Semi-Supervised Learning

William Cohen

Outline

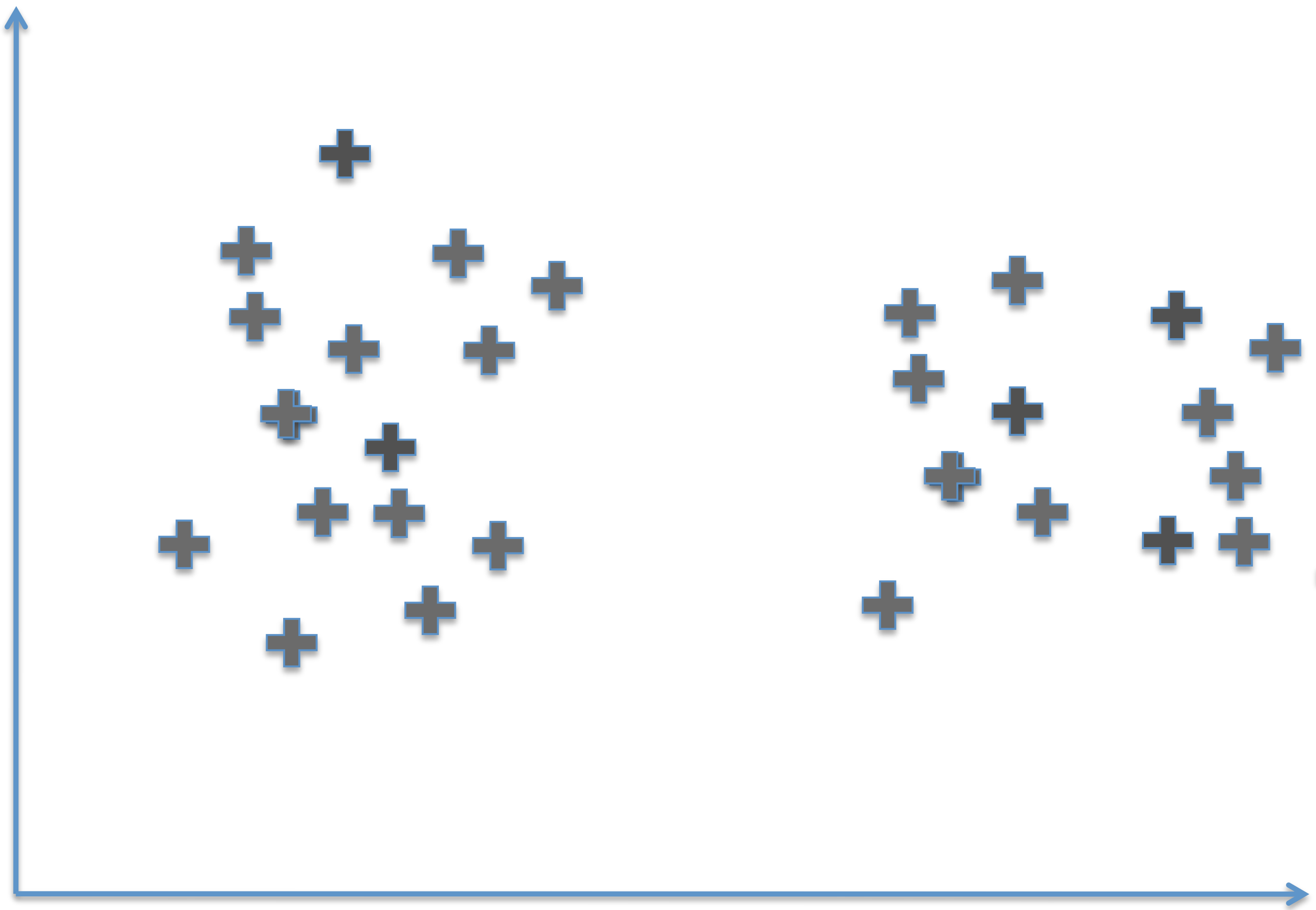
- The general idea and an example (NELL)
- Some types of SSL
  - Margin-based: transductive SVM
    - Logistic regression with entropic regularization
  - Generative: seeded k-means
  - Nearest-neighbor like: graph-based SSL

INTRO TO SEMI-SUPERVISED LEARNING (SSL)

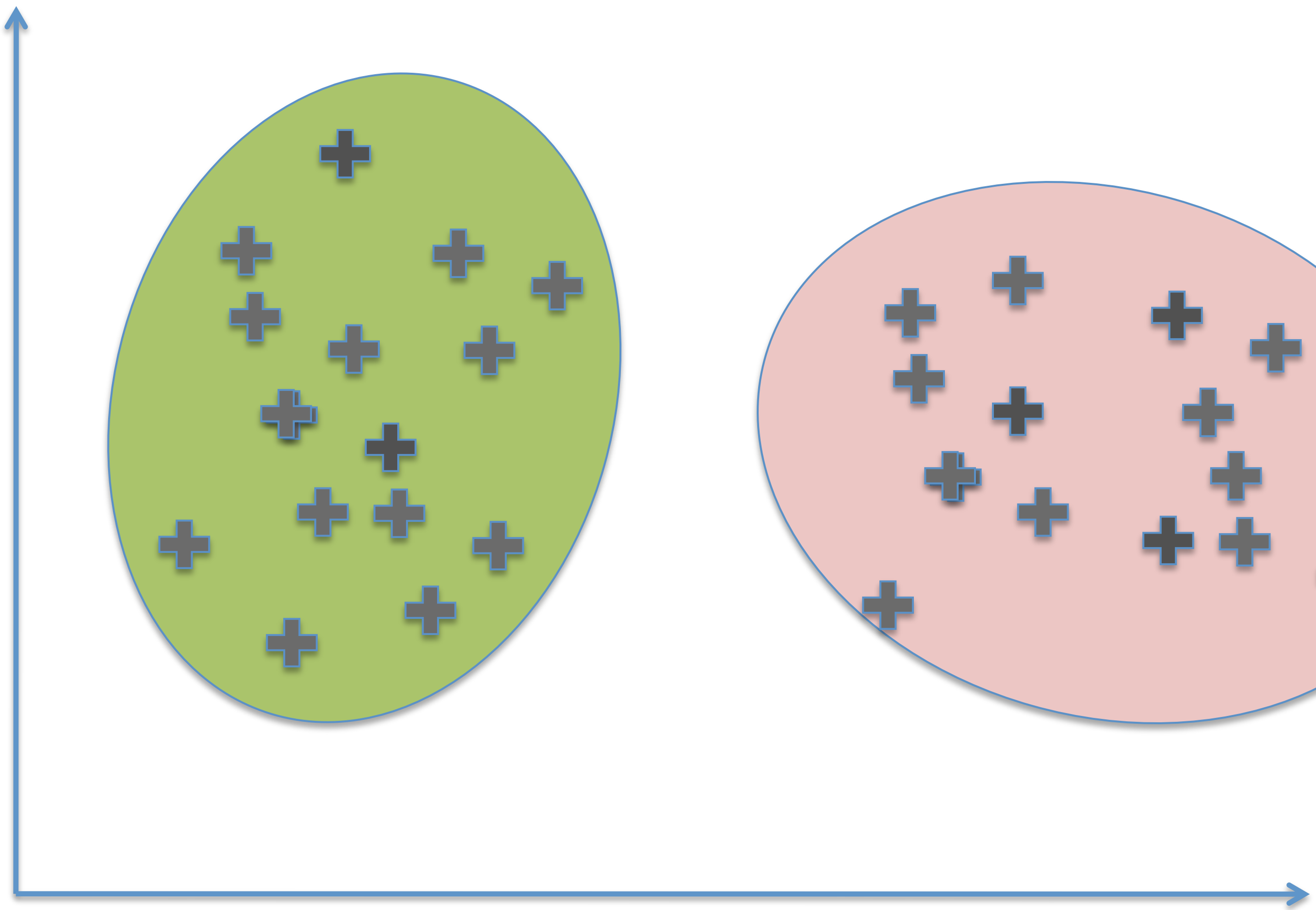
Semi-supervised learning

- Given:
  - A pool of labeled examples L
  - A (usually larger) pool of unlabeled examples U
- Option 1 for using L and U :
  - Ignore U and use supervised learning on L
- Option 2:
  - Ignore labels in L+U and use k-means, etc find clusters; then label each cluster using L
- Question:
  - Can you use both L and U to do better?

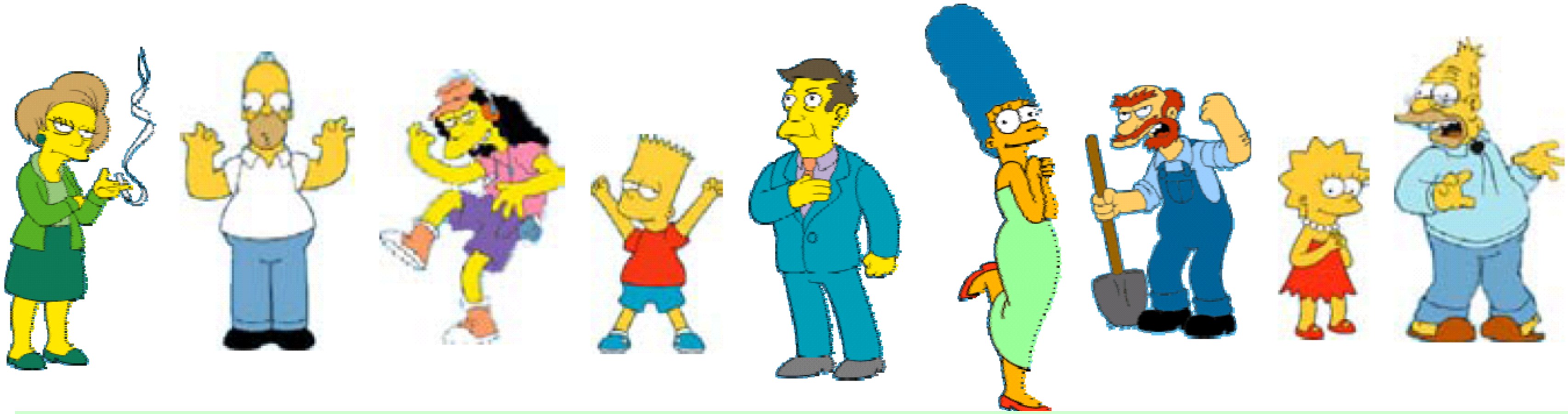
SSL is Somewhere Between Clustering and Supervised Learning

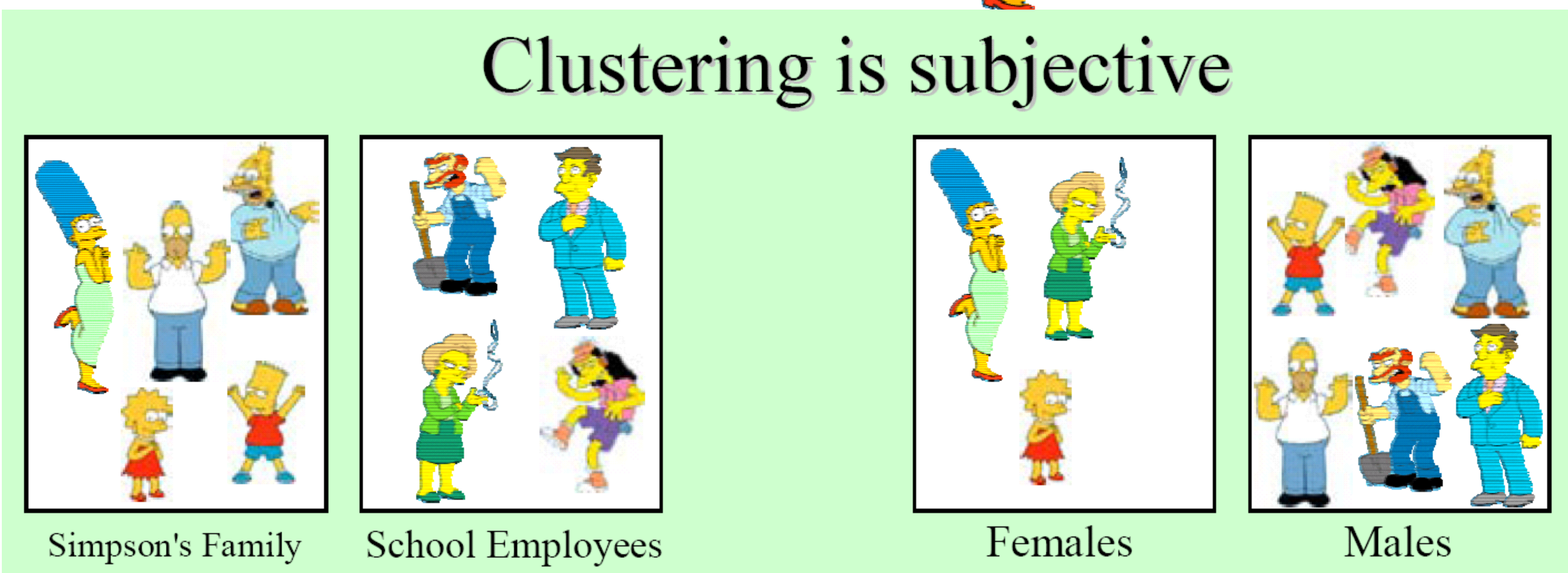


SSL is Between Clustering and SL



What is a natural grouping among these objects?

