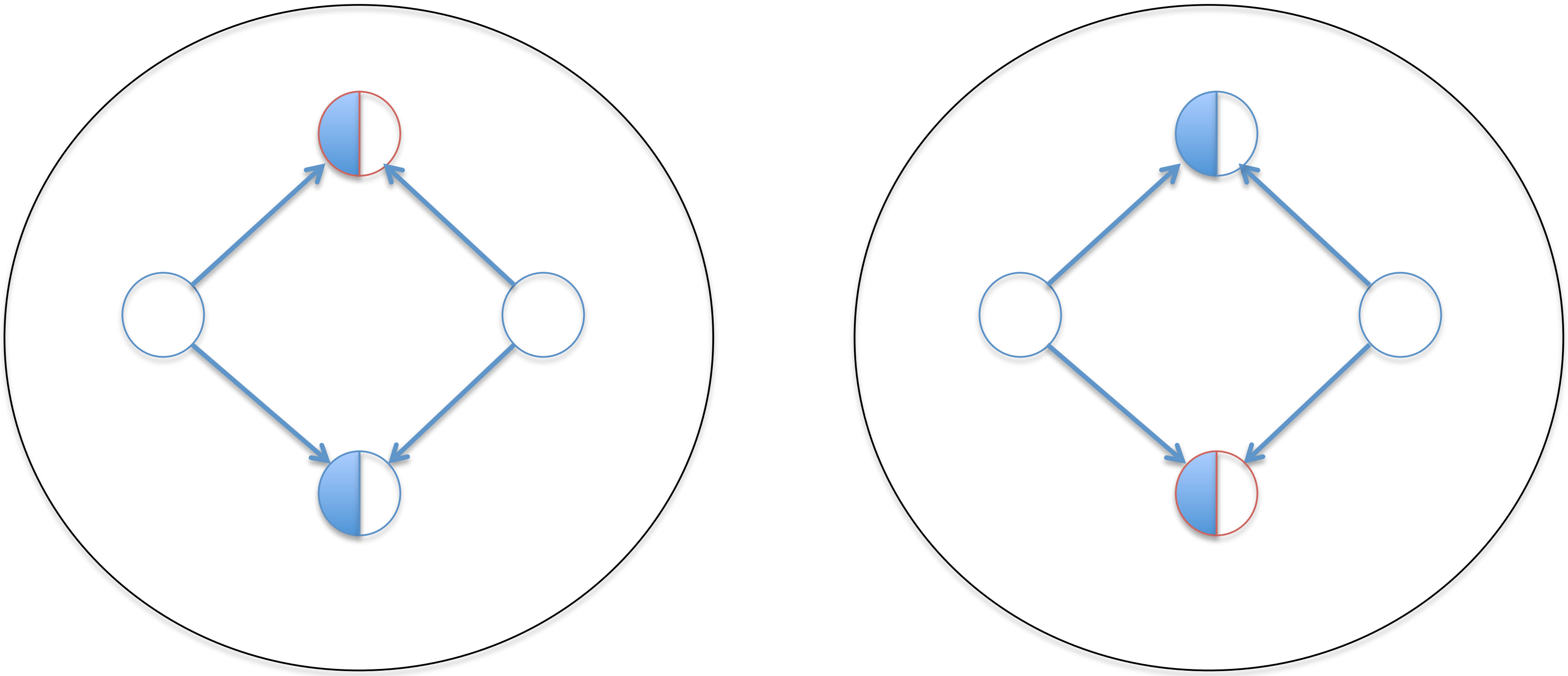
Algorithm Overview: Step 2

- Perform coarse clustering: Monarch procedure
- Greedy
- Good guarantees



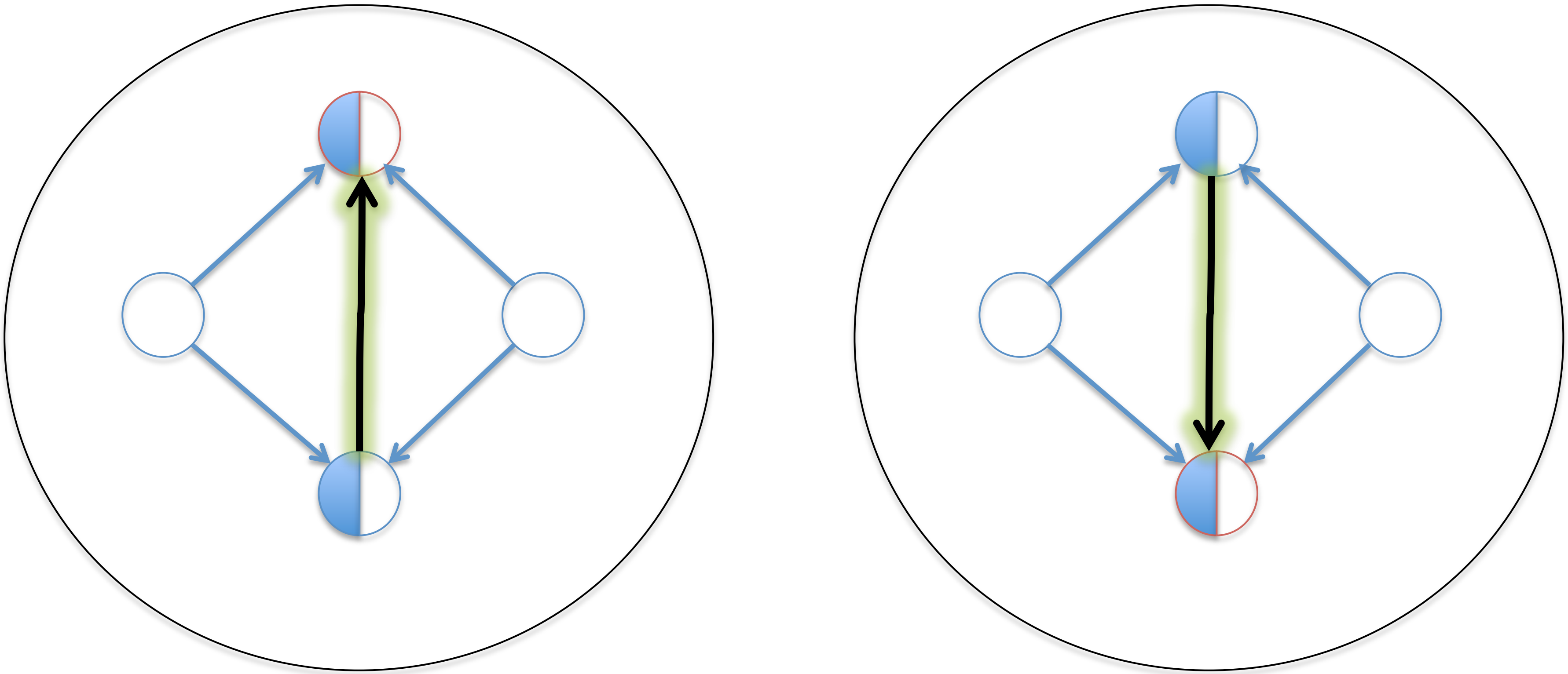
Algorithm Overview: Step 2

- Round opening within each coarse cluster



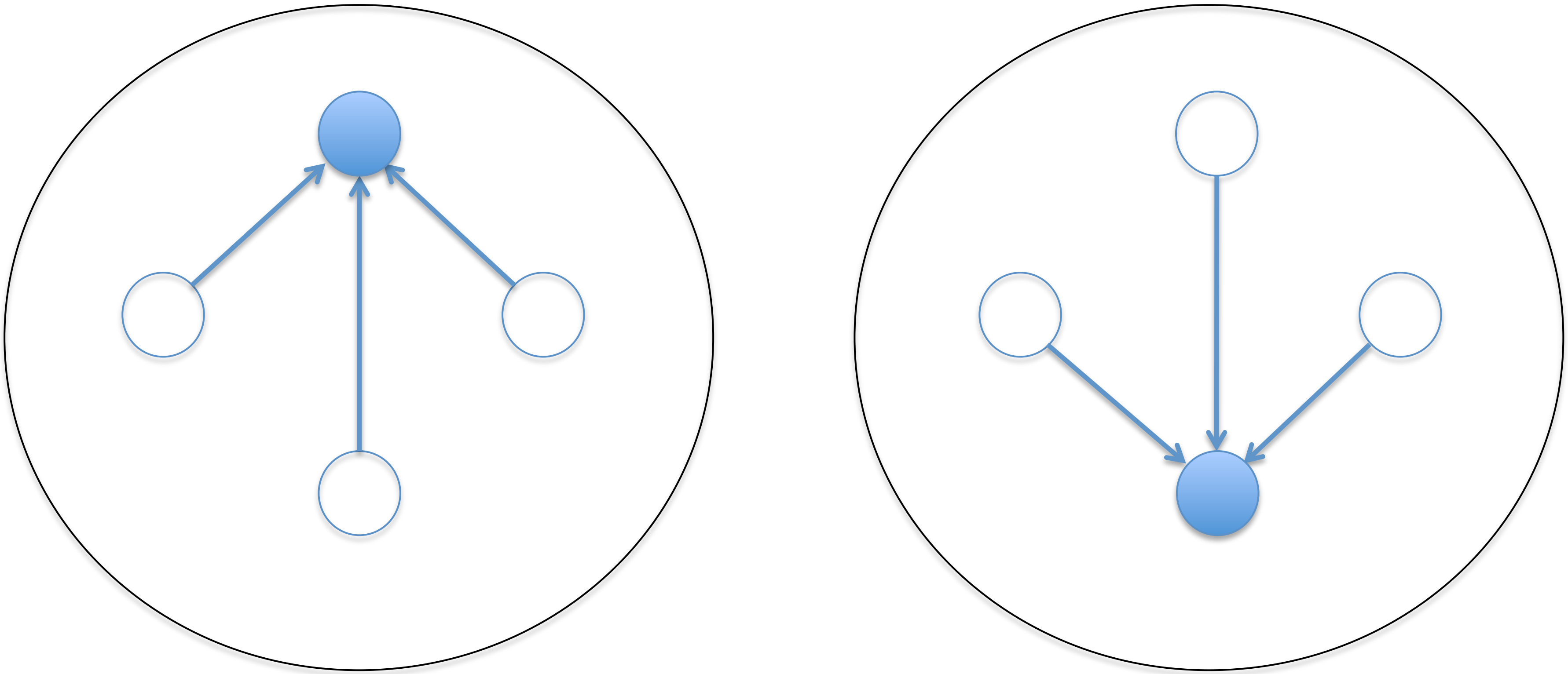
Algorithm Overview: Step 2

- Round opening within each coarse cluster



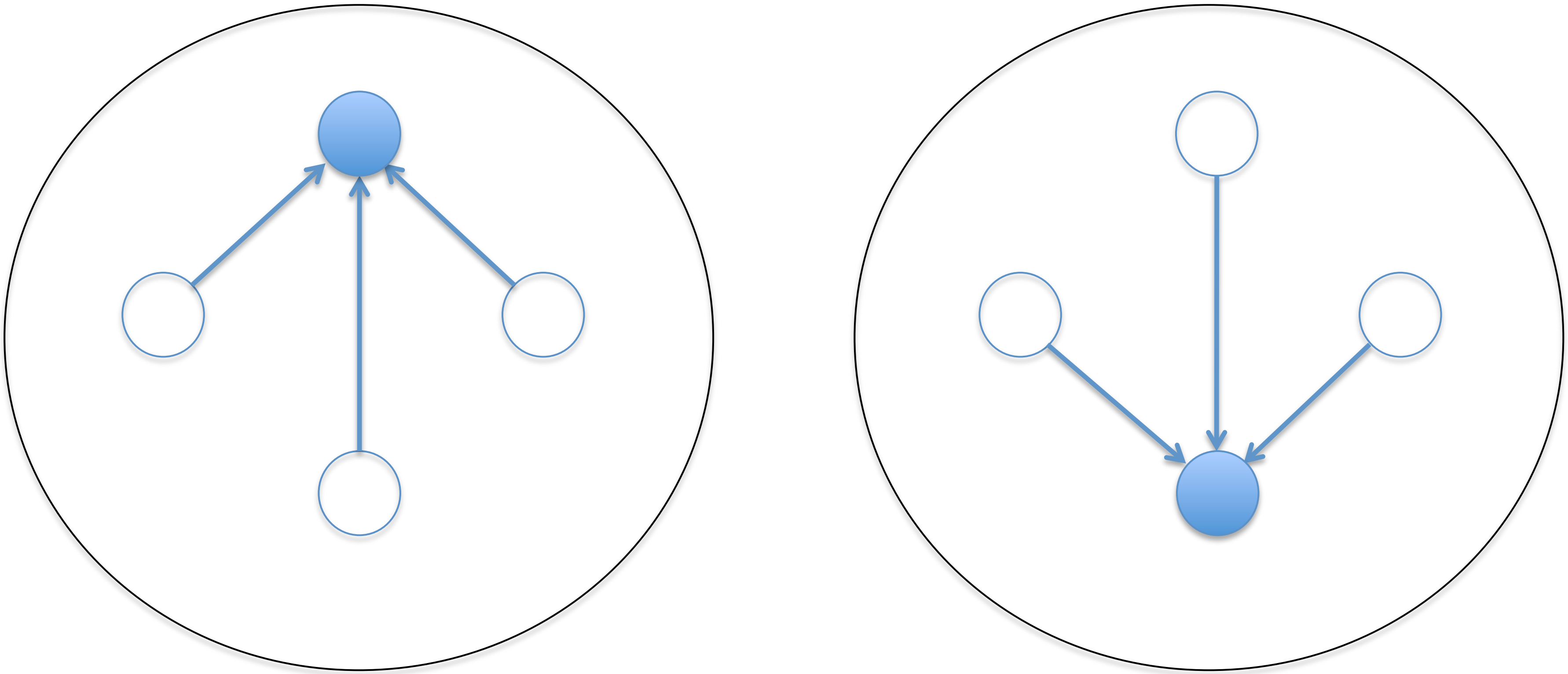
Algorithm Overview: Step 2

- Round opening within cluster



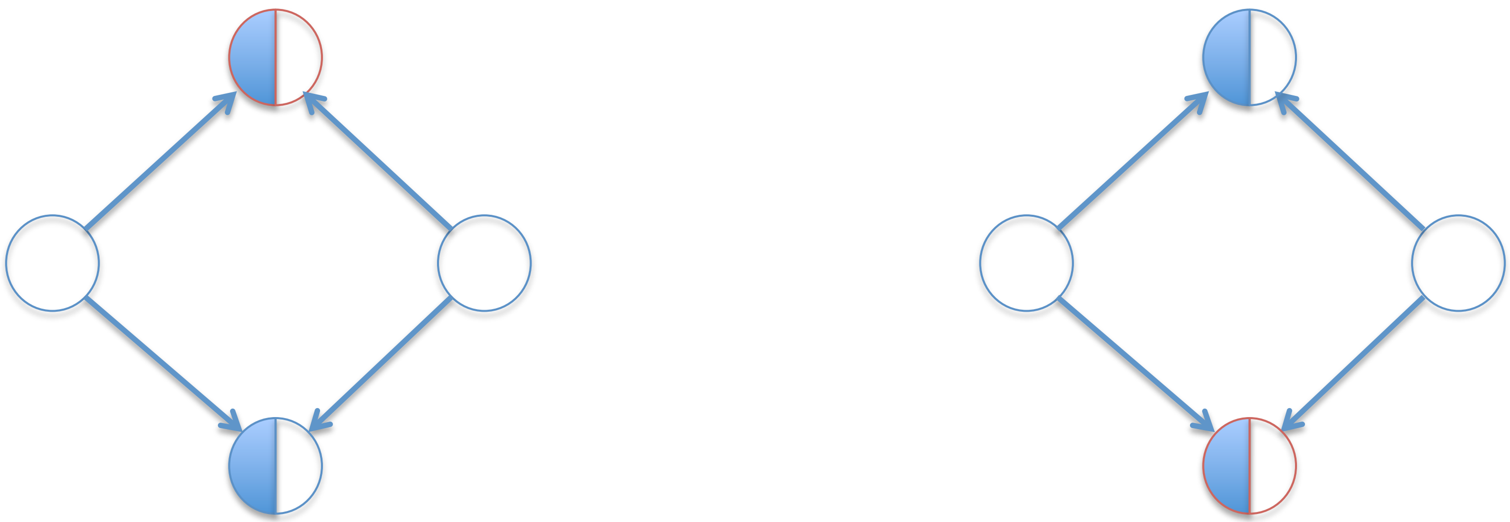
Algorithm Overview: Step 3

- Round Assignments with min-cost flow.