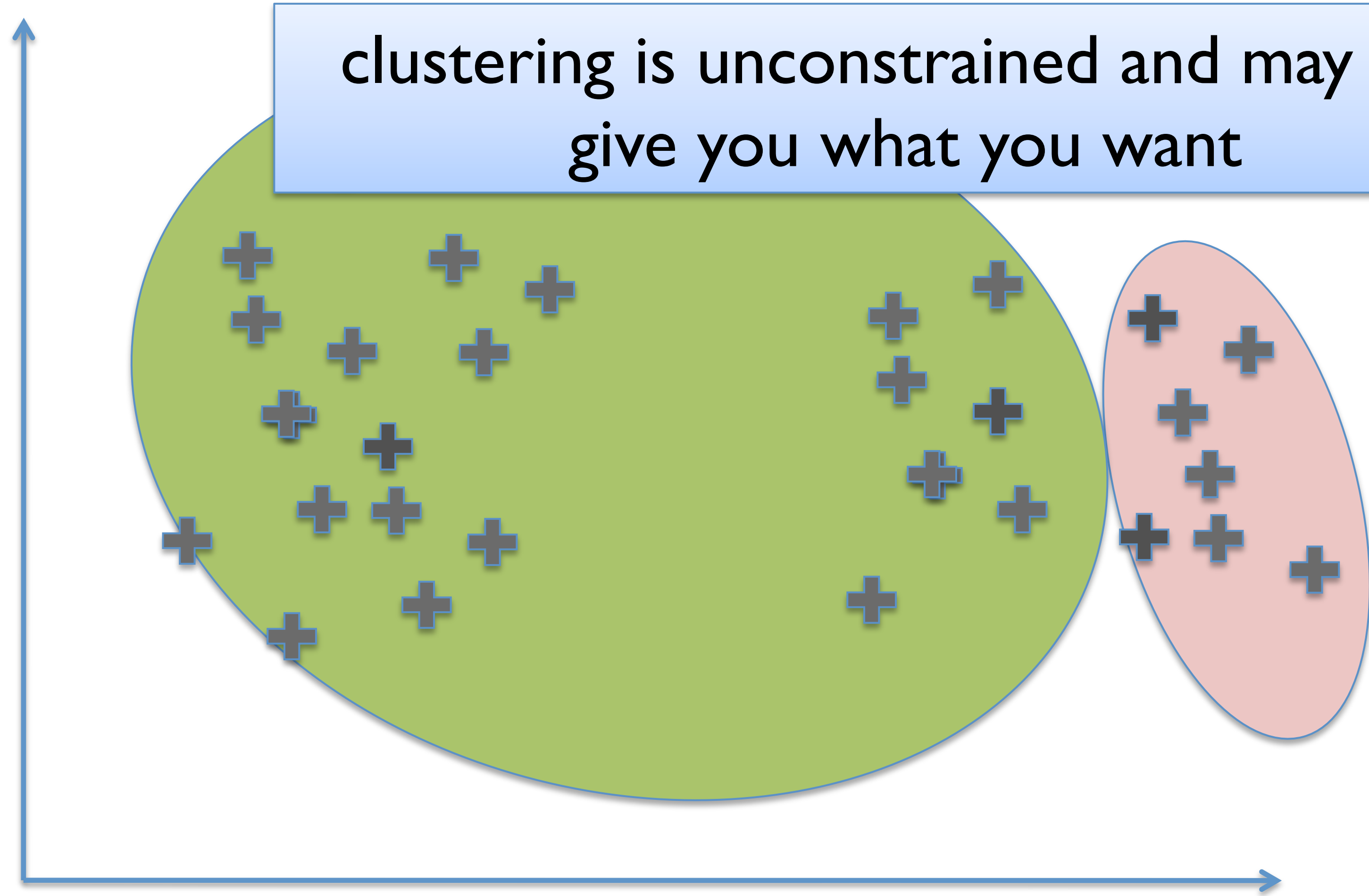




slides: Bhavana Dalvi

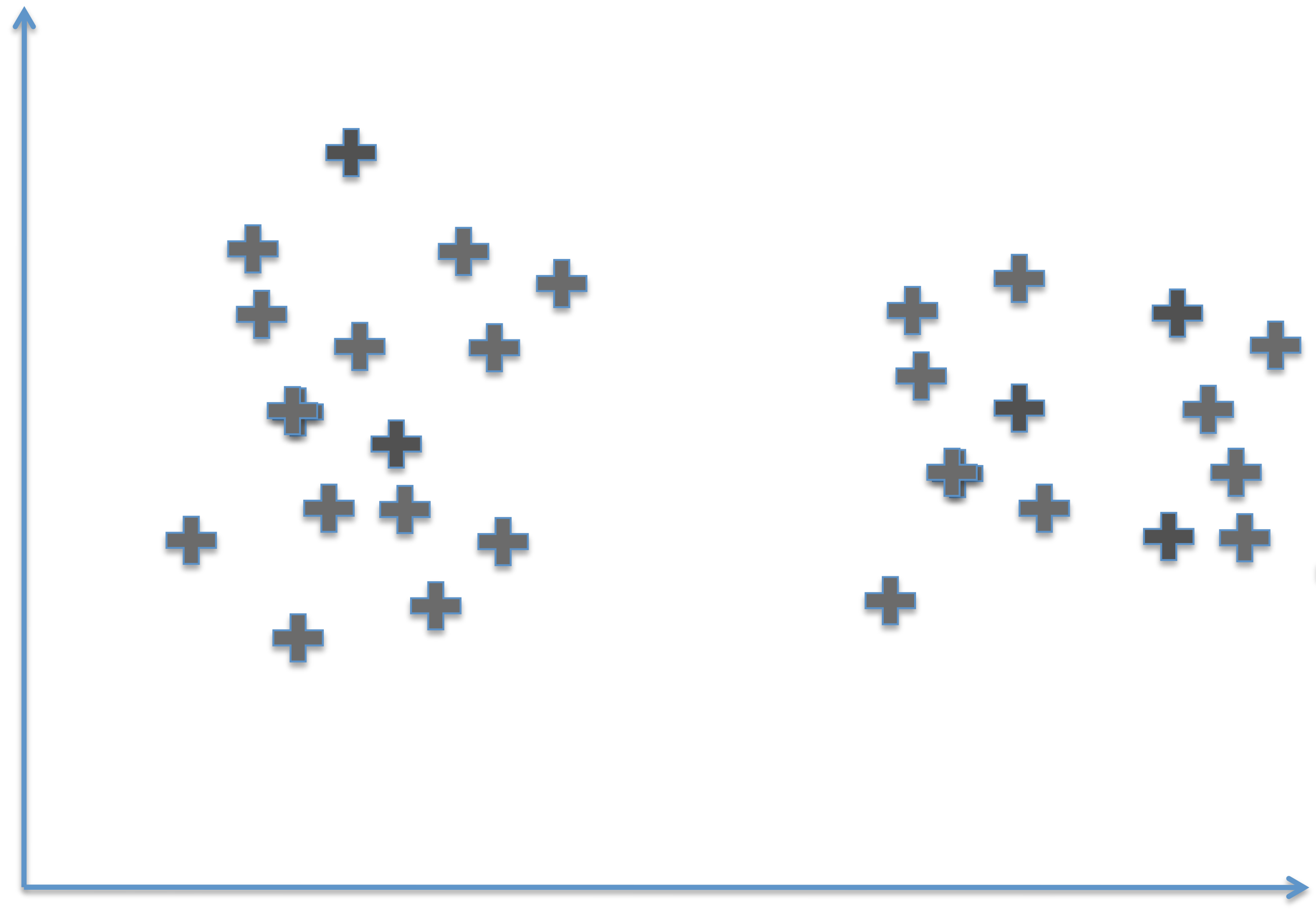
SSL is Between Clustering and SL

clustering is unconstrained and may not give you what you want

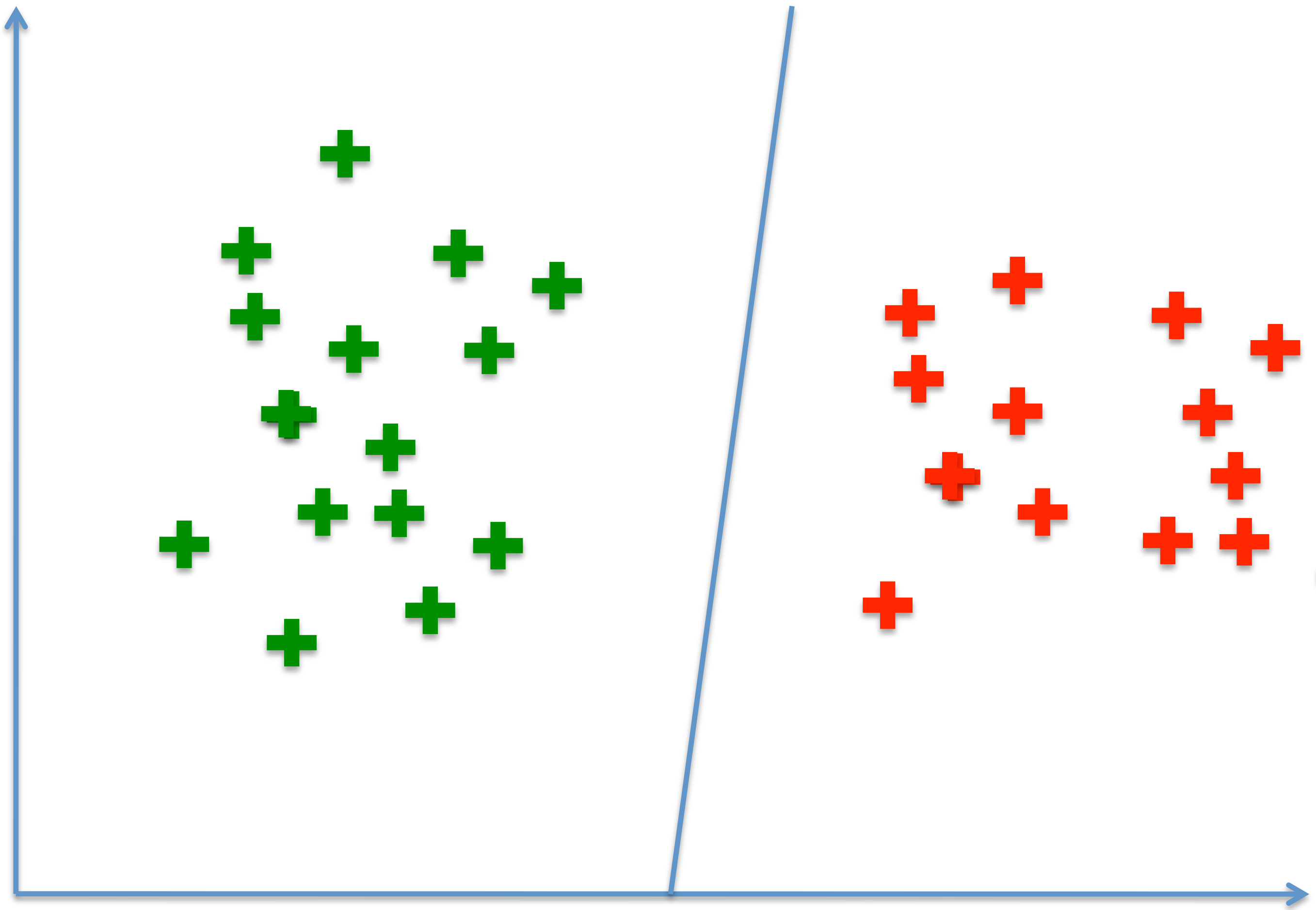


maybe this clustering is as good as the other

SSL is Between Clustering and SL

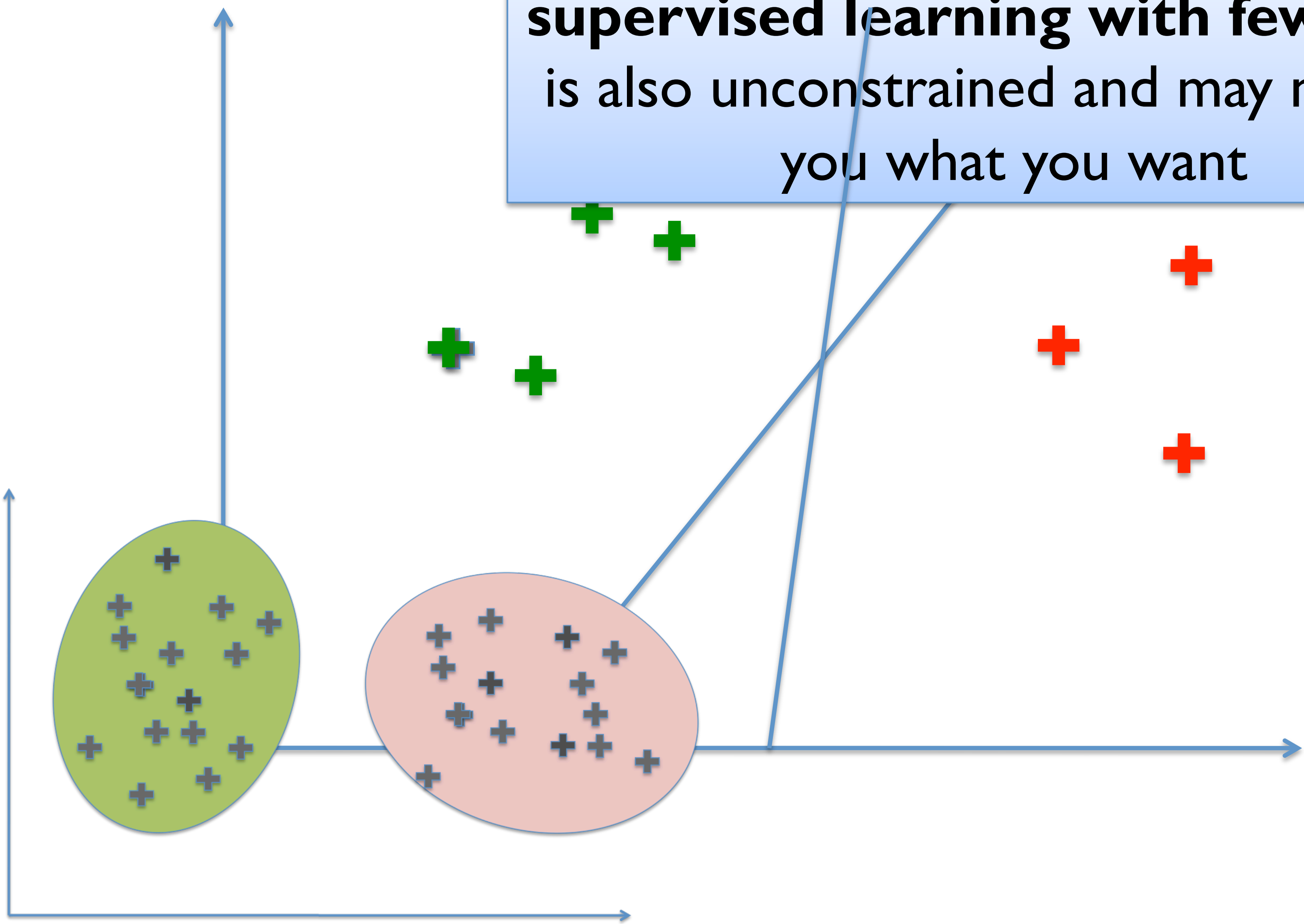


SSL is Between Clustering and SL

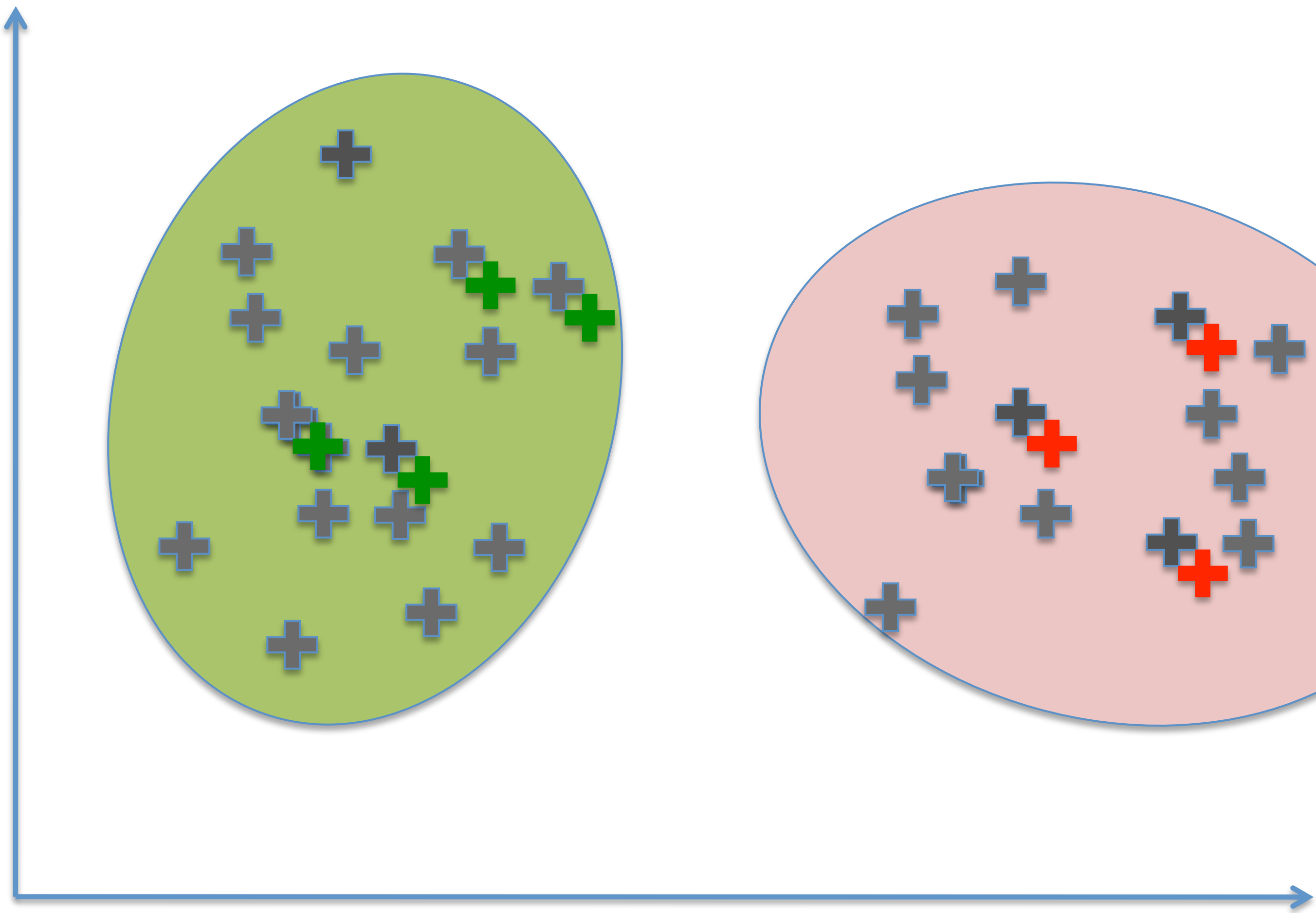


SSL is Between Clustering and SL

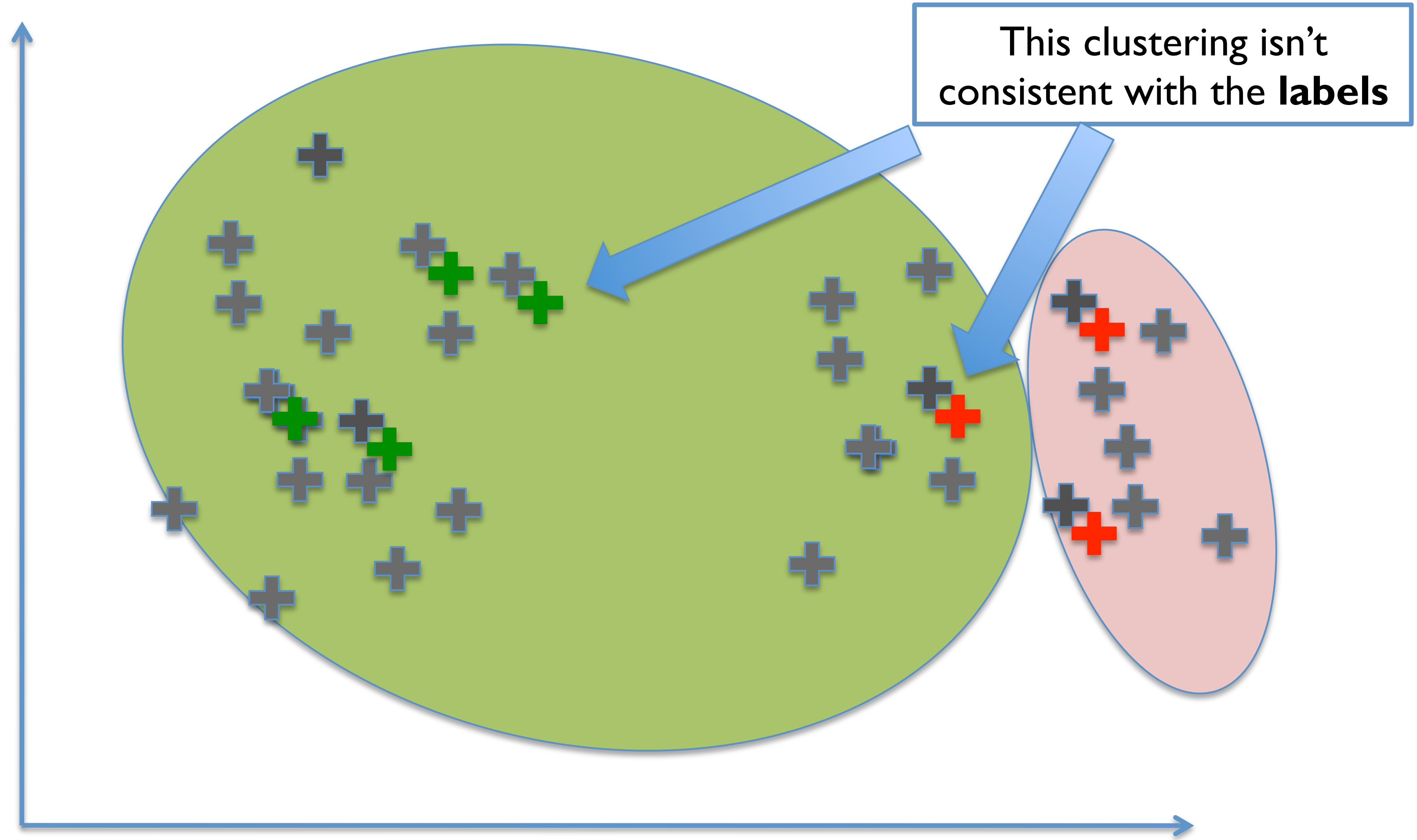
supervised learning with few labels is also unconstrained and may not give you what you want



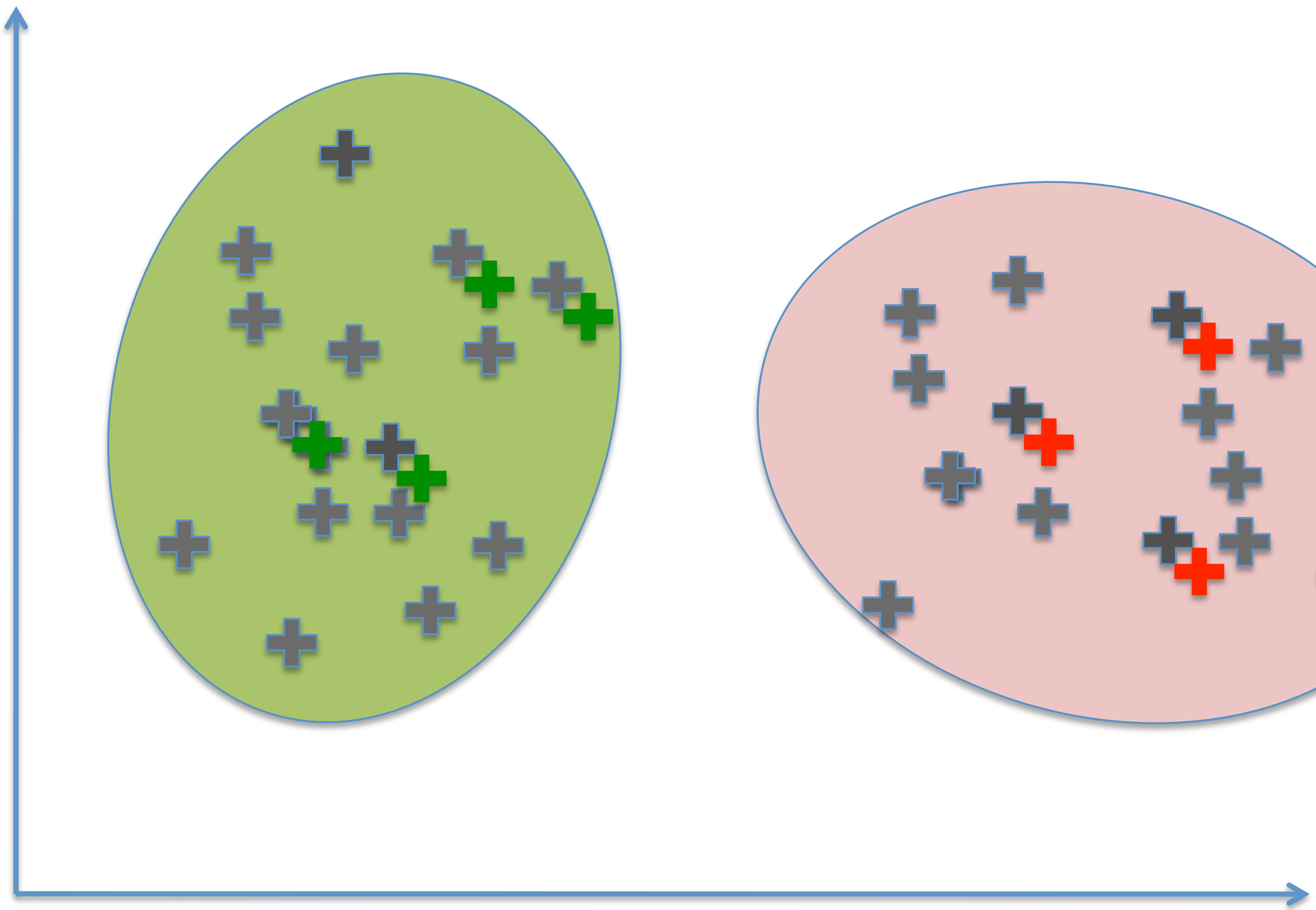
SSL is Between Clustering and SL



SSL is Between Clustering and SL



SSL is Between Clustering and SL



|Predicted Green|/|U| ~= 50%

SSL in Action: The NELL System

Outline

- The general idea and an example (NELL)
- Some types of SSL
  - Margin-based: transductive SVM
    - Logistic regression with entropic regularization