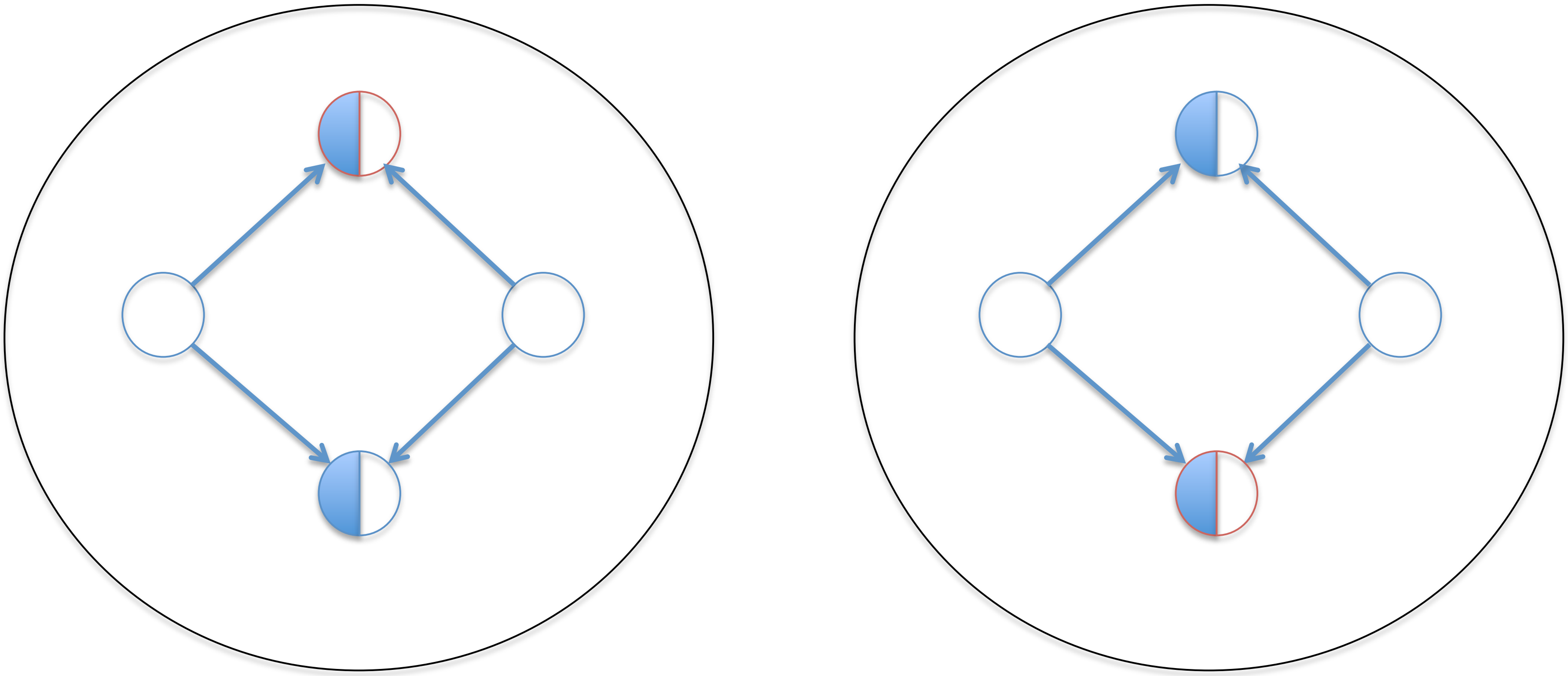
Step 2: Monarch Procedure

- Greedily pick ≤k points as monarchs



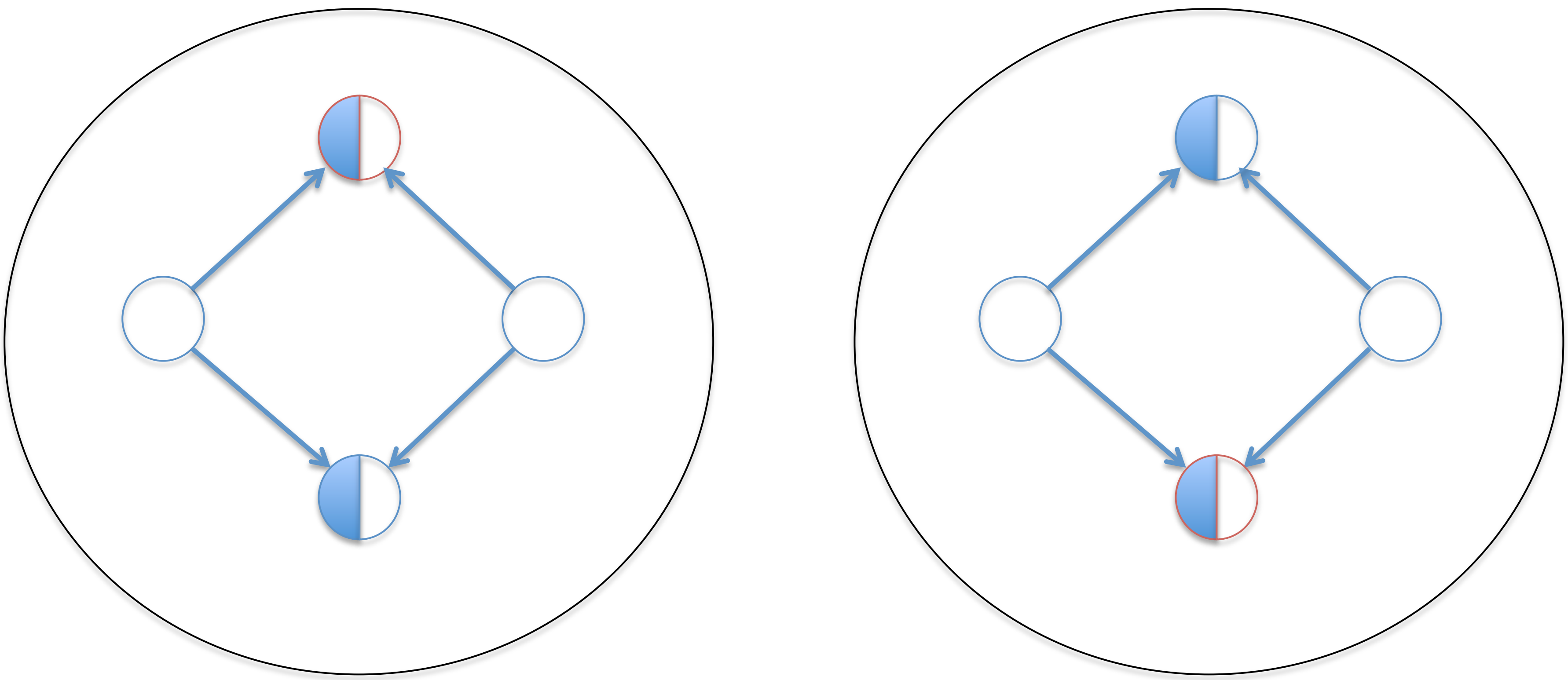
Step 2: Monarch Procedure

- Empires: Voronoi partitions about monarchs



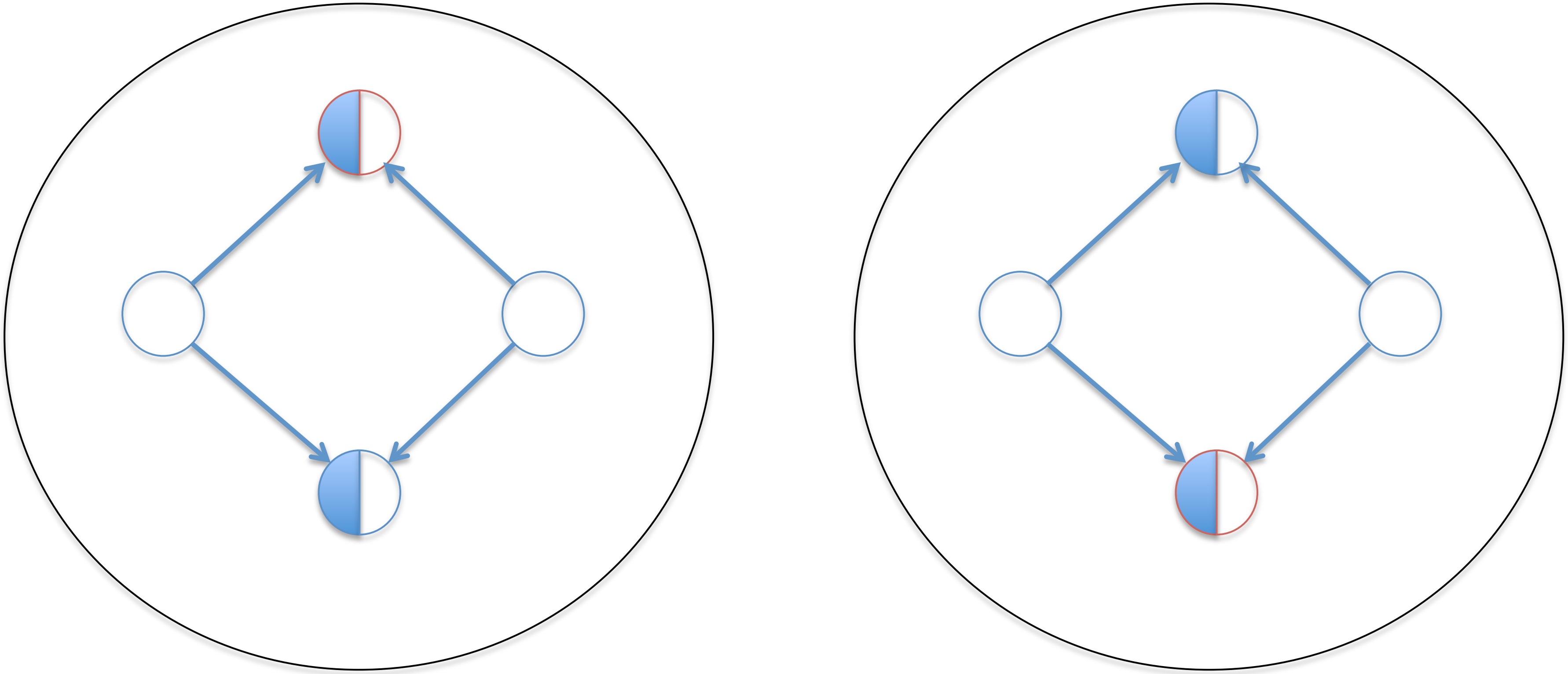
Step 2: Monarch Procedure

- Greedy rule: pick point with highest contribution to the objective (as long as it does not have a monarch nearby)



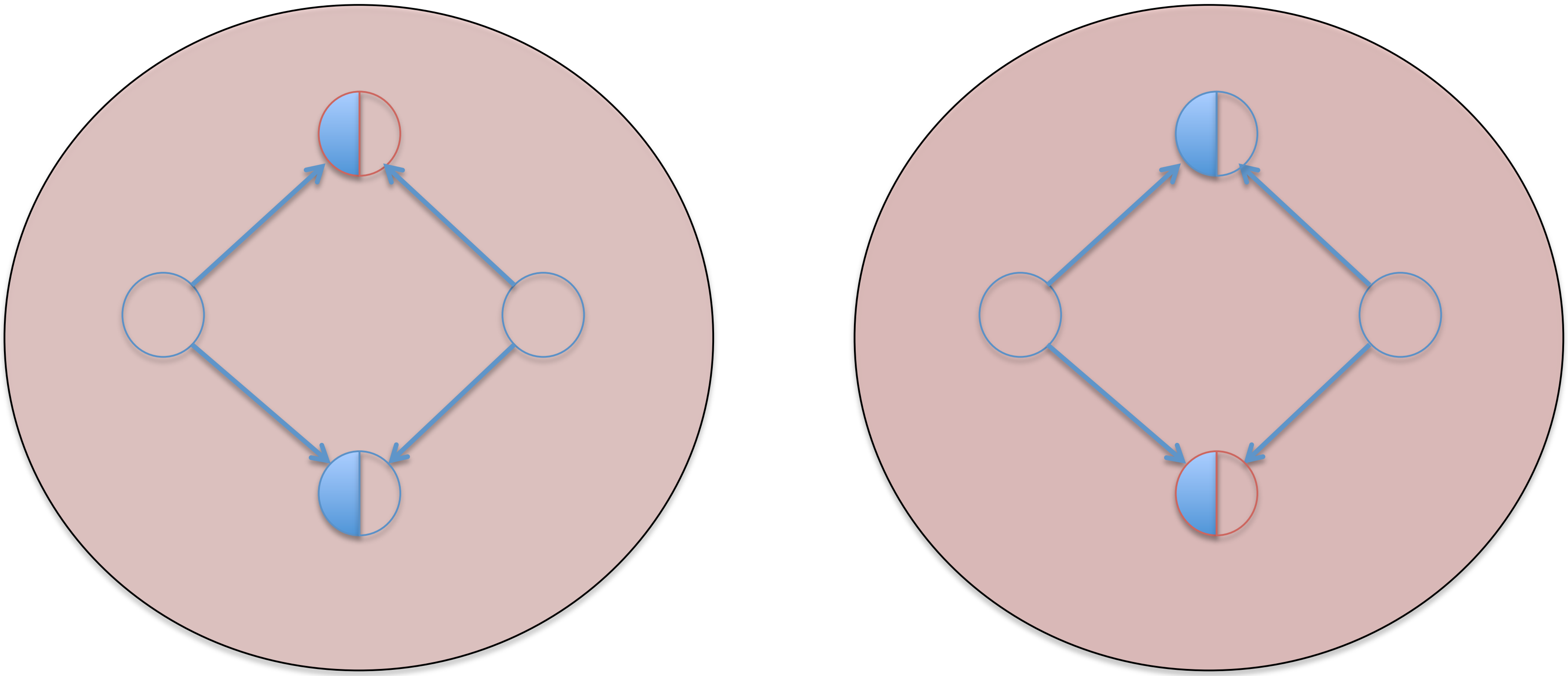
Step 2: Monarch Procedure

- Why this greedy rule?