  - Generative: seeded k-means
  - Nearest-neighbor like: graph-based SSL

TRANSDUCTIVE SVM

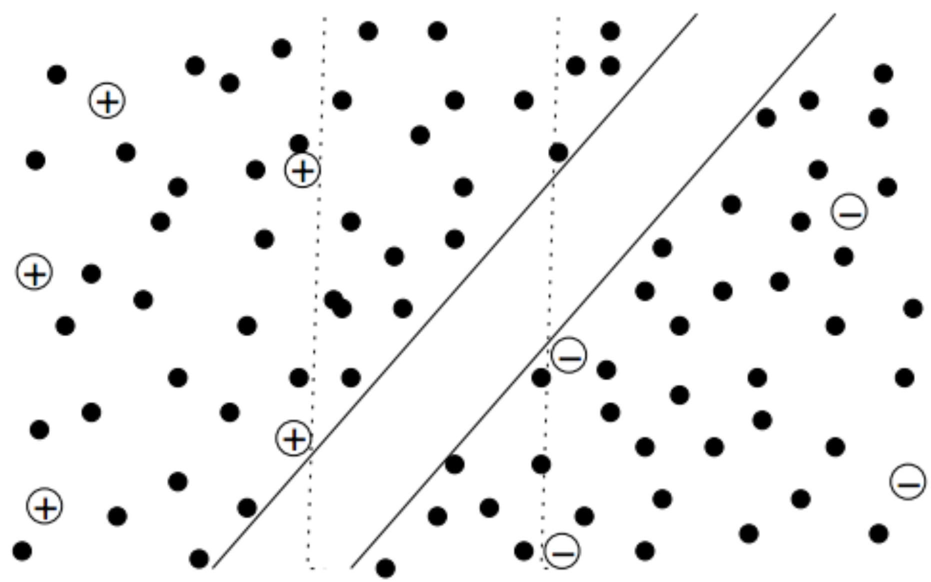
Two Kinds of Learning

- **Inductive** SSL:
  - Input: training set
    - $(x_1,y_1),\ldots,(x_n,y_n)$
    - $x_{n+1}, x_{n+2}, \ldots, x_{n+m}$
  - Output: classifier
    - $f(x) = y$
  - Classifier can be run on any test example *x*

- **Transductive** SSL:
  - Input: training set
    - $(x_1,y_1),\ldots,(x_n,y_n)$
    - $x_{n+1}, x_{n+2}, \ldots, x_{n+m}$
  - Output: classifier
    - $f(x_i) = y$
  - Classifier is only defined for x_i's *seen at training time*

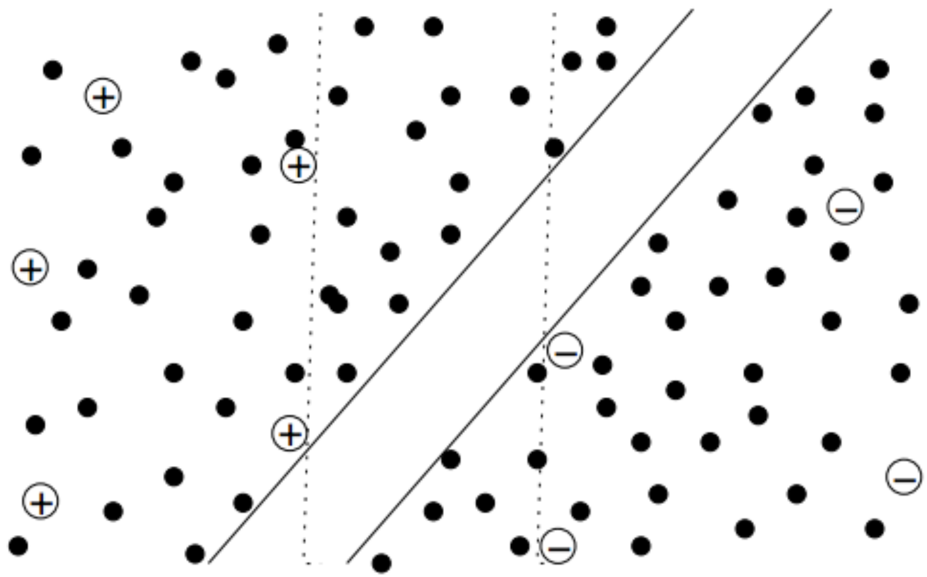
Transductive Support Vector Machines

Instead of finding maximum margin between labelled points, optimize over both margin and labels of unlabelled points.



Transductive Support Vector Machines

Instead of finding maximum margin between labelled points, optimize over both margin and labels of unlabelled points.



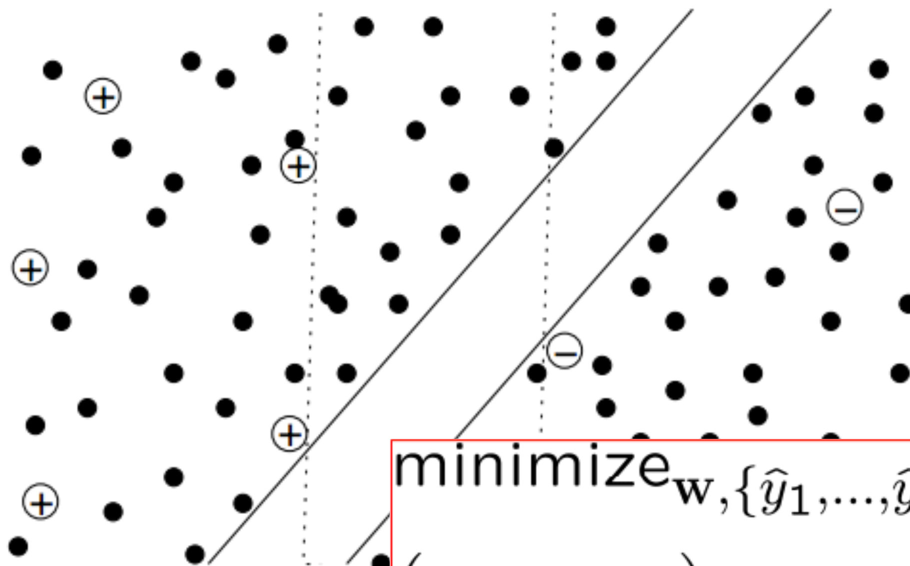
$$\text{minimize}_{\mathbf{w}} \quad \mathbf{w}.\mathbf{w}$$

Standard SVM

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \;\; \forall j$$

Transductive Support Vector Machines

Instead of finding maximum margin between labelled points, optimize over both margin and labels of unlabelled points.



$$\text{minimize}_{\mathbf{w},\{\widehat{y}_1,\ldots,\widehat{y}_{n_U}\}} \quad \mathbf{w}.\mathbf{w}$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \ \forall j = 1, \ldots, n_L$$

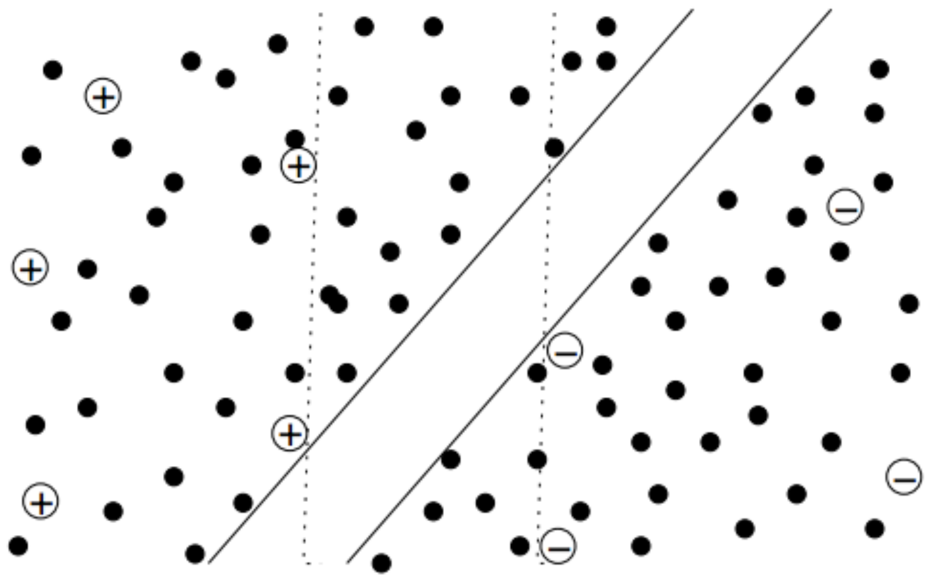
$$(\mathbf{w}.\mathbf{x}_u + b)\, \widehat{y}_u \geq 1, \ \forall u = 1, \ldots, n_U$$

$$\widehat{y}_u \in \{-1, +1\}, \ \forall u = 1, \ldots, n_U$$

Tranductive SVM

Transductive Support Vector Machines

Instead of finding maximum margin between labelled points, optimize over both margin and labels of unlabelled points.



Not a convex problem – need to do some sort of search to guess the labels for the unlabeled examples

SSL using regularized SGD for logistic regression

1. P(y|**x**)=logistic***(*****x . w)**
2. Define loss function

$$\mathrm{LCL}_D(\mathbf{w}) \equiv \sum_i \log P(y_i \mid \mathbf{x}_i, \mathbf{w}) - \boxed{\mu \|\mathbf{w}\|_2^2}$$

3. Differentiate the function and use gradient descent to learn

SSL using regularized SGD for logistic regression

1. P(y|**x**)=logistic(**x . w**)
2. Define loss function

$$\mathrm{LCL}_D(\mathbf{w}) \equiv \sum_i \log P(y_i \mid \mathbf{x}_i, \mathbf{w}) - \mu \|\mathbf{w}\|_2^2$$

$$-\sum_j \boxed{\sum_{y'} P(y' \mid \mathbf{x}_j, \mathbf{w}) \log P(y' \mid \mathbf{x}_j, \mathbf{w})}$$

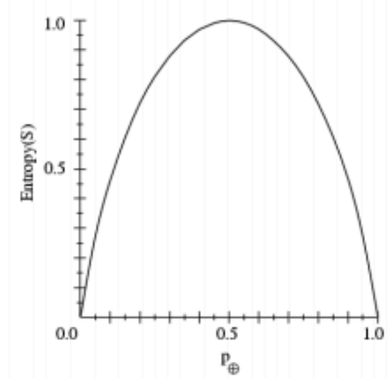
Entropy of predictions on the unlabeled examples

3. D

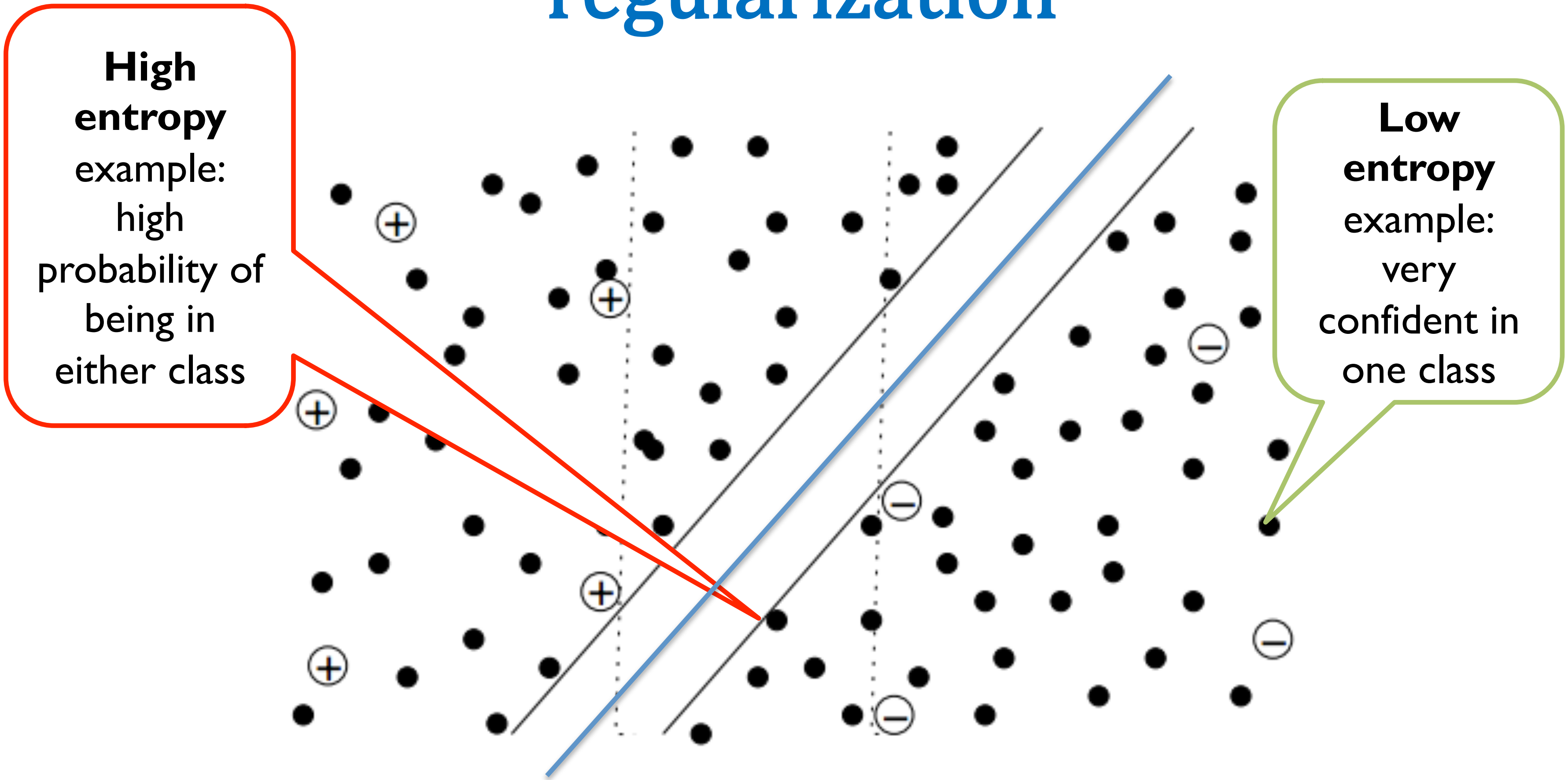
Sample Entropy of a Labeled Dataset

- S is a sample of training examples
- $p_\oplus$ is the proportion of positive examples in S.
- $p_\ominus$ is the proportion of negative examples in S.
- Entropy measures the impurity of S.

$$H(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$



Logistic regression with entropic regularization



Again, a convex problem – need to do some sort of search to guess the labels for the unlabeled examples

SEMI-SUPERVISED
K-MEANS AND MIXTURE MODELS

k-means

Common Heuristic: The Lloyd's method

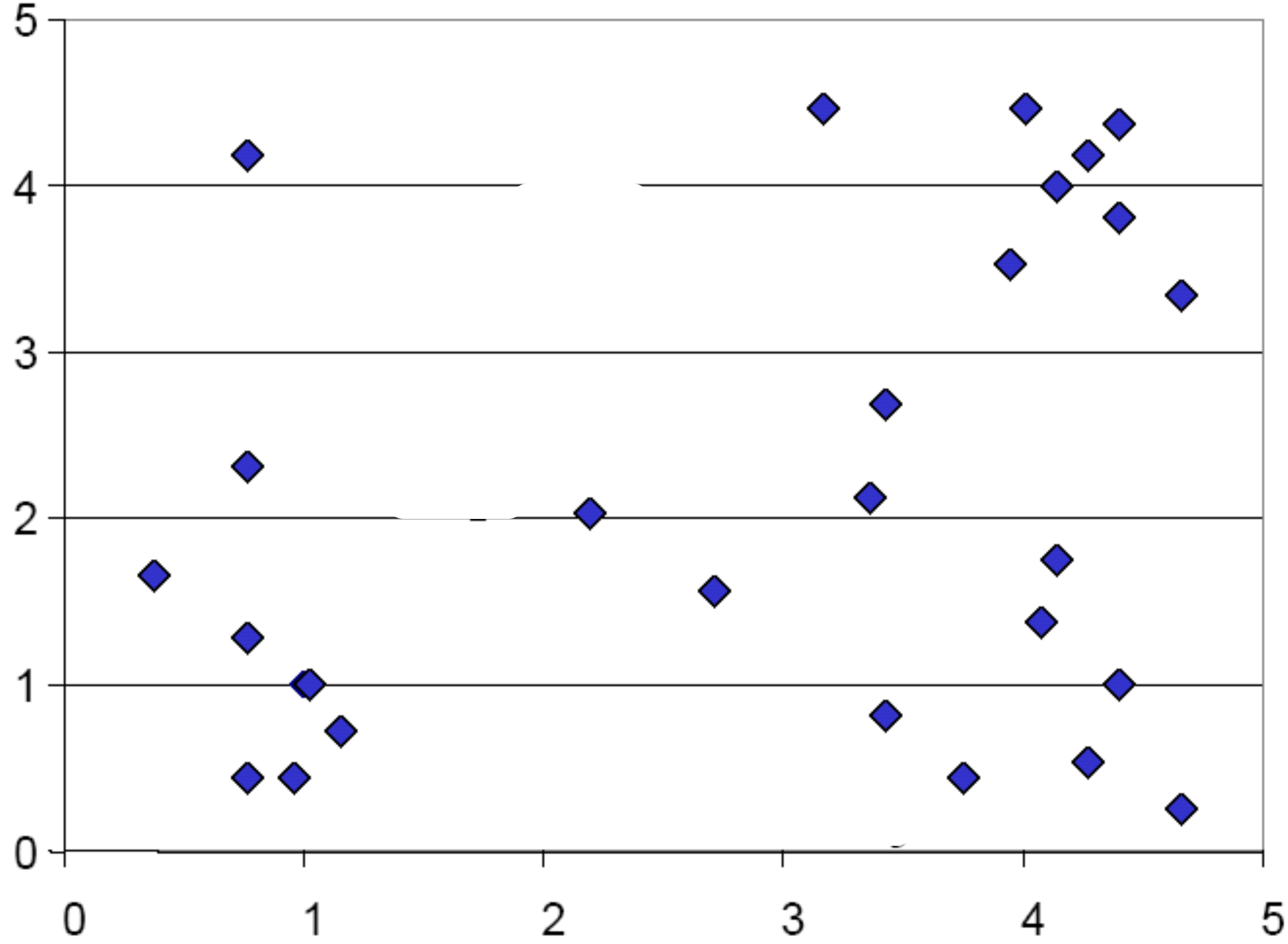
Input: A set of n datapoints $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ in R^d

Initialize centers $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k \in R^d$ and
clusters $C_1, C_2, \dots, C_k$ in any way.

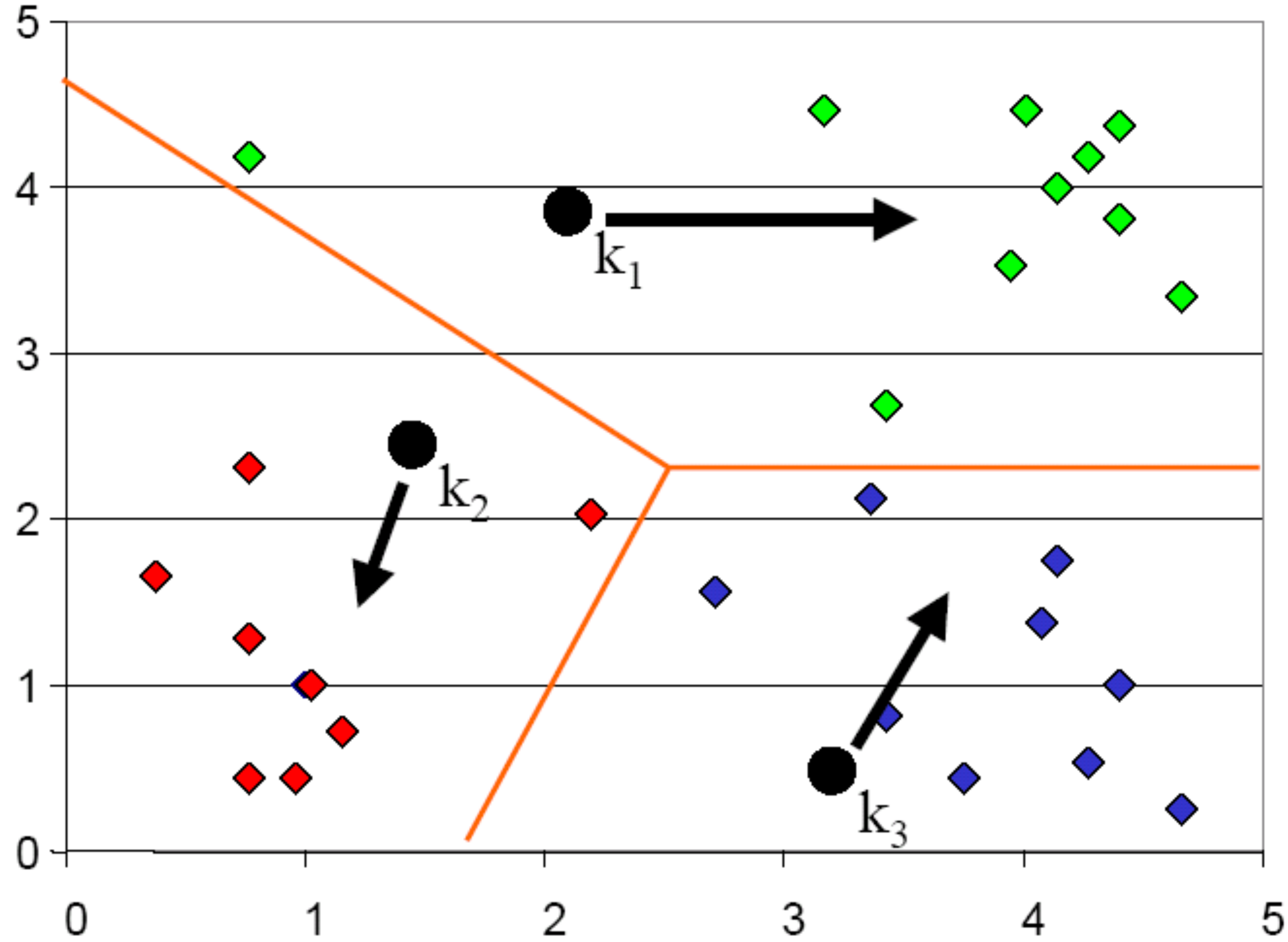
Repeat until there is no further change in the cost.