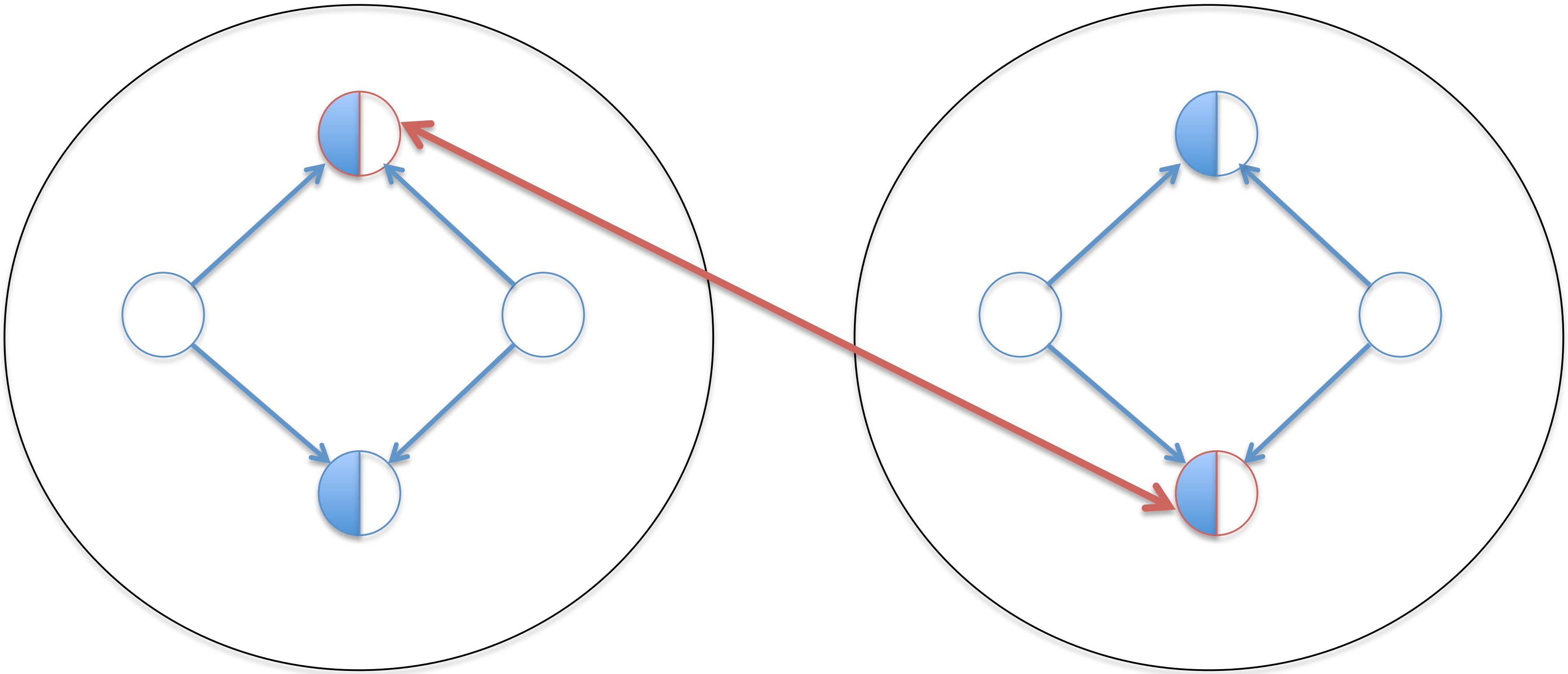
Step 2: Monarch Procedure Guarantees

- Points within an empire are close



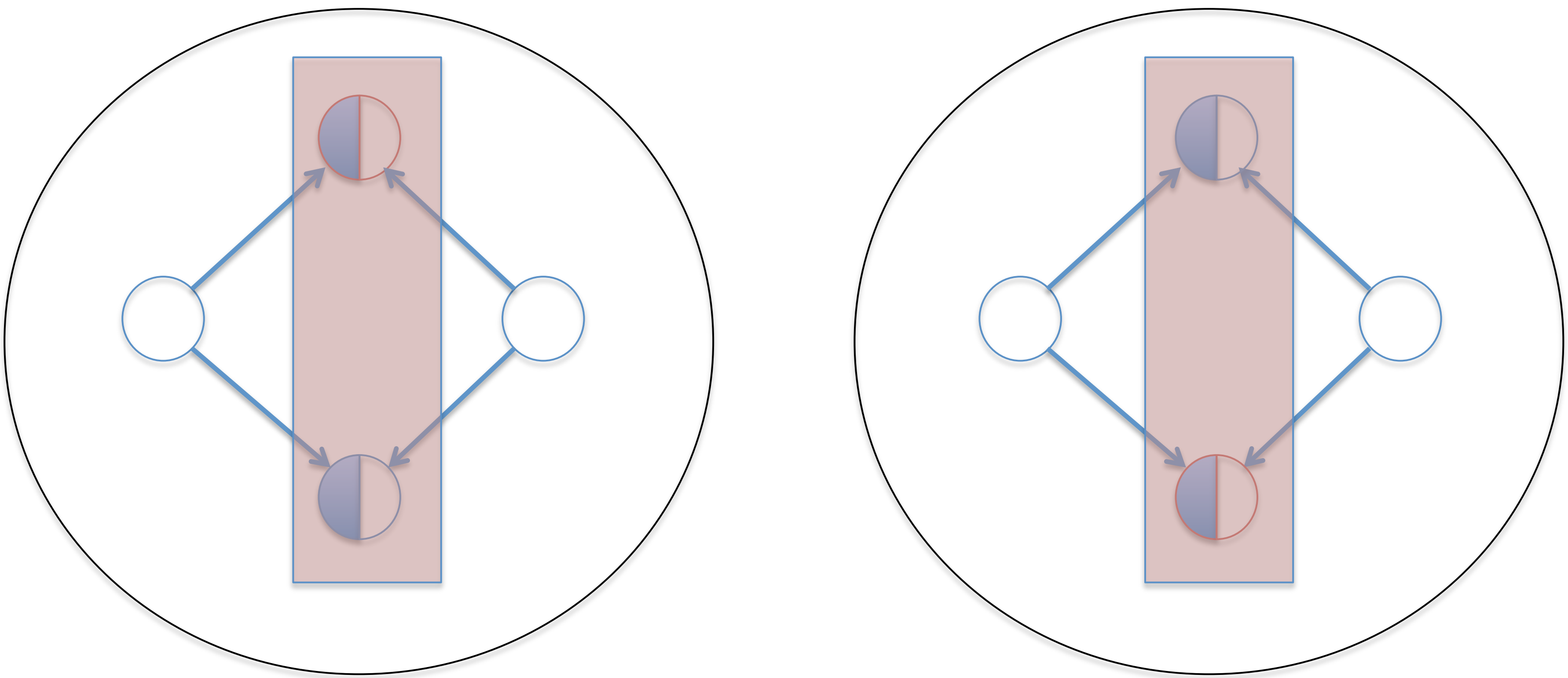
Step 2: Monarch Procedure Guarantees

- Monarchs are far apart



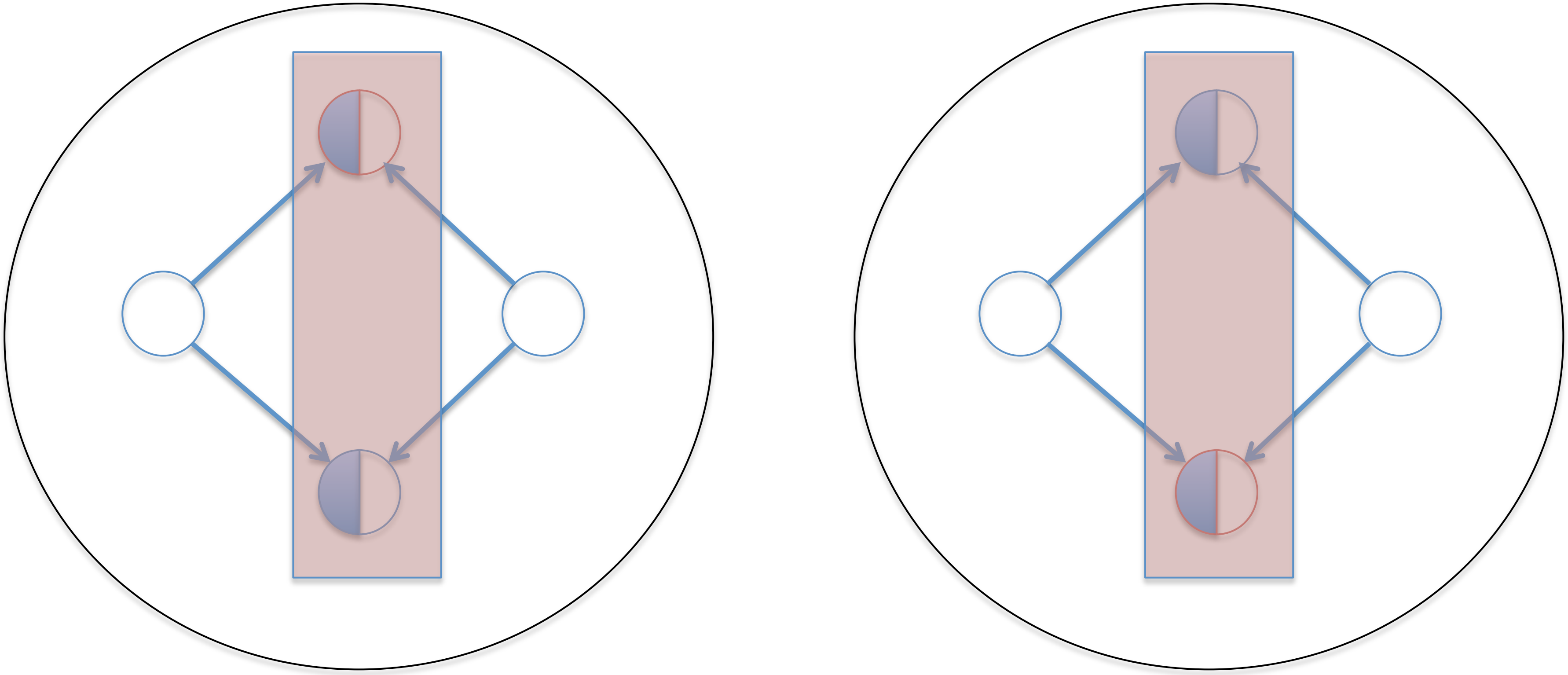
Step 2: Monarch Procedure Guarantees

- Each empire has opening ≥$p/2$: Markov Inequality



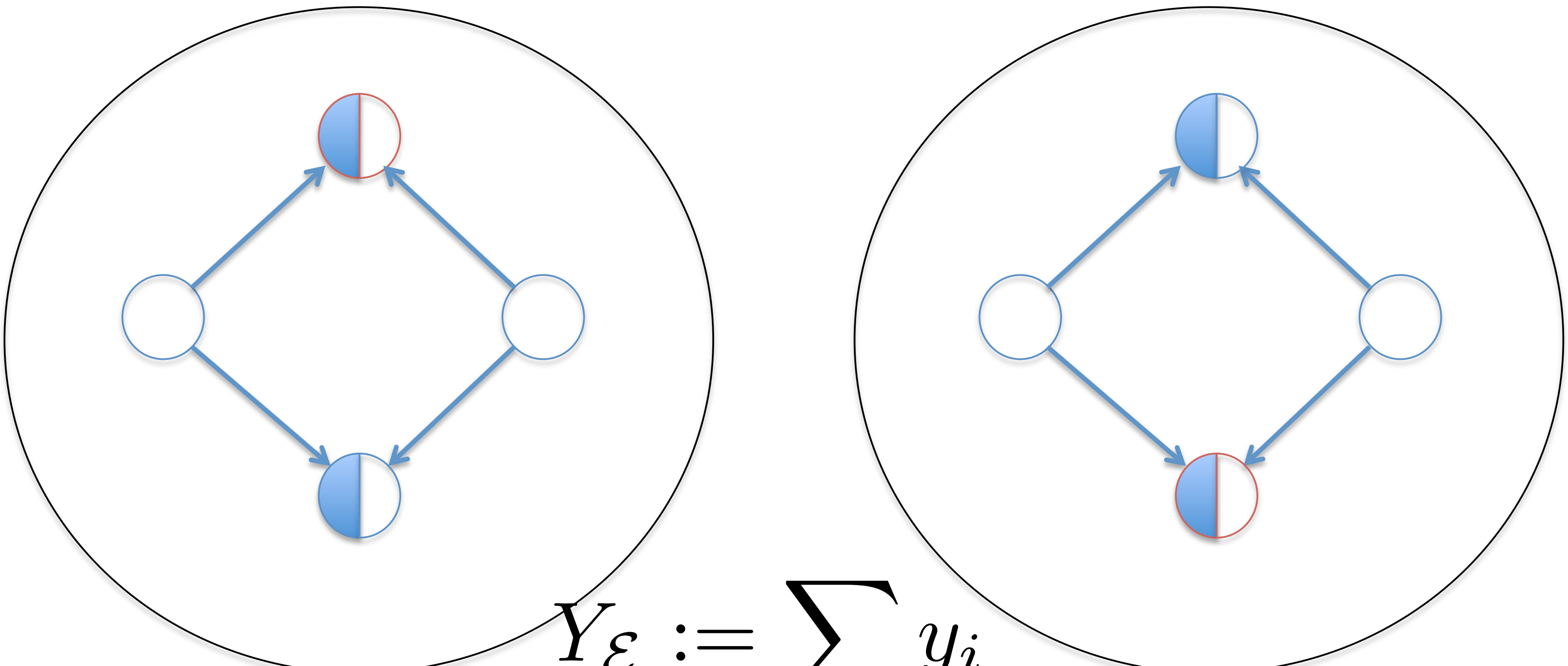
Step 2: Monarch Procedure Guarantees

- Each empire has opening at least 1! (for *p>1*)
- Round *locally* within each empire!