- For each j: $C_j \leftarrow \{x \in S$ whose closest center is $\mathbf{c}_j\}$
- For each j: $\mathbf{c}_j \leftarrow$mean of C_j

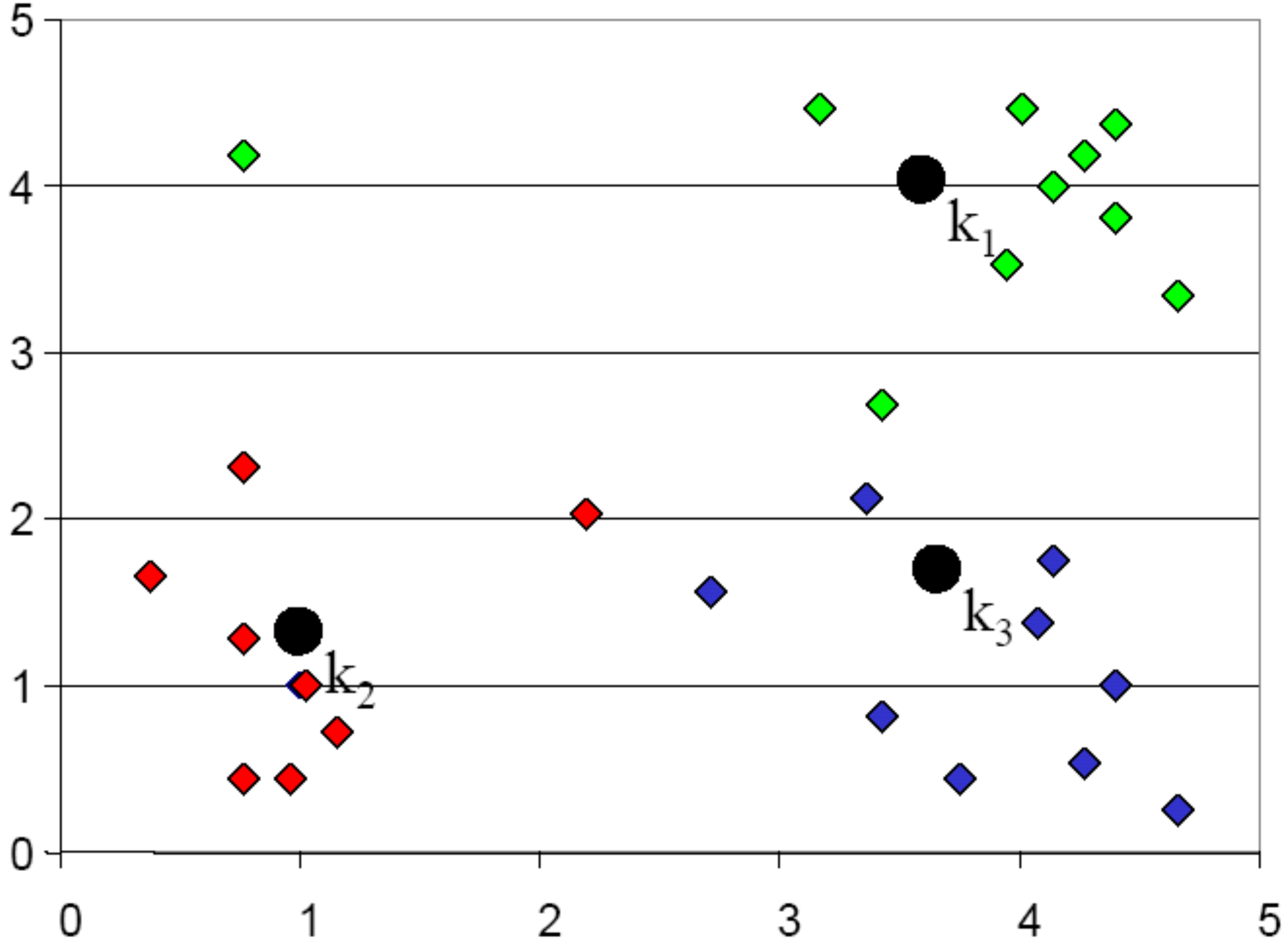
K-means Clustering: Step 1



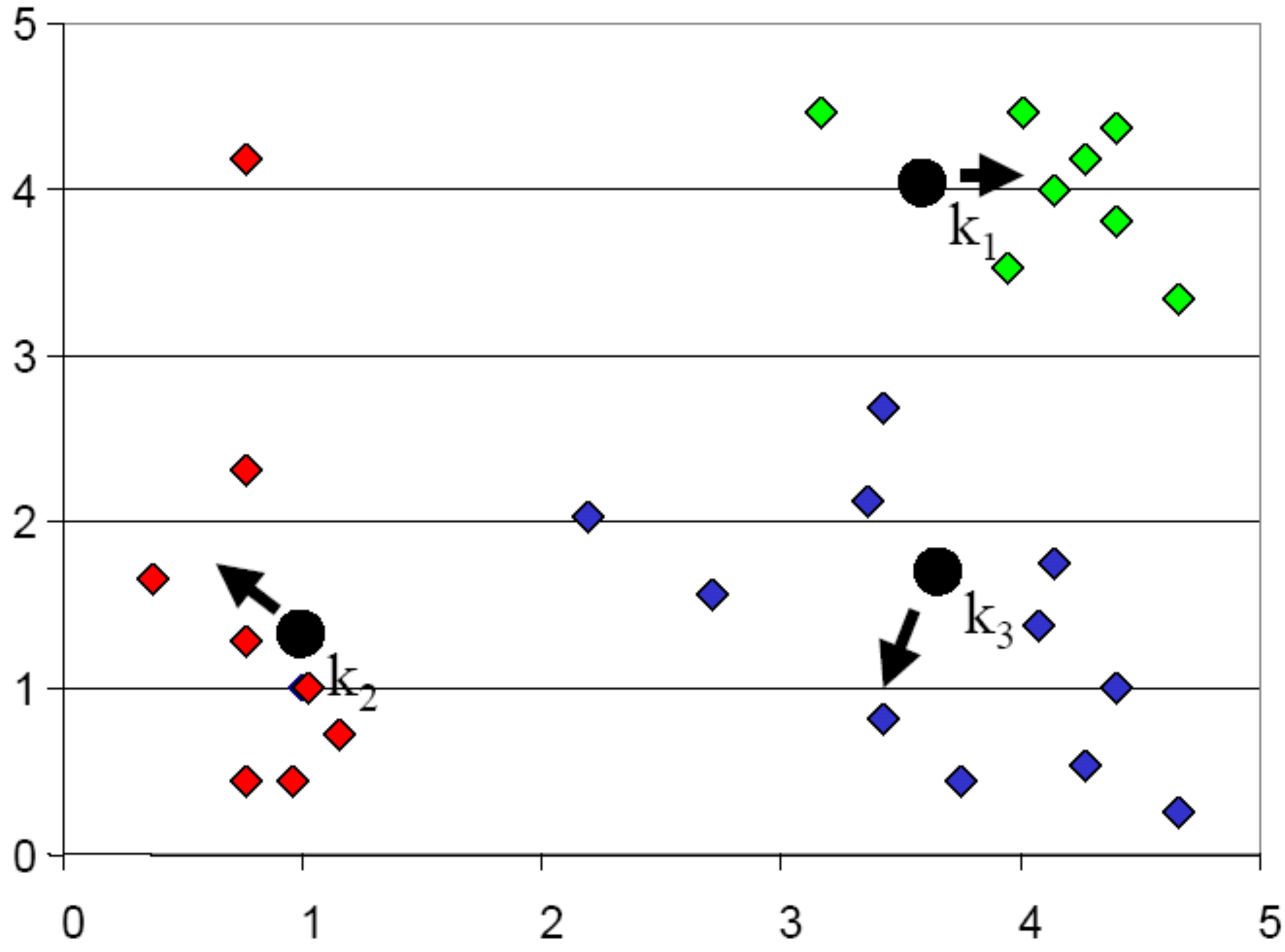
K-means Clustering: Step 2



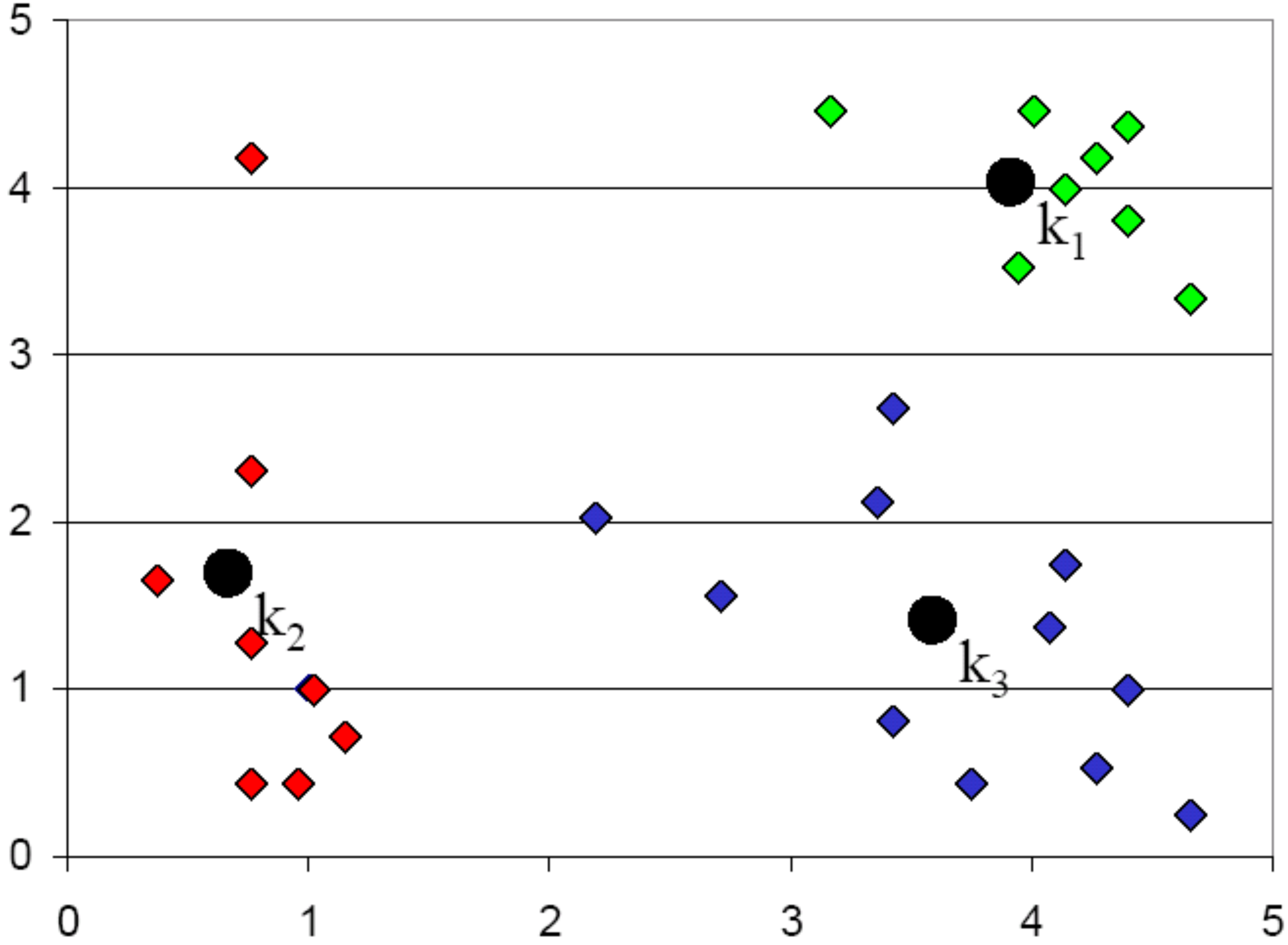
K-means Clustering: Step 3



K-means Clustering: Step 4



K-means Clustering: Step 5



K-Means

Algorithm

1. Decide on a value for *k*.
2. Initialize the *k* cluster centers randomly if necessary.
3. Decide the class memberships of the *N* objects by assigning them to the nearest cluster centroids (aka the center of gravity or mean)

$$\vec{\mu}_k = \frac{1}{\mathcal{C}_k} \sum_{i \in \mathcal{C}_k} \vec{x}_i$$

4. Re-estimate the *k* cluster centers, by assuming the memberships found above are correct.
5. If none of the *N* objects changed membership in the last iteration, exit. Otherwise go to 3.

Seeded k-means

Algorithm

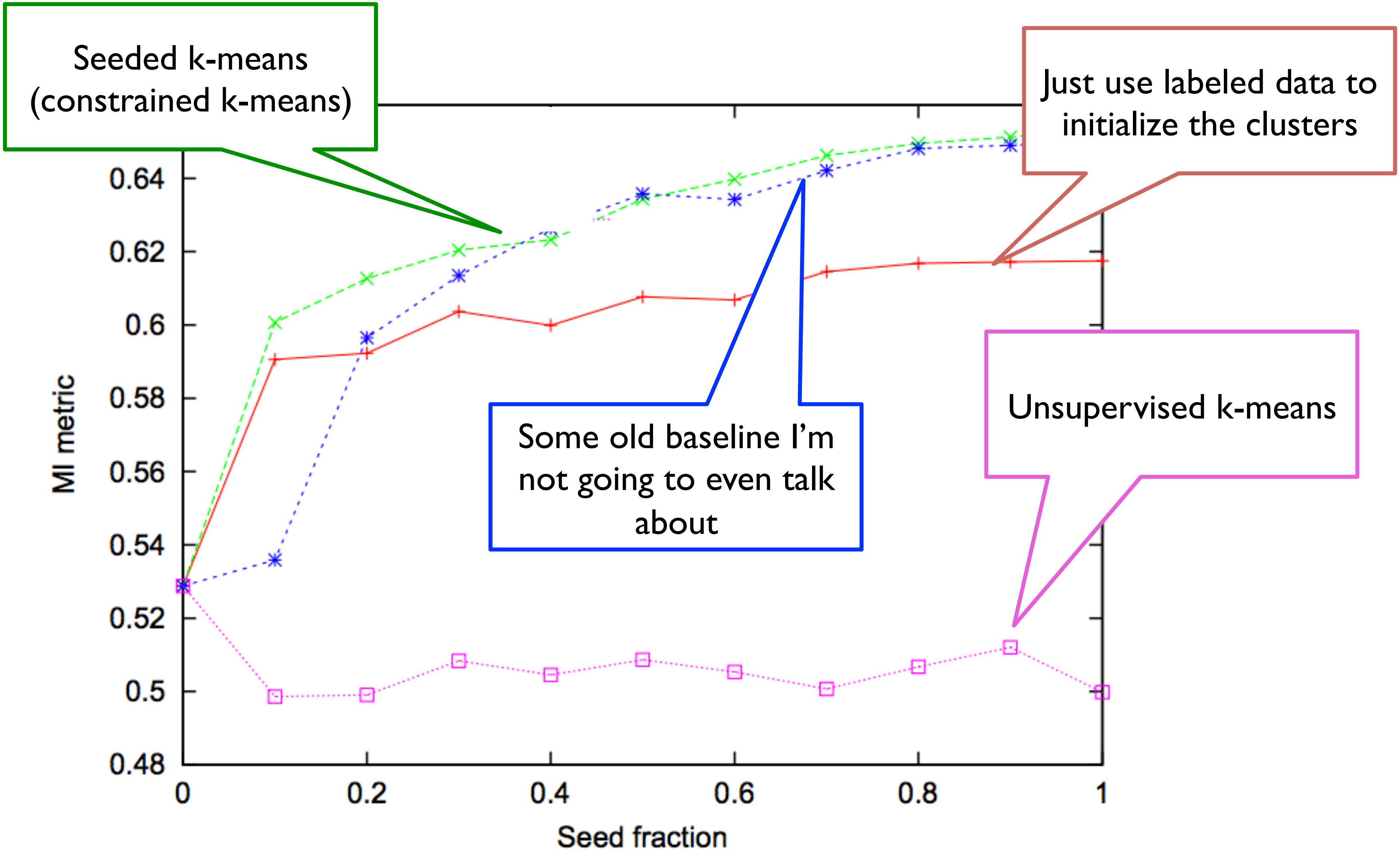
1. Decide on a value for k. *k is* the number of classes
2. Initialize the k cluster centers using the labeled "seed" data
3. Decide the class memberships of the N objects by assigning them to the nearest cluster centroids (aka the center of gravity or mean)

$$\vec{\mu}_k = \frac{1}{\mathcal{C}_k} \sum_{i \in \mathcal{C}_k} \vec{x}_i$$

except keep the seeds in the class they are *known* to belong to

4. Re-estimate the k cluster centers, by assuming the memberships found above are correct.
5. If none of the N objects changed membership in the last iteration, exit. Otherwise go to 3.

Basu and Mooney ICML 2002

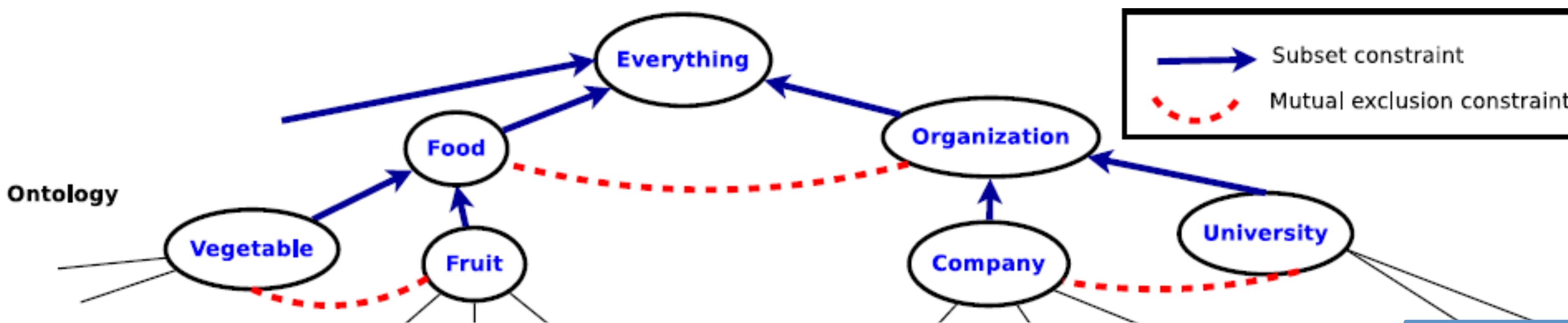


20 Newsgroups dataset

Outline

- The general idea and an example (NELL)
- Some types of SSL
  - Margin-based: transductive SVM
    - Logistic regression with entropic regularization
  - Generative: seeded k-means
    - Some recent extensions….
  - Nearest-neighbor like: graph-based SSL

Seeded k-means for a hierarchical classification tasks



Bit vector with one bit for each category

Simple extension:

1. Don't assign to one of K classes: instead make a decision about *every* class in the ontology
   - ~~example → {1,..K}~~ example → 00010001
2. Pick "closest" bit vector consistent with constraints
   - this is an (ontology-sized) optimization problem that you solve independently for each example

Seeded k-means

Algorithm

1. Decide on a value for k. *k is* the number of classes
2. Initialize the k cluster centers using the labeled "seed" data
3. Decide the class memberships of the N objects by assigning them to the best consistent set of categories from the ontology

$$\vec{\mu}_k = \frac{1}{\mathcal{C}_k} \sum_{i \in \mathcal{C}_k} \vec{x}_i$$

except keep the seeds in the class<u>es</u> they are *known* to belong to

4. Re-estimate the k cluster centers, by assuming the memberships found above are correct.
5. If none of the N objects changed membership in the last iteration, exit. Otherwise go to 3.

Automatic Gloss Finding for a Knowledge Base

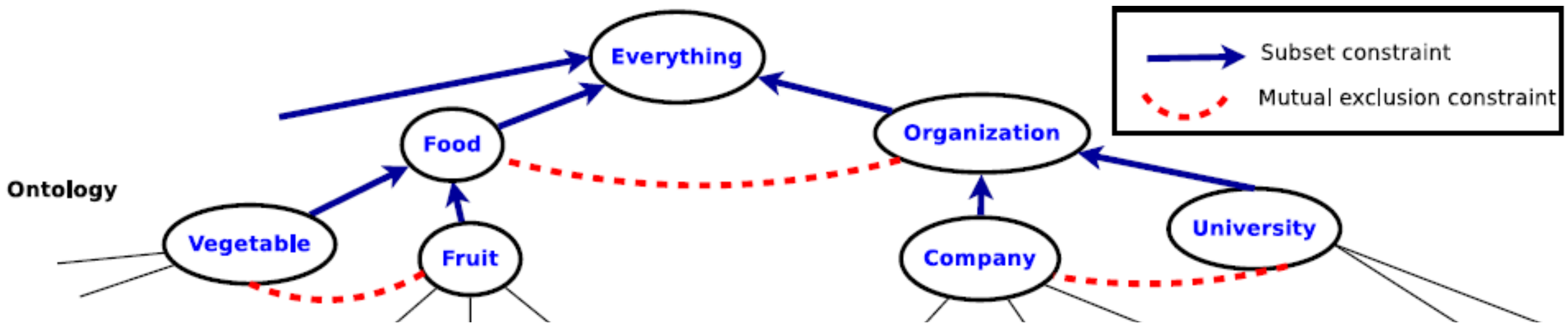
- **Glosses:** Natural language definitions of named entities.

  *E.g. "**Microsoft**" is an American multinational corporation headquartered in Redmond that develops, manufactures, licenses, supports and sells computer software, consumer electronics and personal computers and services ...*

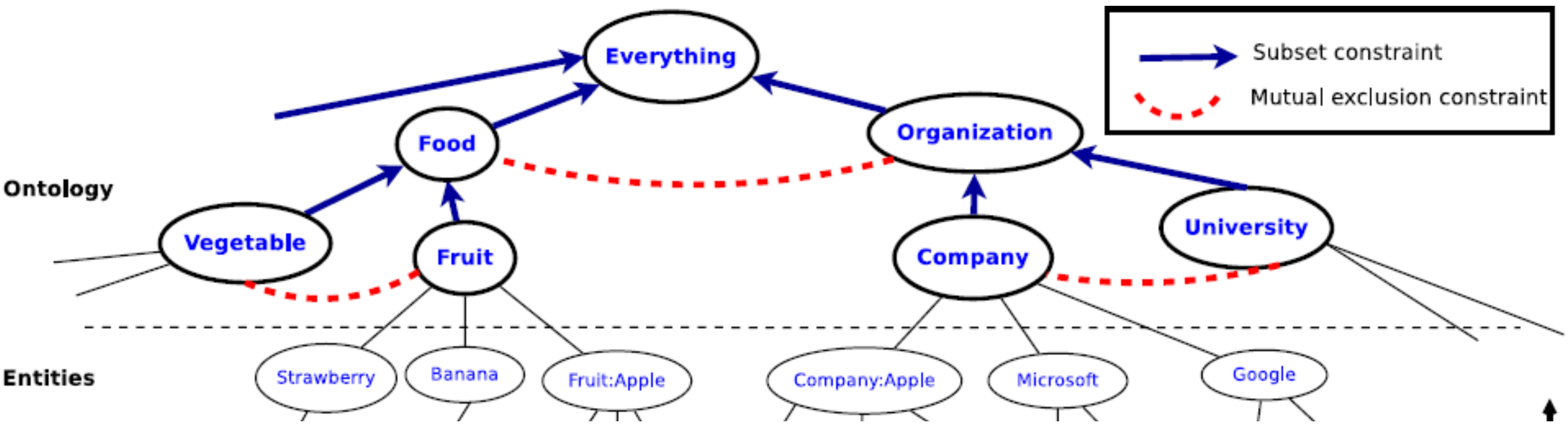
  – **Input:** Knowledge Base i.e. a set of concepts (e.g. company) and entities belonging to those concepts (e.g. Microsoft), and a set of potential glosses.
  – **Output:** Candidate glosses matched to relevant entities in the KB. *"Microsoft is an American multinational corporation headquartered in Redmond ..."* is mapped to **entity "Microsoft" of type "Company"**.

*[**Automatic Gloss Finding for a Knowledge Base using Ontological Constraints**, Bhavana Dalvi Mishra, Einat Minkov, Partha Pratim Talukdar, and William W. Cohen, 2014, Under submission]*