Step 2: Rounding within Empire

- Pick $\lfloor Y_{\mathcal{E}} \rfloor$ central points from each empire, each with opening $Y_{\mathcal{E}}/\lfloor Y_{\mathcal{E}} \rfloor$; make centers



$$Y_{\mathcal{E}} := \sum_{i \in \mathcal{E}} y_i$$

LP Relaxation

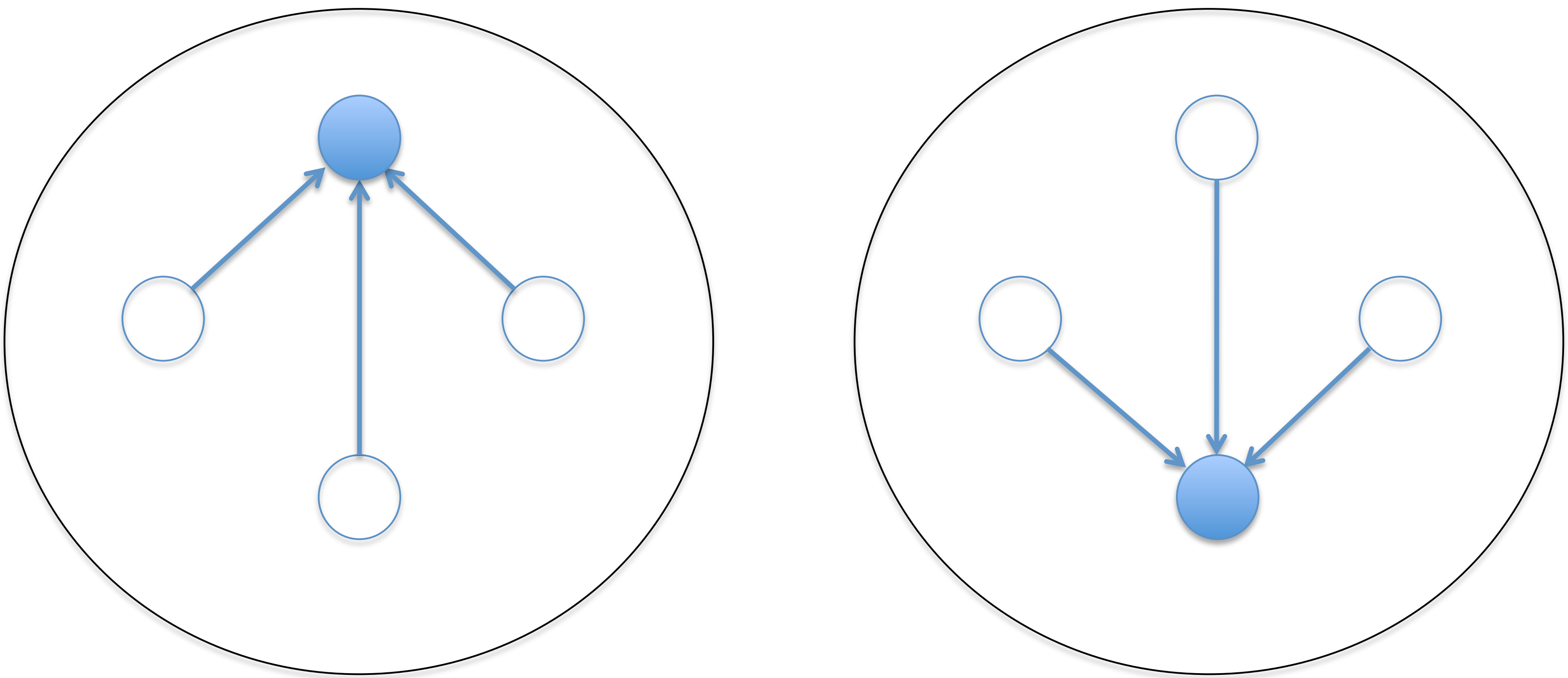
K-median: $c_{i,j} = d(i,j)$
K-means: $c_{i,j} = d(i,j)^2$

y_i : opening
x_{ij} : assignment

$$\begin{aligned}
\min \quad & \sum_{i,j \in V} c_{ij} x_{ij} & \\
\text{subject to:} \quad & \sum_{i \in V} x_{ij} = p, & \forall j \in V \\
& \ell y_i \le \sum_{j \in V} \frac{x_{ij}}{n} \le L y_i, & \forall i \in V \\
& \sum_{i \in V} y_i \le k; & \\
& 0 \le x_{ij} \le y_i \le 1, & \forall i, j \in V.
\end{aligned}$$

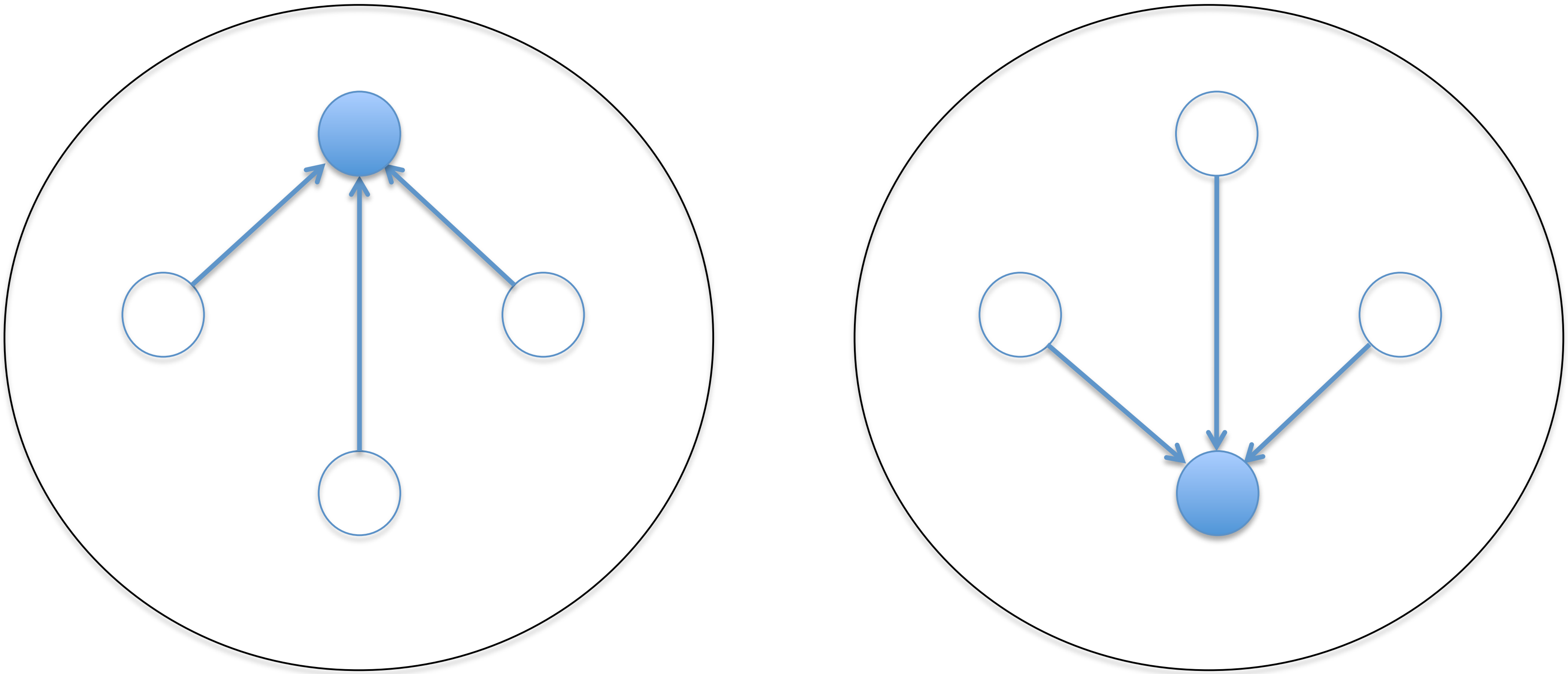
Step 2: Rounding within Empire

- Same factor appears as violation of cluster size constraint: $Y_{\mathcal{E}} / \lfloor Y_{\mathcal{E}} \rfloor \leq {p+2}/{p}$



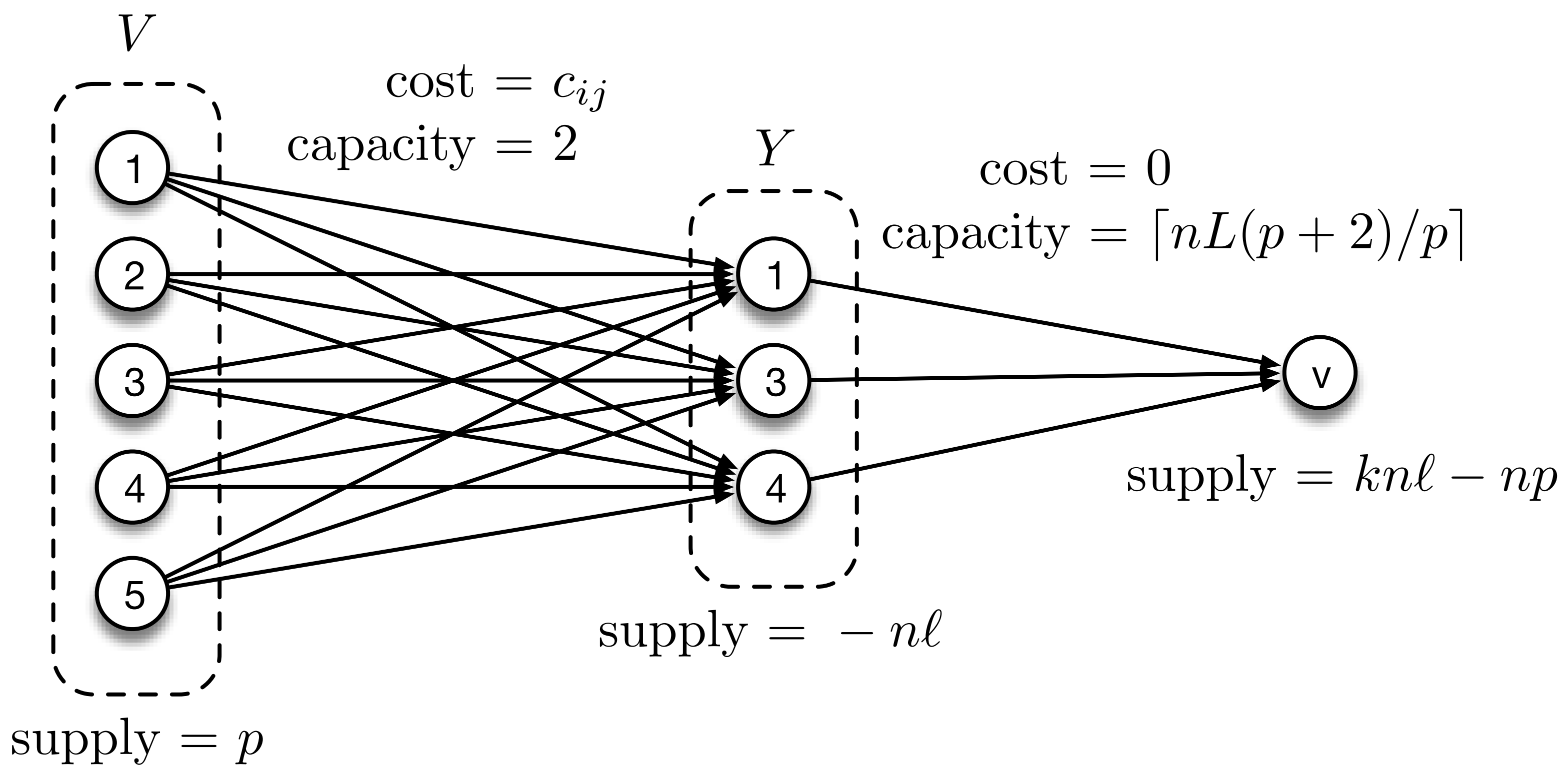
Step 2: Rounding Guarantee

- Obtain a feasible solution with integral *y*
- Cost bounded by triangle inequality



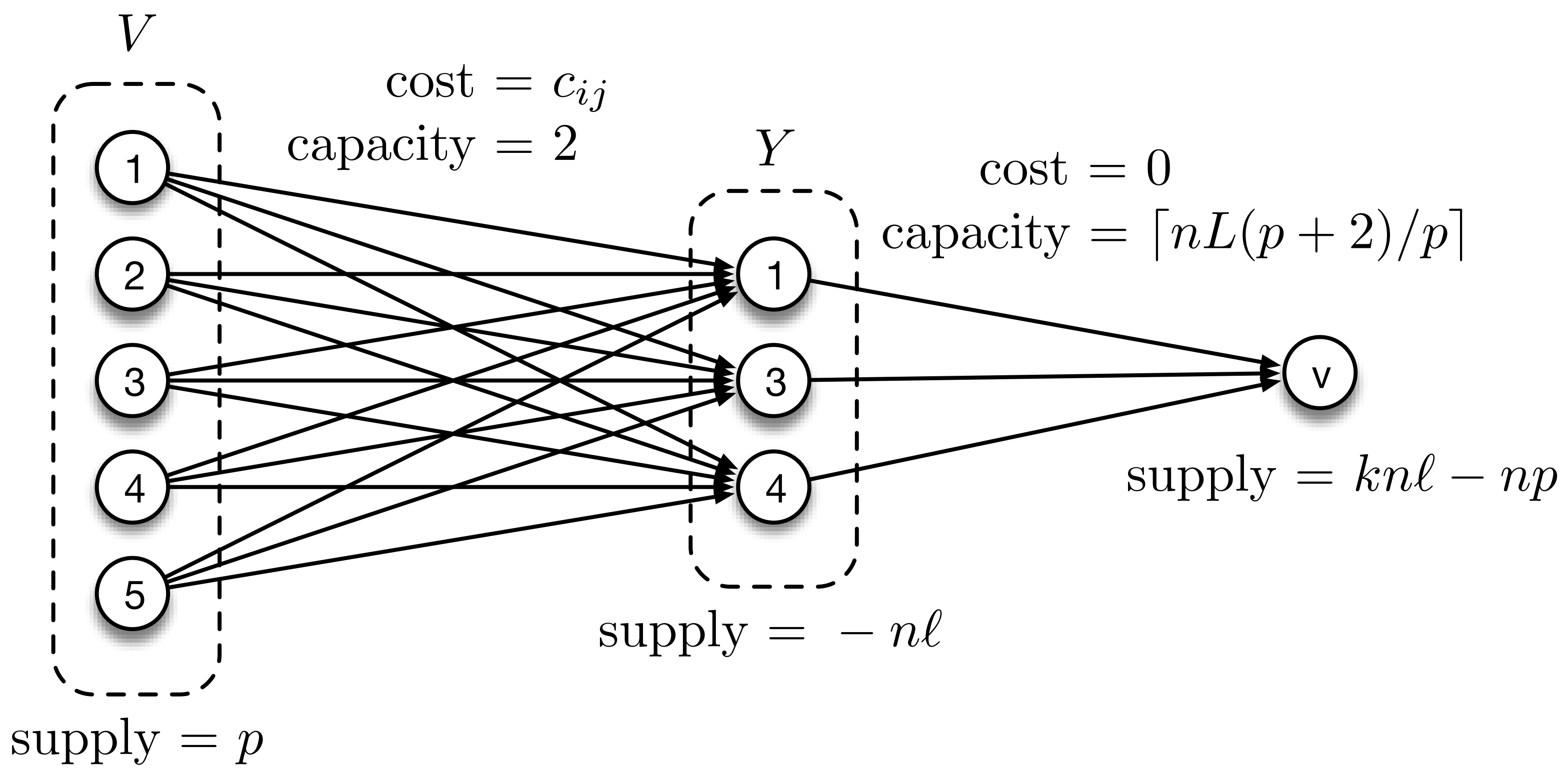
Step 3: Rounding assignments

- Easy: Min cost flow



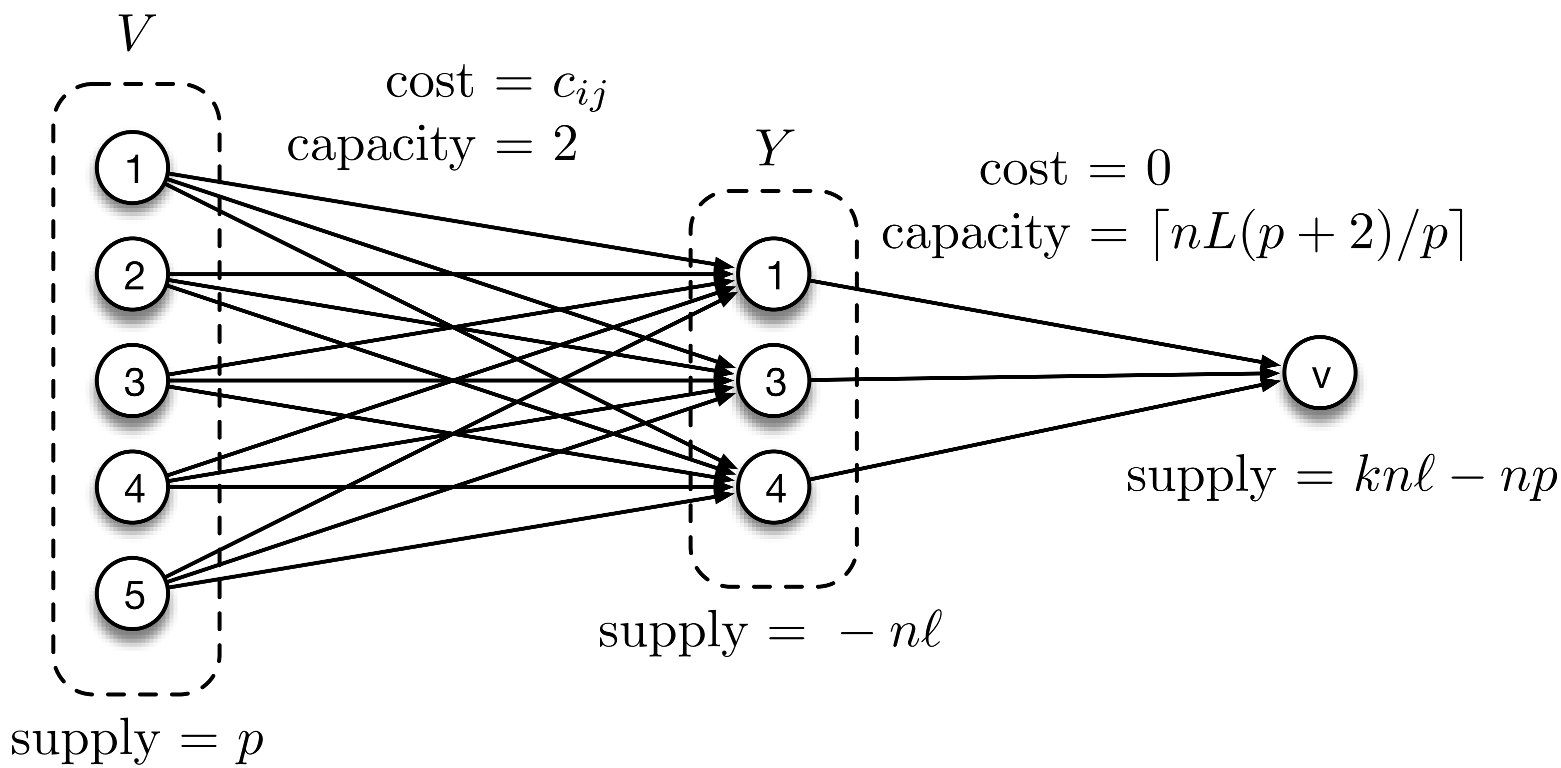
Step 3: Rounding assignments

- Fractional LP solution implies a feasible flow



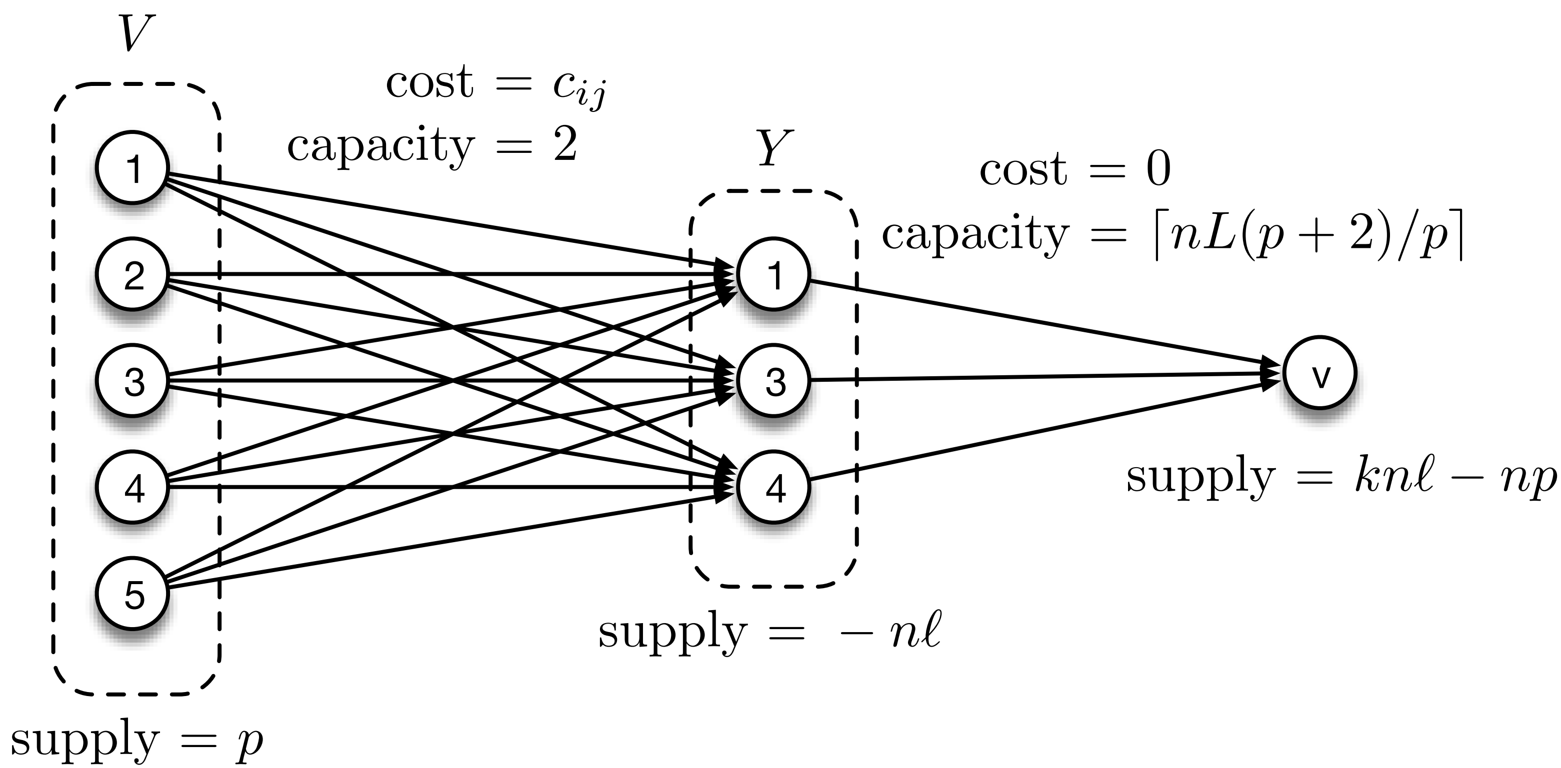
Step 3: Rounding assignments

- By Integral Flow Theorem, there is an optimal integral flow



Step 3: Rounding assignments

- Can be computed by standard algorithms



To sum up…

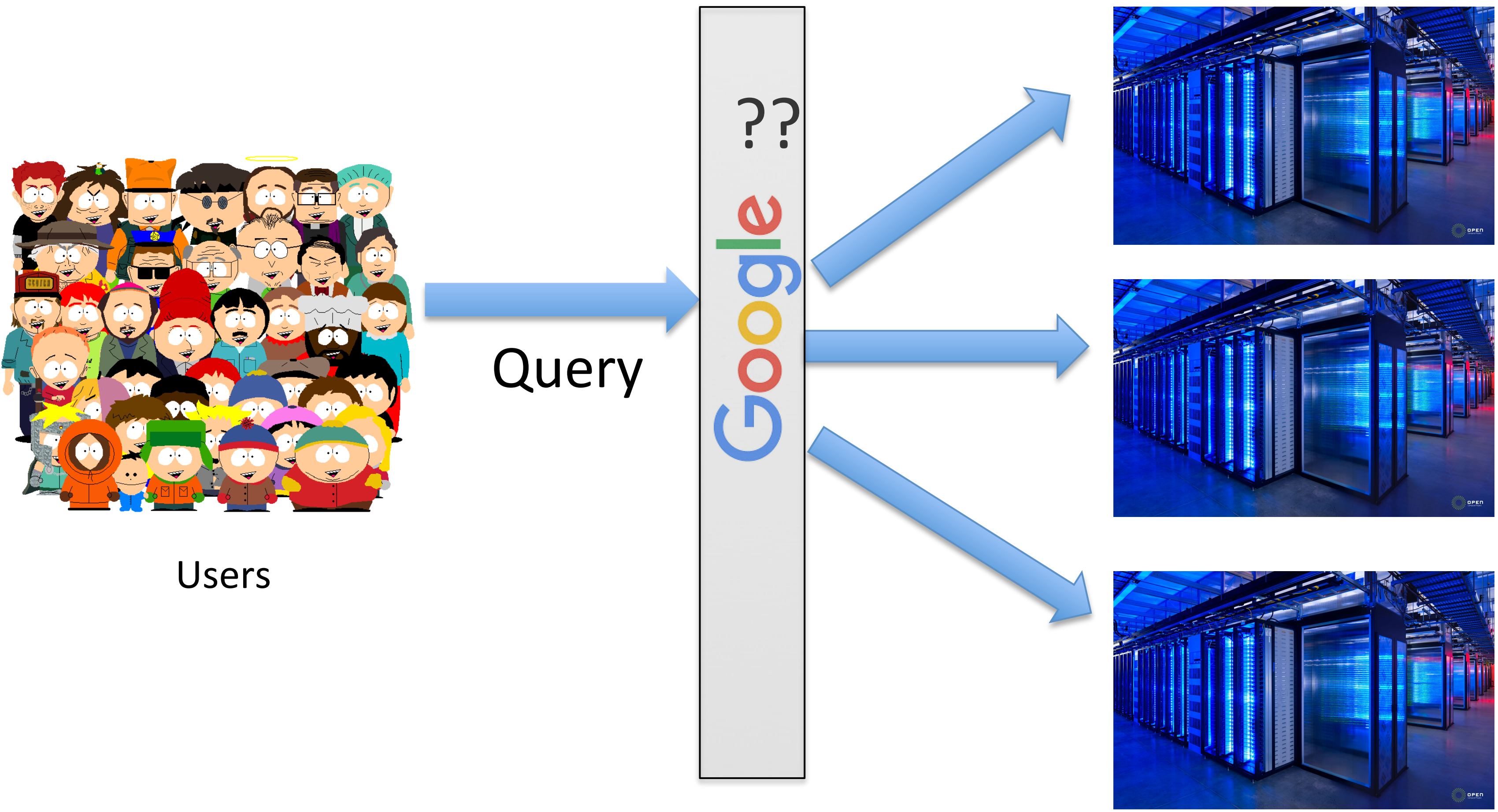
Theorem: There exist poly time approximation algorithms for balanced *k*-clustering with fault tolerance

- that output
  - 5 approx. for *k*-center
  - 11 approx. for *k*-median
  - 95 approx. for *k*-means, and
- cluster size constraint is violated by $(p+2)/p$
- replication between p and $p/2$.

Nearest Neighbor Dispatch

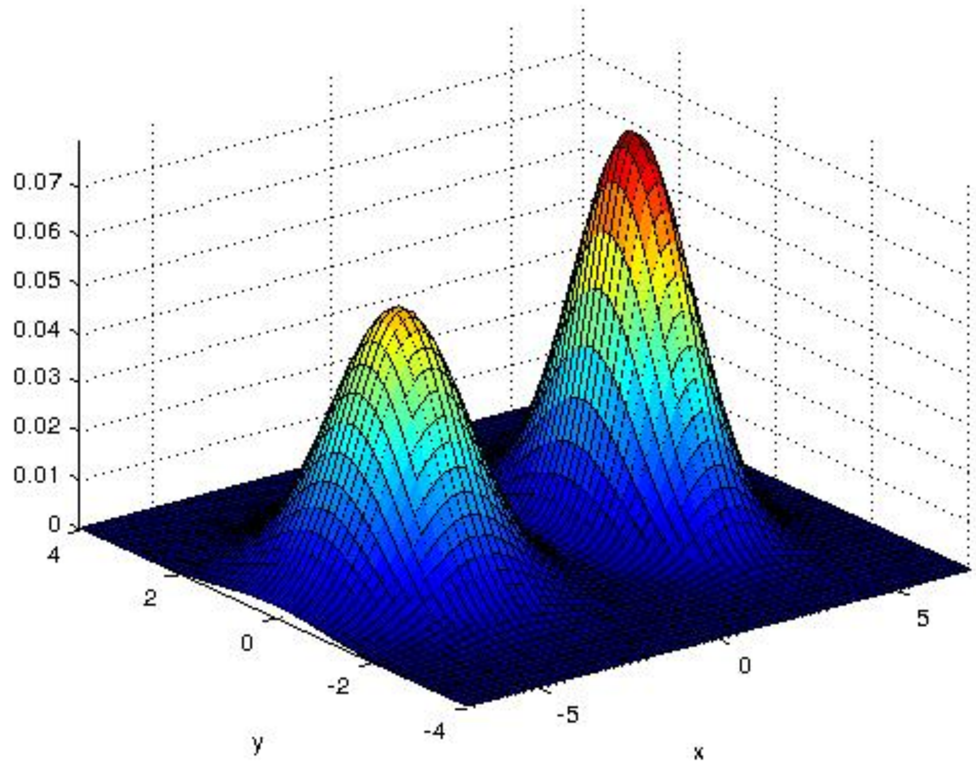
Requirements

- Dispatch a new point correctly and efficiently



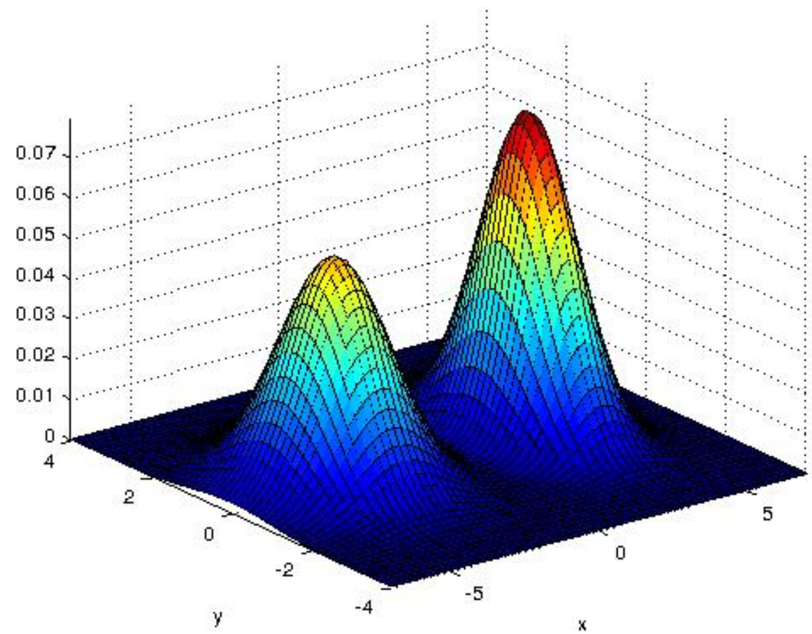
Goal

- PAC Assumption: data are drawn iid from some fixed unknown distribution μ



Goal

- Given an iid sample from μ, cluster the distribution
- Balance constraints: Probability mass of each cluster is within *(l,L)*.



Our solution

- Cluster a sample (previous section)
- Extend clustering to the distribution
- How?