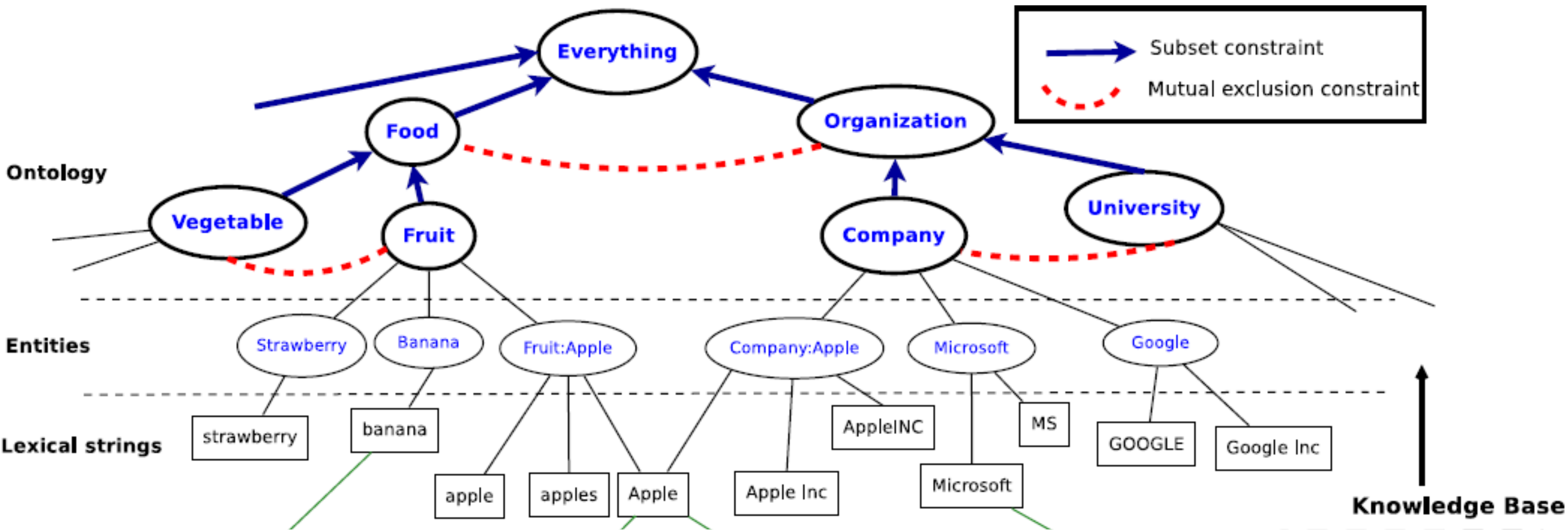
Example: Gloss finding



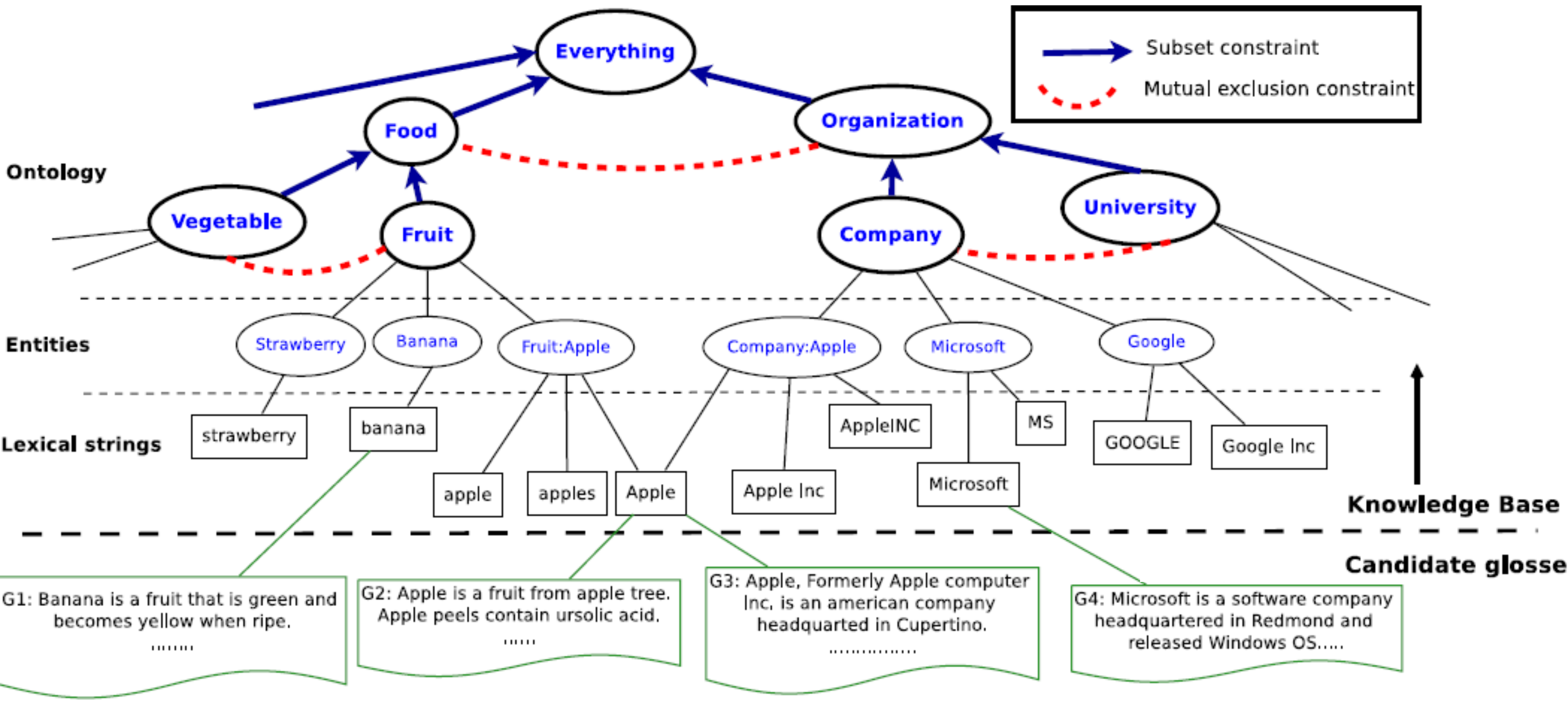
Example: Gloss finding



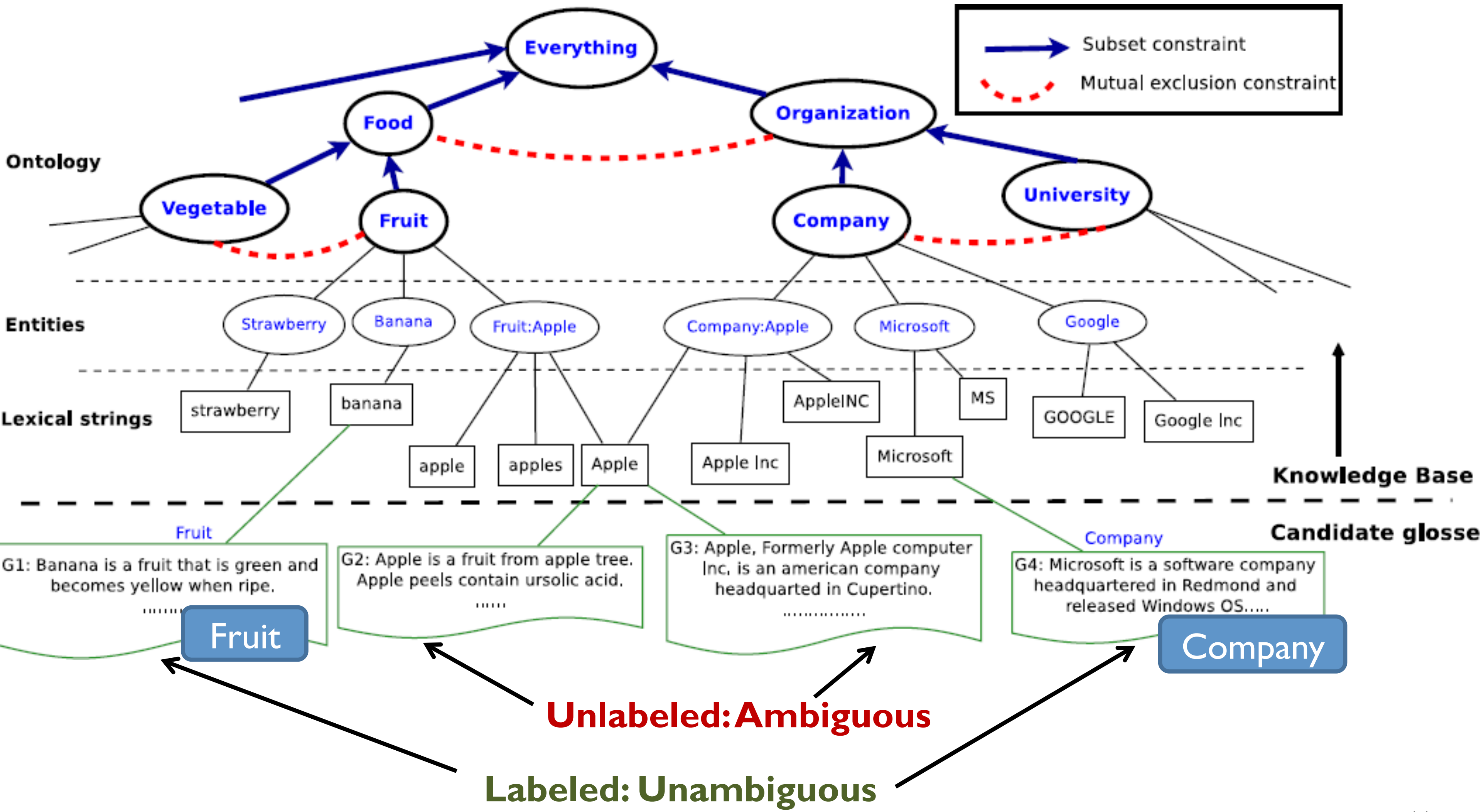
Example: Gloss finding



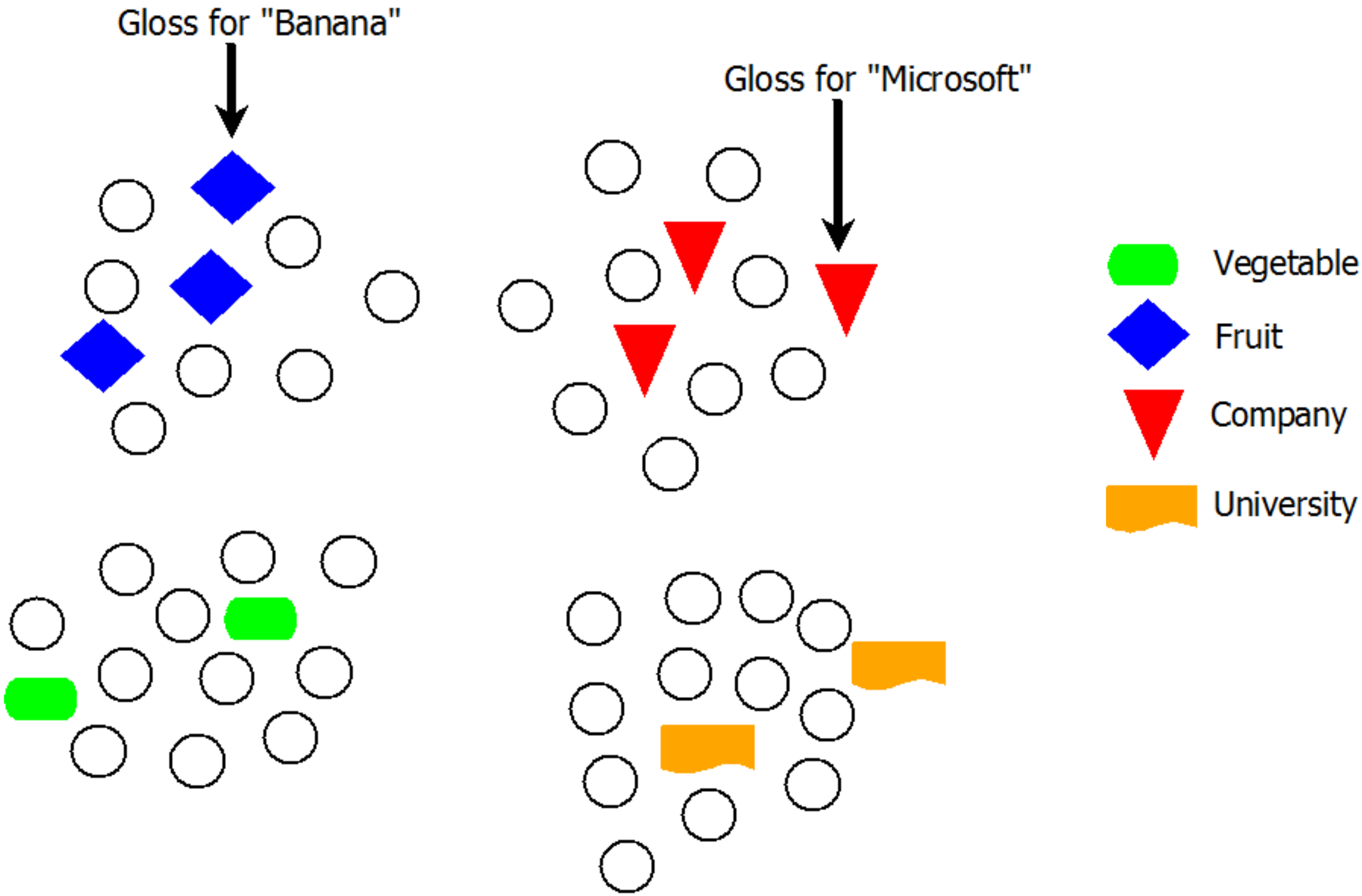
Example: Gloss finding



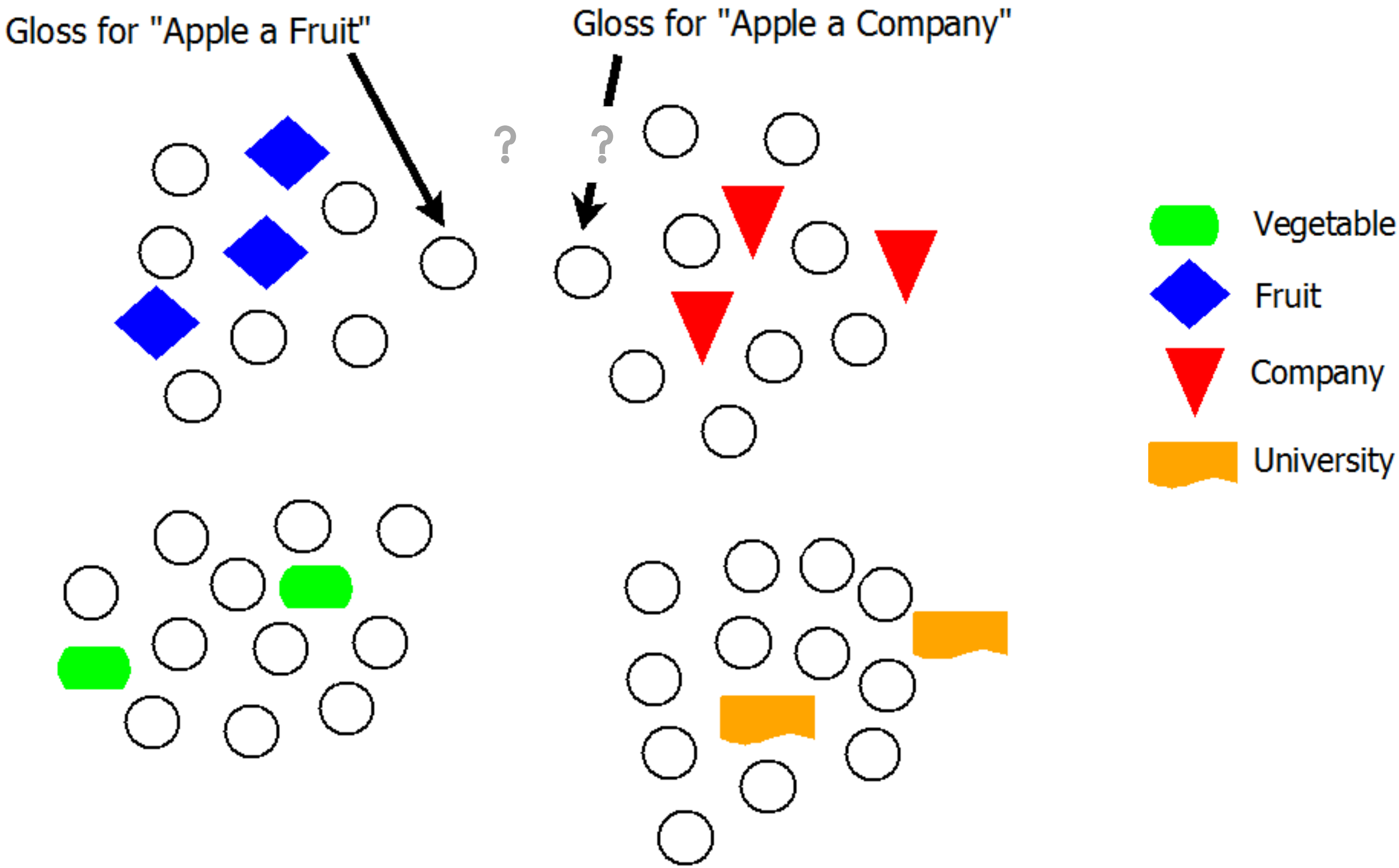
Training a clustering model



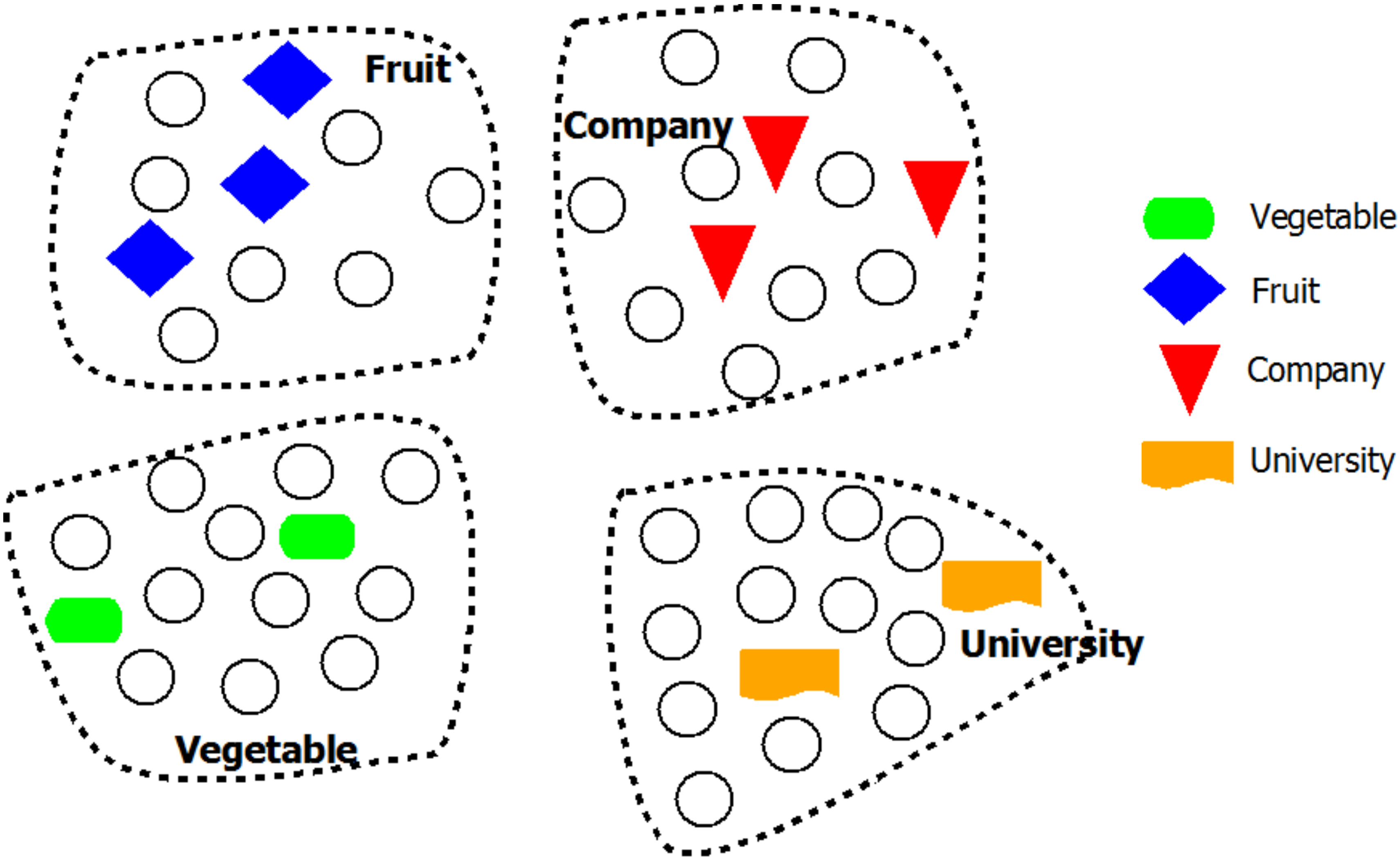