Clustering a Distribution

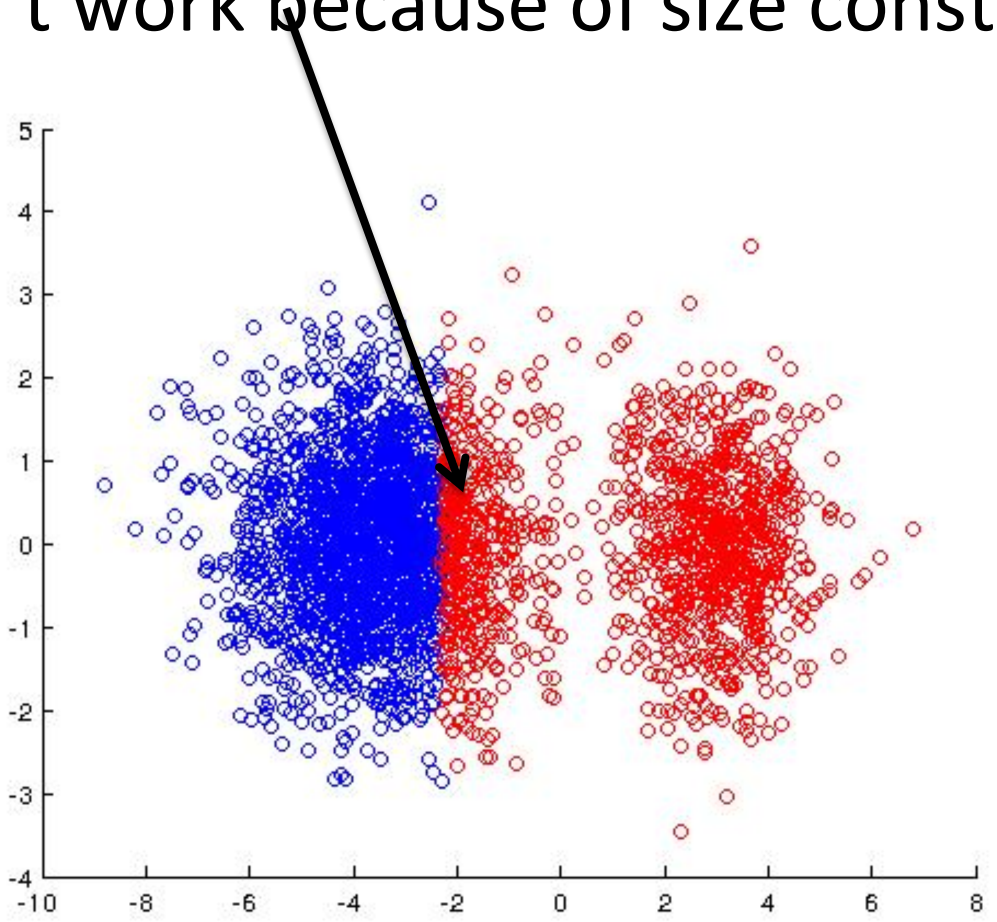
Assignments $f : \mathcal{X} \to \binom{k}{p}$

Centers $c : [k] \to \mathcal{X}$

K-median: $\min_{f,c} \quad \mathbb{E}_{x\sim\mu}\left[\sum_{i\in f(x)} \|x - c(i)\|\right]$

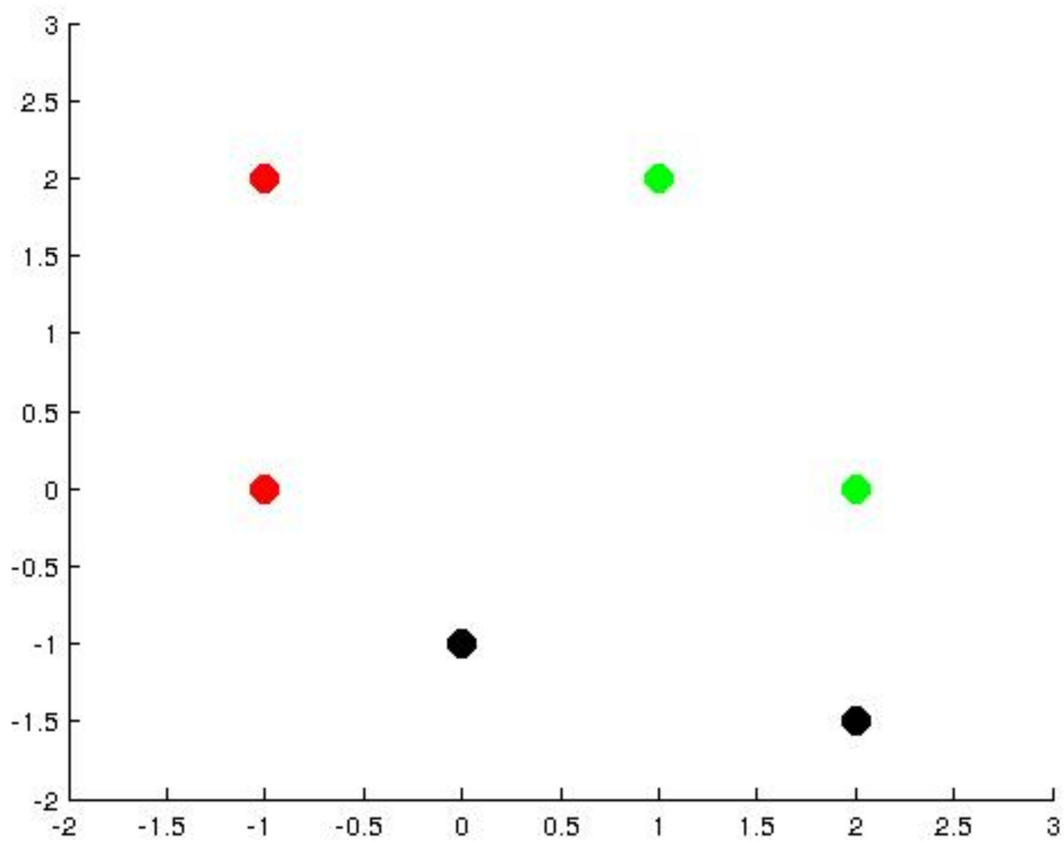
Find the Nearest Center?

- Doesn't work because of size constraints



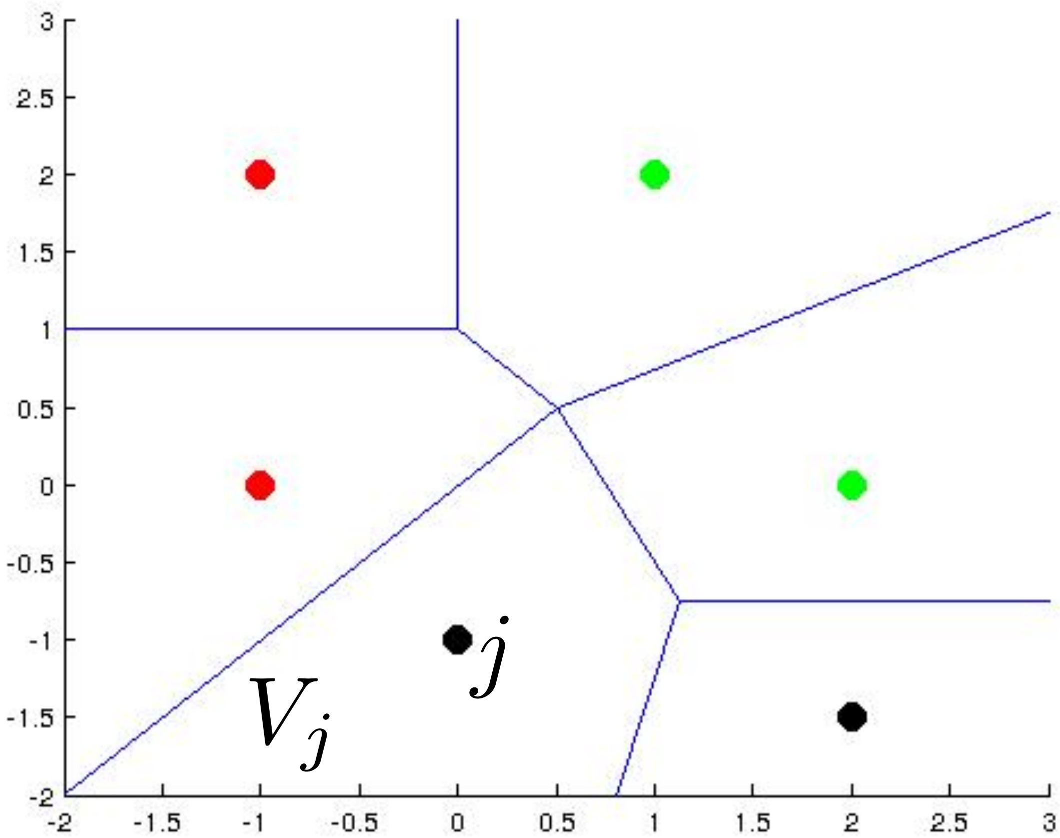
An Idea: NN Extension

- Find nearest point from the original sample



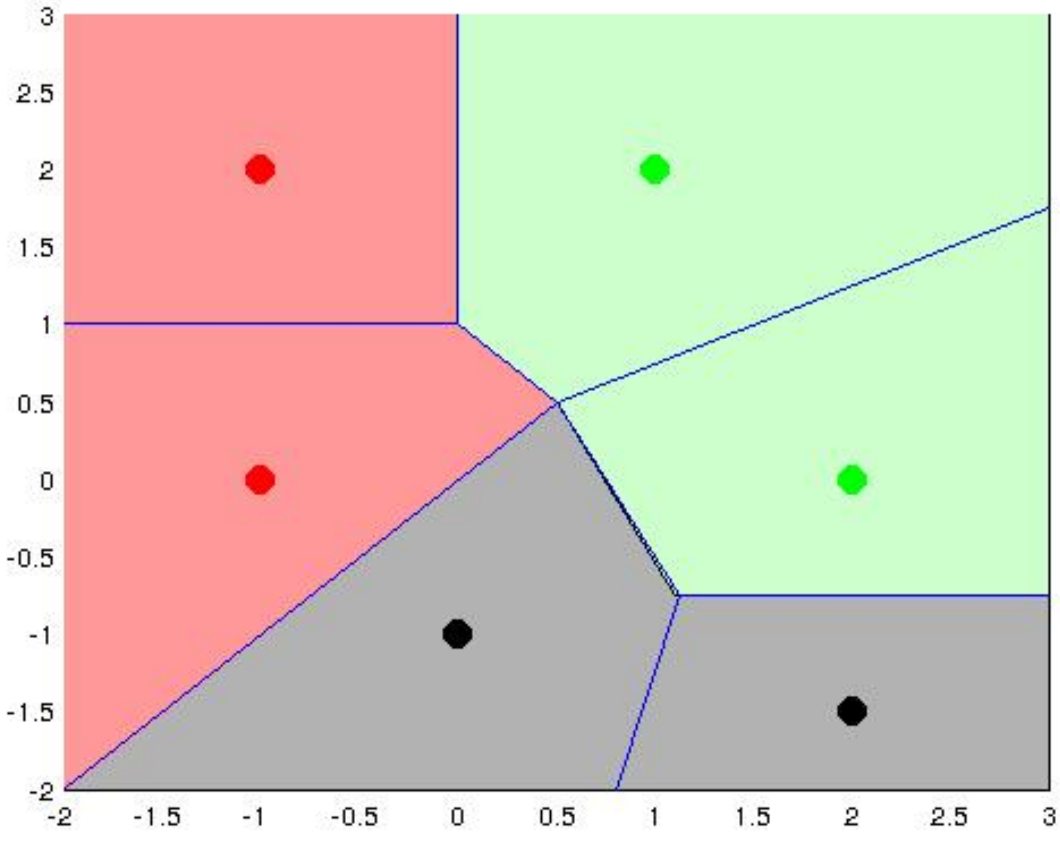
An Idea: NN Extension

- Find nearest point from the original sample



An Idea: NN Extension

- Find nearest point from the original sample

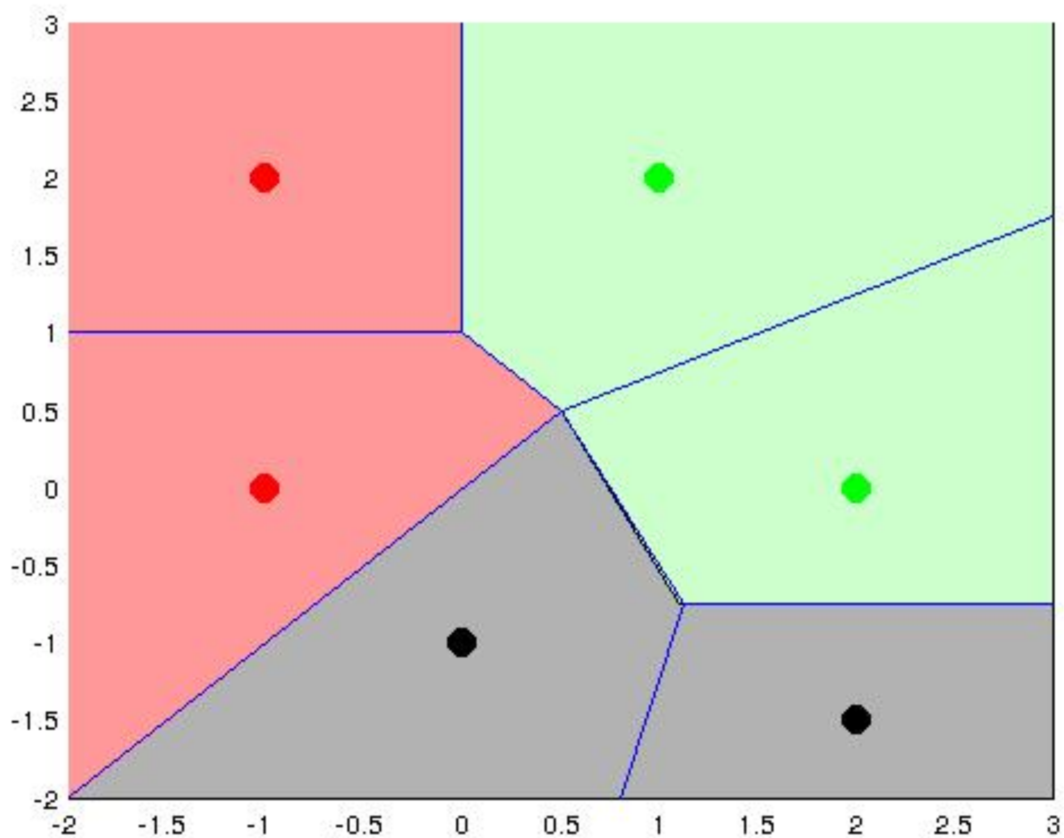


NN Extension of a Clustering

Defined on distribution

Defined on sample

$$\bar{g}_n(x) := g_n(NN_S(x))$$



NN Extension

- Each point represents its Voronoi cell
- Sample level objective:

$$g_n : S \to \binom{k}{p}$$

$$c_n : [k] \to S$$

$$\min_{g_n, c_n} \sum_{j=1}^{n} w_j \Big[\sum_{i \in g_n(x_j)} \|x_j - c_n(i)\| \Big]$$

$$\text{where} \quad w_j = \mathbb{P}_{x \sim \mu}(NN_S(x) = x_j)$$

NN Extension

- Weights are unknown
- Estimate weights from another sample drawn iid from μ.
- Cluster sample with estimated weights
- Use approx algo discussed earlier

NN Dispatch Algorithm

- Draw a second sample *S'* of size *n'*.
- Approximate weights w_j with estimates:

$$\hat{w}_j = \frac{|S' \cap V_j|}{n'}$$

- Find a balanced clustering (g_n, c_n) using estimated weights
- Return its NN extension

$$\bar{g}_n(x) = g_n(NN_S(x))$$

NN Dispatch

- Guarantee: NN Dispatch returns a good clustering of the distribution.
- Sub-optimality depends on
  - Quality of approximation on sample
  - Average 'radius' of Voronoi cell
    $$\alpha(S) = \mathbb{E}_{x\sim\mu}(\|x - NN_S(x)\|)$$
  - Bias from returning clustering that are constant over Voronoi partitions
    $$\beta(S) = \min_{h,c}(Q(\bar{h}, c) - Q(f^*, c^*))$$
    $$\text{s.t. } h \text{ satisfies size constraints } l, L$$

NN Dispatch

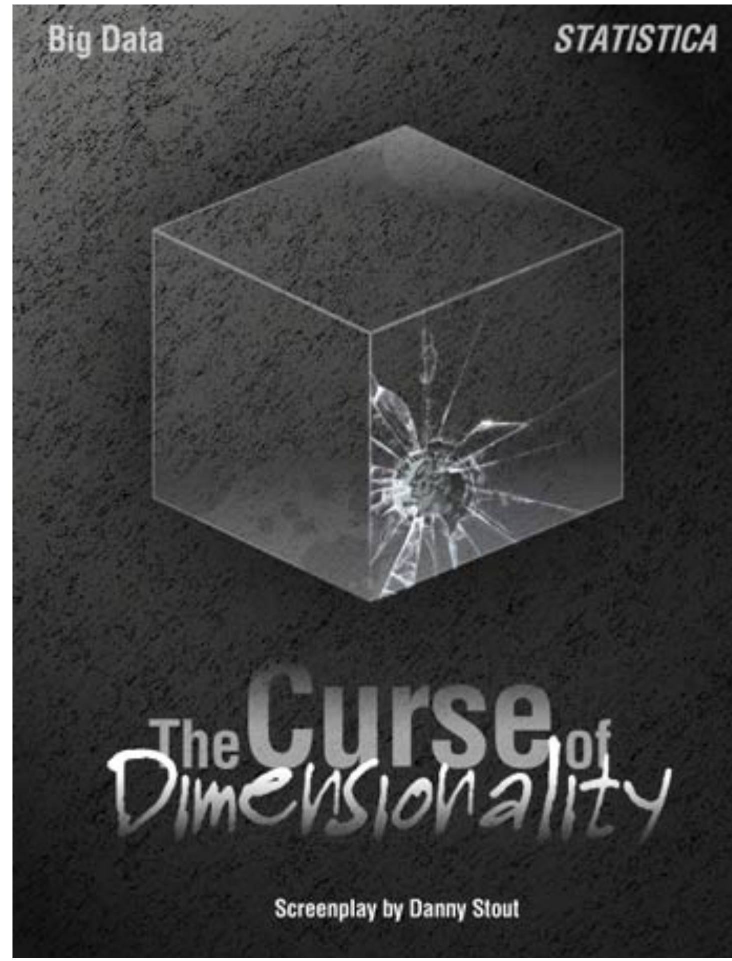
Theorem:

- If $n' = O((n + \ln 1/\delta)/\epsilon^2)$
- Algo on *S* returns solution within $r \cdot \mathcal{OPT} + s$
- Then w.p. $\geq 1 - \delta$
  - $(\bar{g}_n, c_n)$ output satisfies sizes $(l - \epsilon, L + \epsilon)$
  - $$Q(\bar{g}_n, c_n) \leq r \cdot Q(f^*, c^*) + s + 2(r+1)pD\epsilon + p(r+1) \cdot \alpha(S) + r \cdot \beta(S)$$

$$\left(f^*, c^* = \mathcal{OPT}(l + \epsilon, L - \epsilon)\right)$$

NN Dispatch

- Can bound other terms
- Worst case exponential in dimension
  - Curse of dimensionality
- Better bounds with niceness assumptions
  - E.g., Doubling Measure



Experiments

Learning: Approximations

Balanced Clustering	• K-means++, with rebalancing
NN Dispatch	• Estimated weight = 1/n • Random Partition Trees for Approximate NN Search

Algorithm

- Cluster a small sample
- Extend the clustering to the rest of the training set with NN Dispatch
- Learn
  - independent model for each cluster or
  - in tandem, with partial or complete communication
- Testing
  - Query the appropriate model with NN Dispatch

Learning

- No communication:
  - Each cluster learns an independent model
  - Embarrassingly parallel
- Compare against:
  - Random partitioning with no communication
  - Random partitioning with full communication (global model)

Experimental Setup

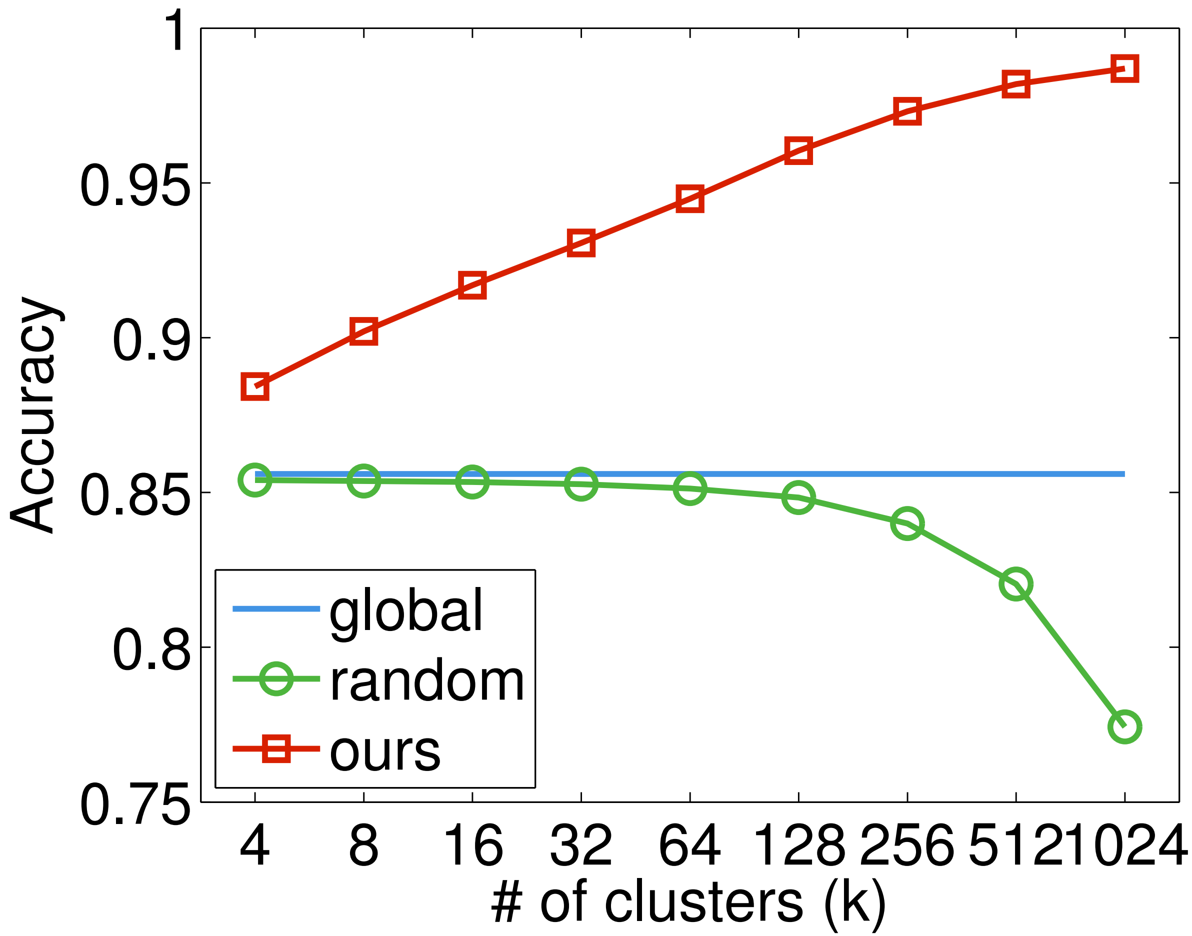
- Run on a cluster with
  - 15 machines
  - 8 cores per machine, each of 2.4GHz
  - 32 GB shared memory per machine

Datasets

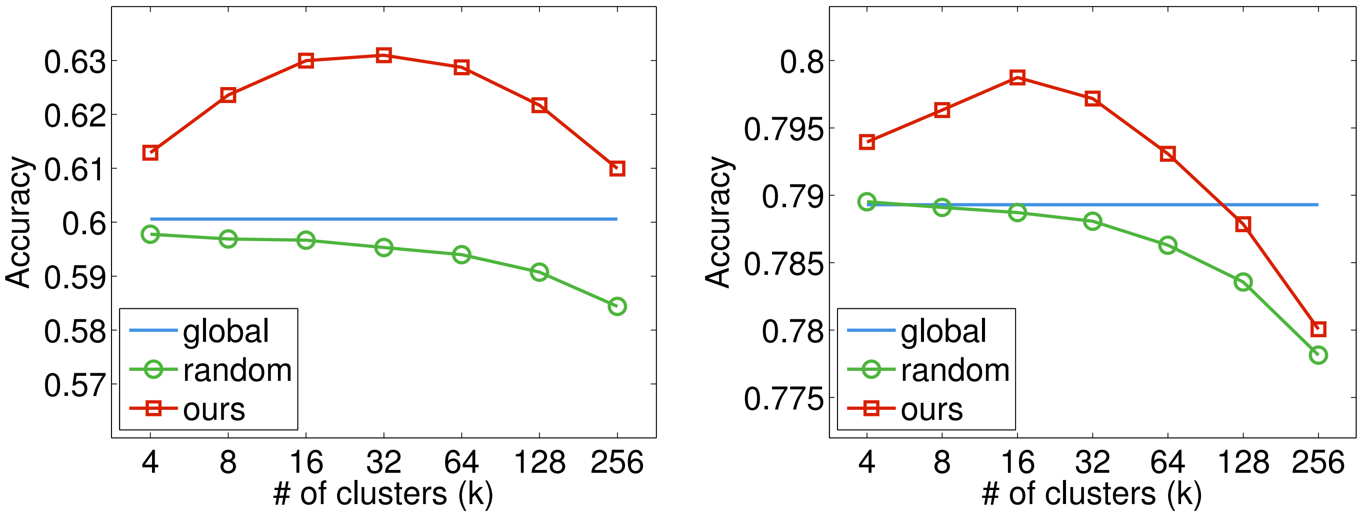
Datasets	Number of examples	Dimensionality
MNIST-8M	8 million	784
CIFAR-10-early	2.5 million	160
CIFAR-10-late	2.5 million	144
CTRc	0.8 million	232
CTRa	0.3 million	13 million
Criteo-Kaggle	45 million	34 million

CTR: Click Through Rate

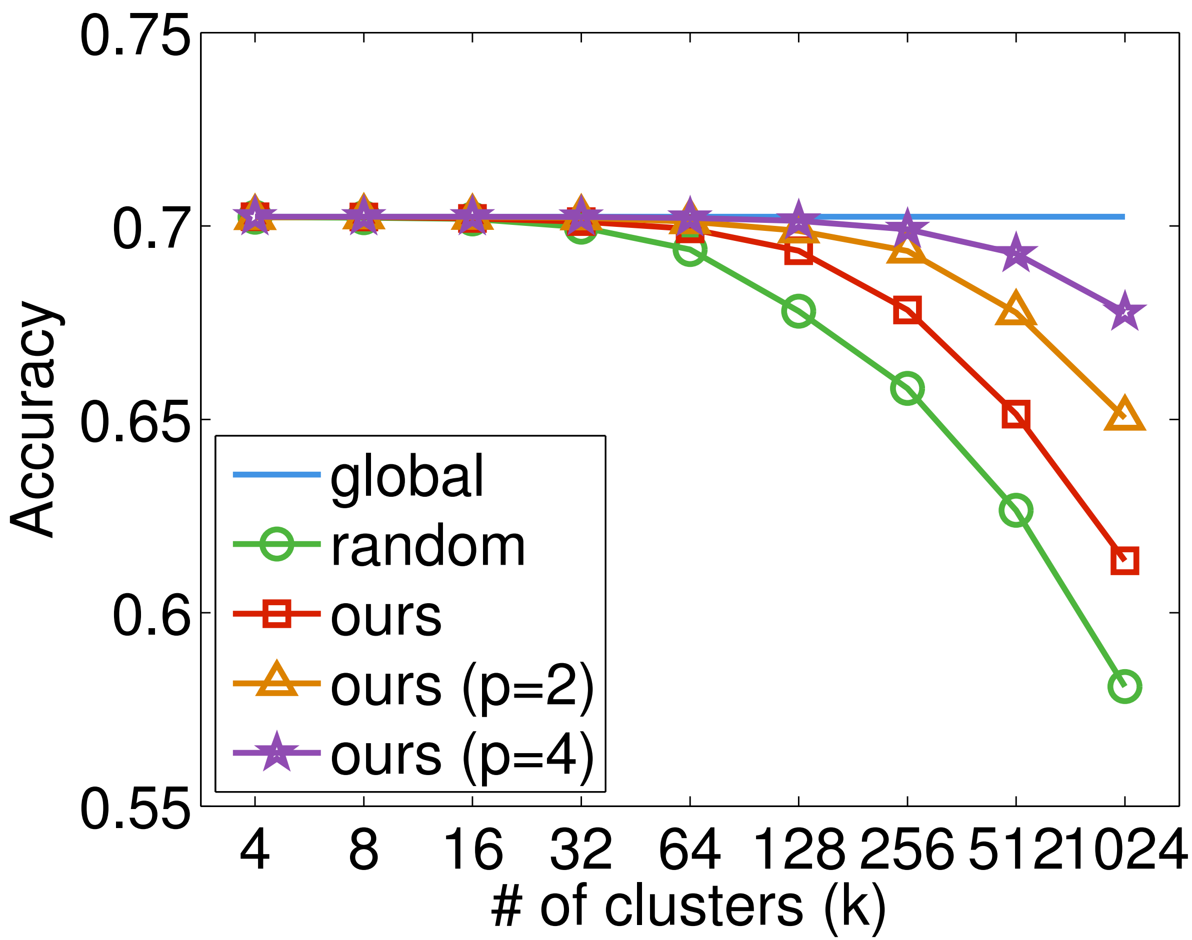
Learning with no communication: MNIST-8M



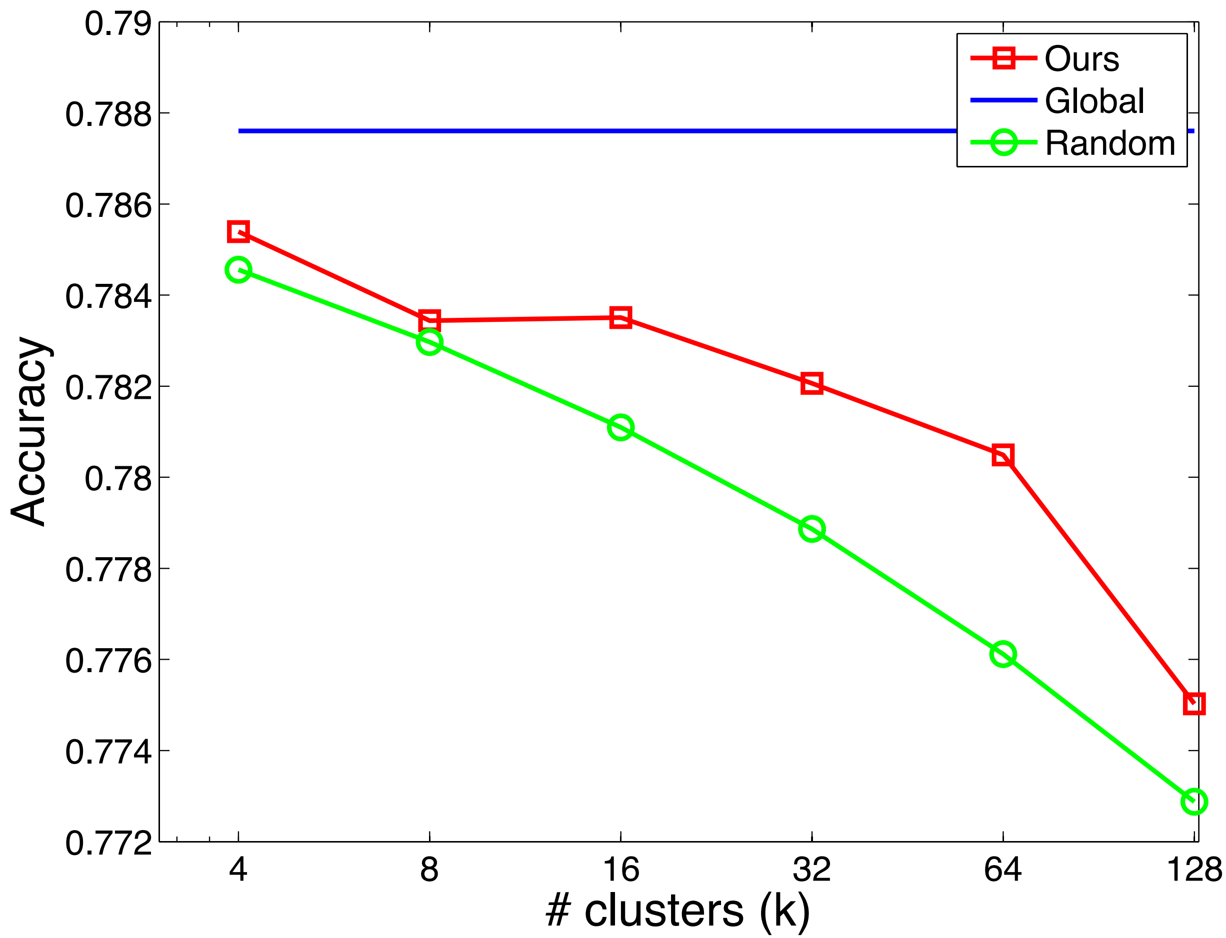
Learning with no communication: CIFAR-10



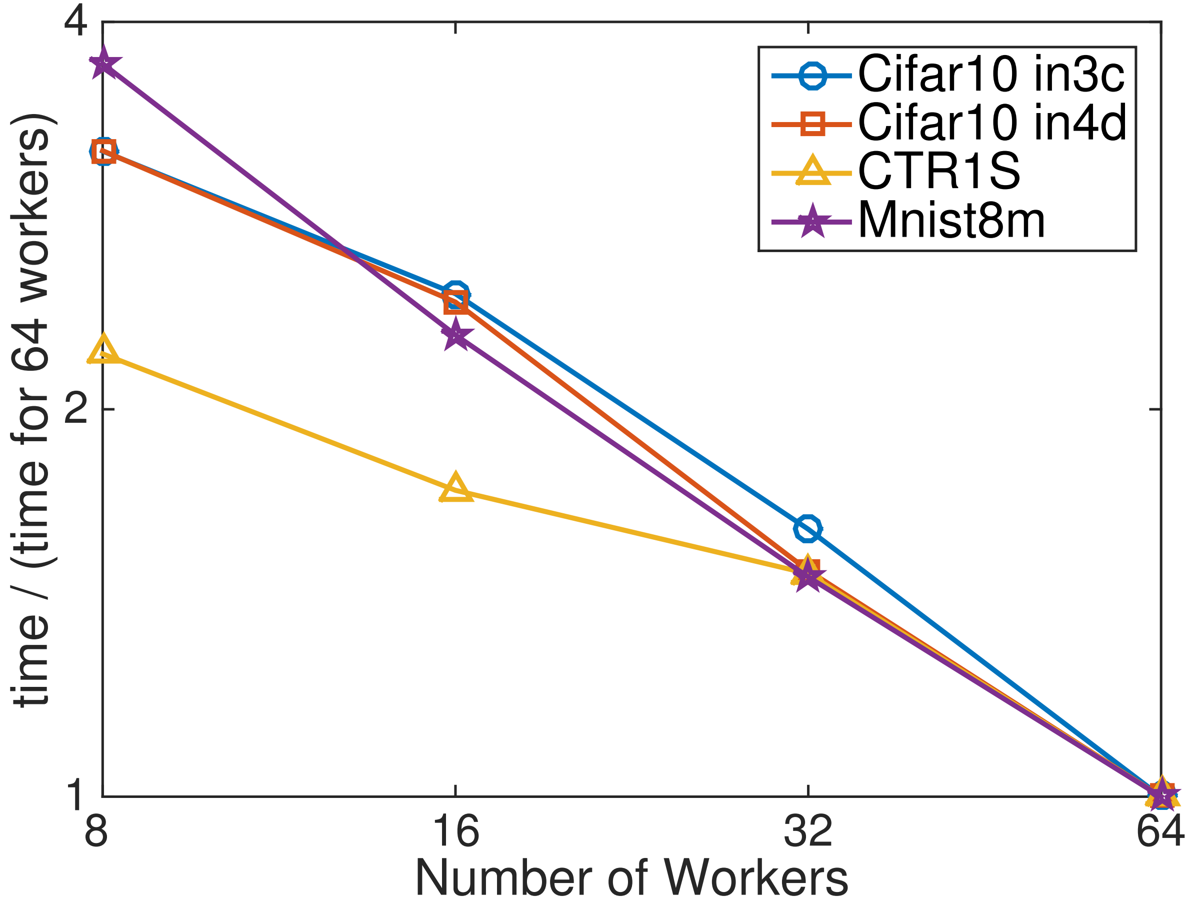
Learning with no communication: CTRc



Learning with no communication: Criteo-Kaggle

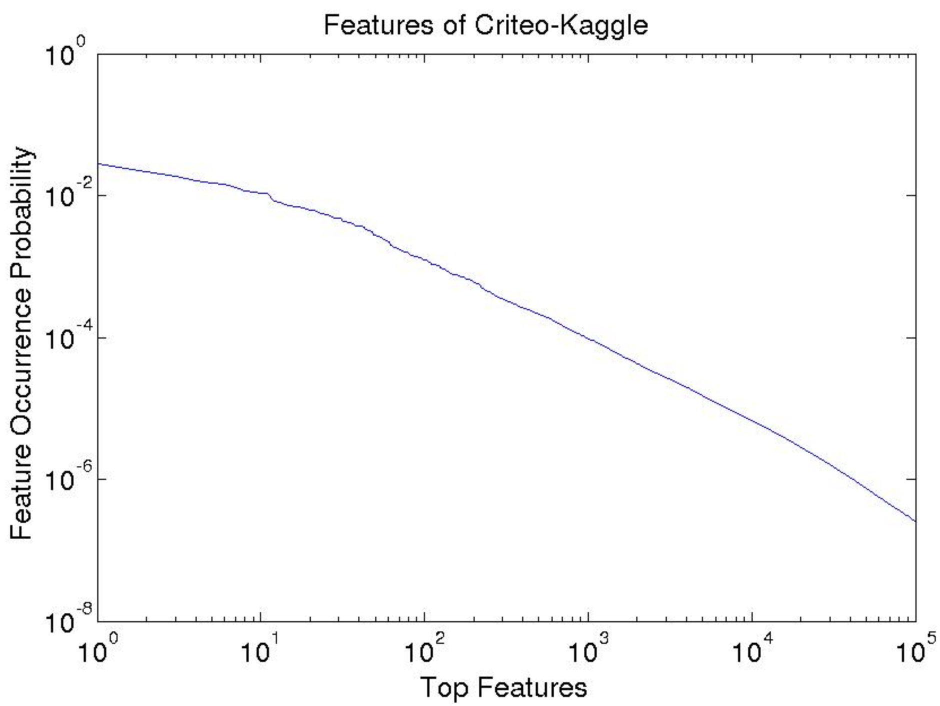


Learning with no communication: Scalability



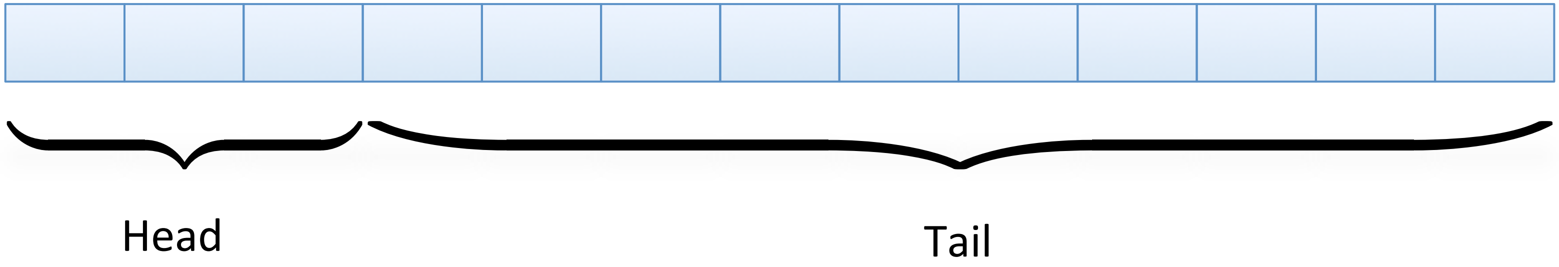
Learning with communication

- High dimensional datasets
- Feature occurrence: approx. power law



Learning with communication

- Tail features cannot be reliably learnt
- Scheme 1: Synchronize on tail features only across all clusters
- Scheme 2: Synchronize on all features, also store a local correction for head features
- Asynchronous Stochastic Gradient Descent



Scheme 1: Partial Communication

- Local model for head and synchronized model for the tail

$$f(\mathbf{x}) = \mathbf{w}_{i(\mathbf{x})} \cdot \mathbf{x}_h + \mathbf{w}_t \cdot \mathbf{x}_t$$

- Communication: not very high for relatively small size of head

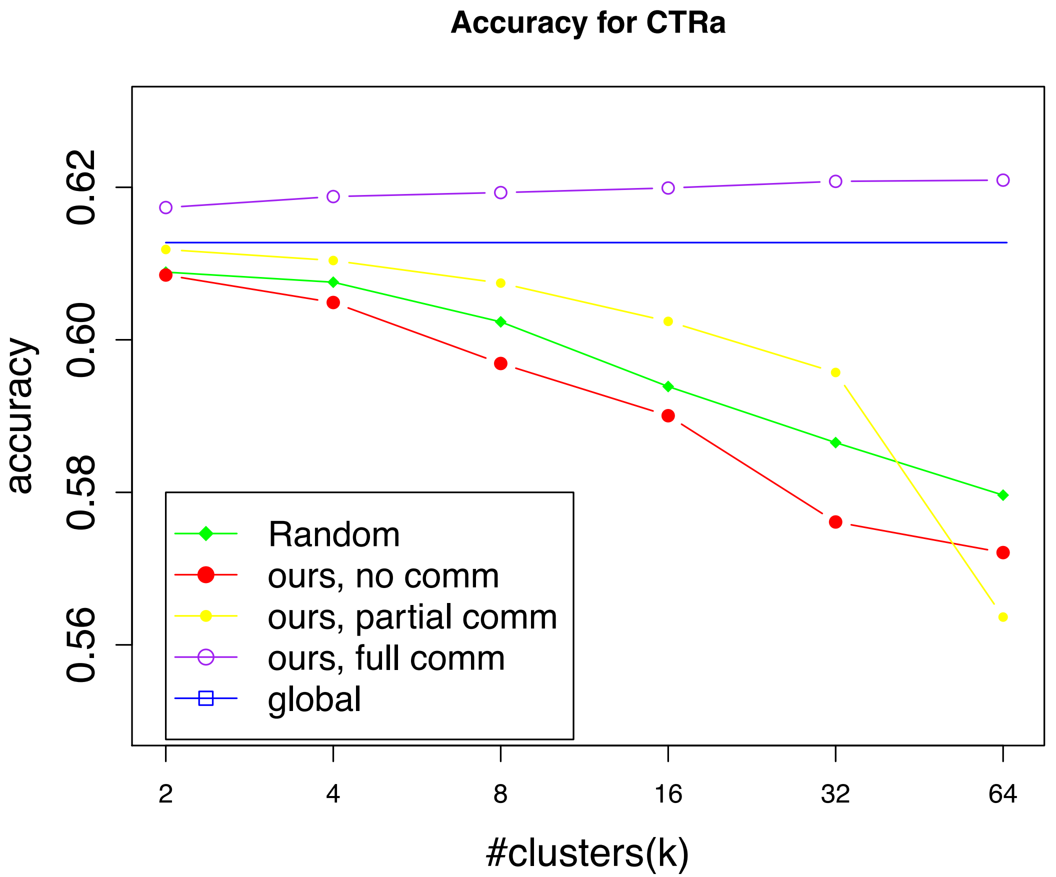
Scheme 2: Full communication

- Each cluster stores a "correction" to the globally synchronized model

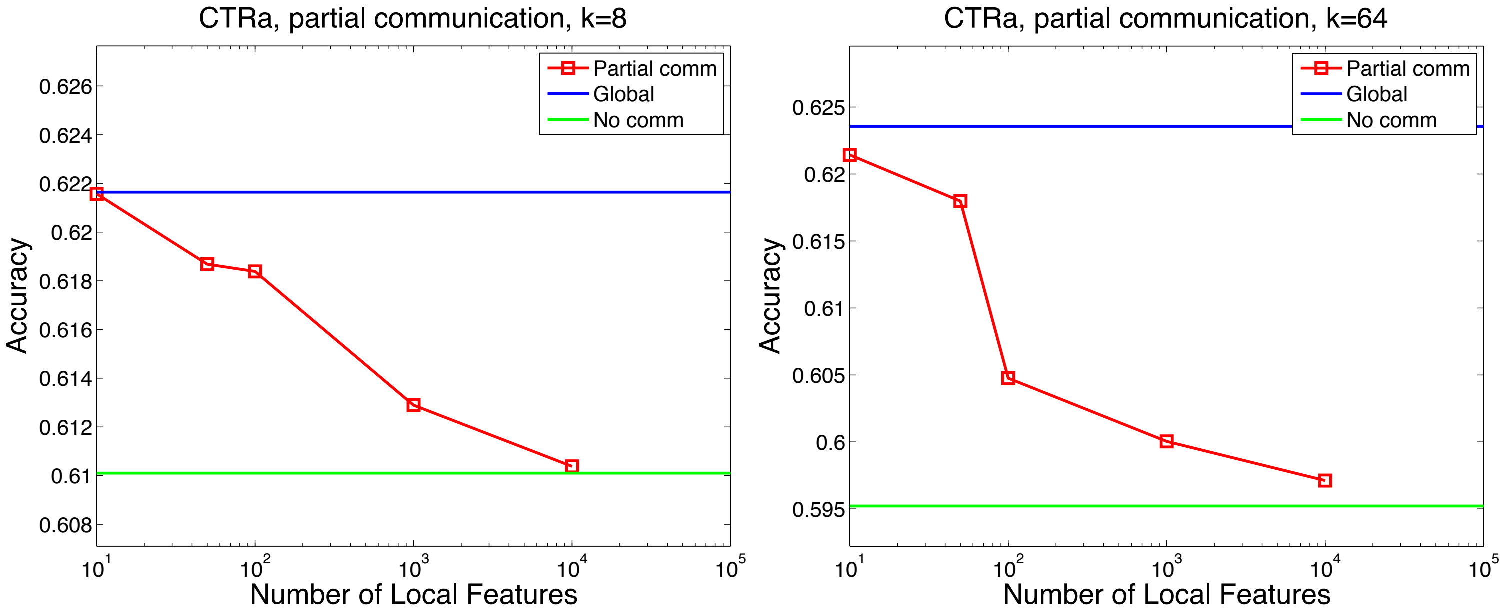
$$f(\mathbf{x}) = \mathbf{w}_{i(\mathbf{x})} \cdot \mathbf{x}_h + \mathbf{w}_g \cdot \mathbf{x}$$

- Communication: Equal to communication of fully synchronized global model

Performance on CTR data



How many local features?



Which scheme should I use?

- Dense data, images: No communication
- High dimensional data: With communication

Conclusion

- Data-dependent partitioning is good in both theory and practice!
- Balanced Clustering
- Nearest Neighbor Extension
- Experimental Evaluation

Thank You! Questions?

Collaborators