GLOFIN: Clustering glosses



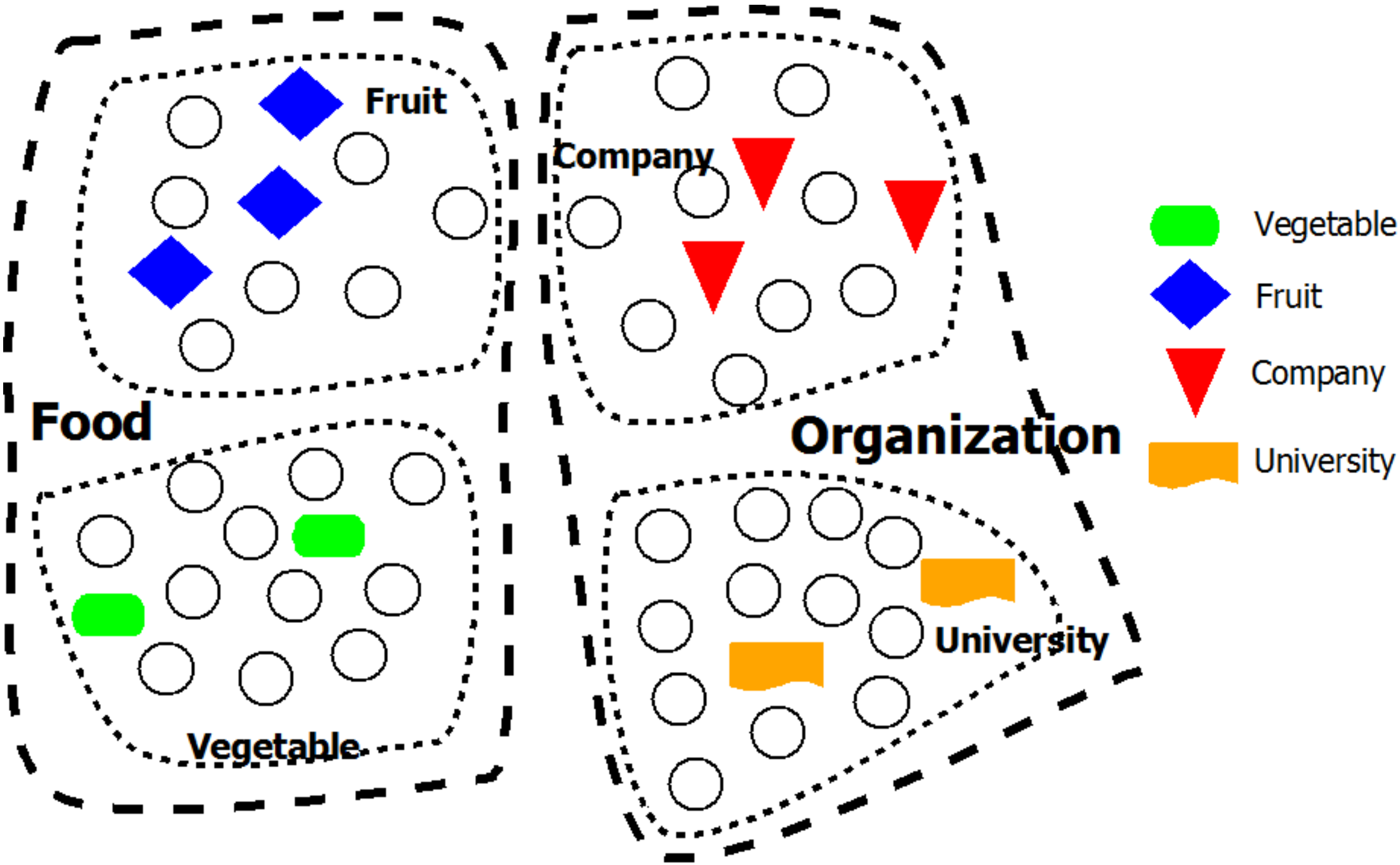
GLOFIN: Clustering glosses



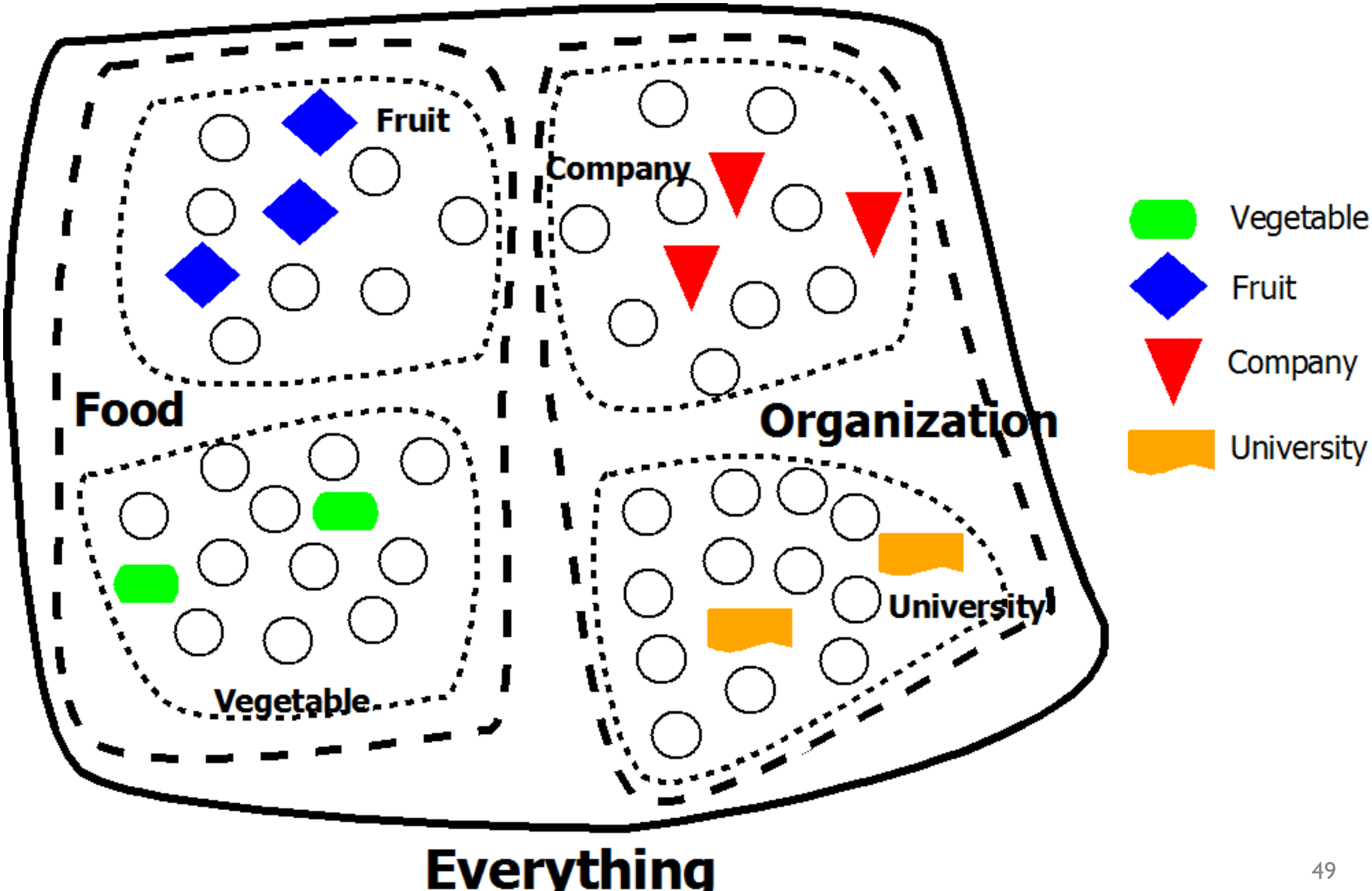
GLOFIN: Clustering glosses



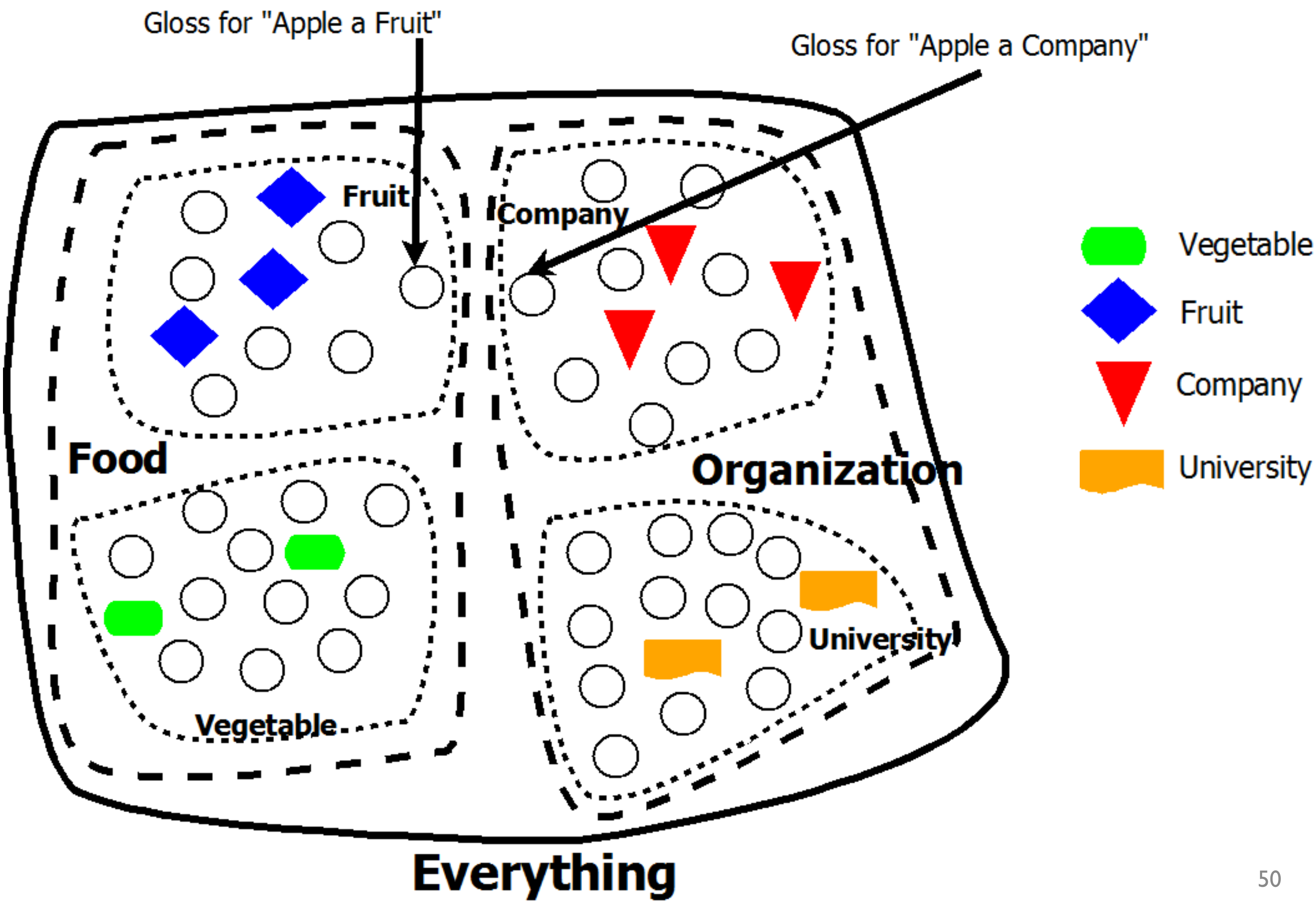
GLOFIN: Clustering glosses



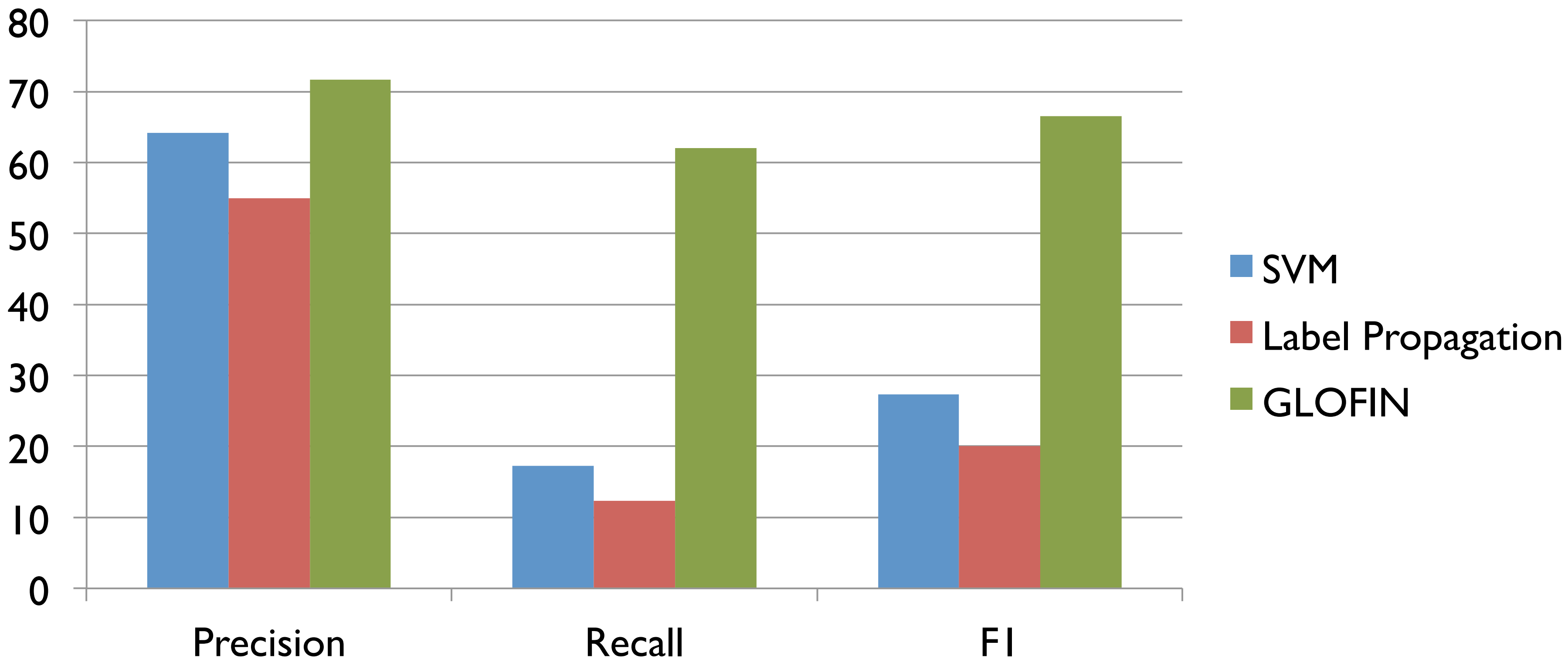
GLOFIN: Clustering glosses



GLOFIN: Clustering glosses



GLOFIN on NELL Dataset



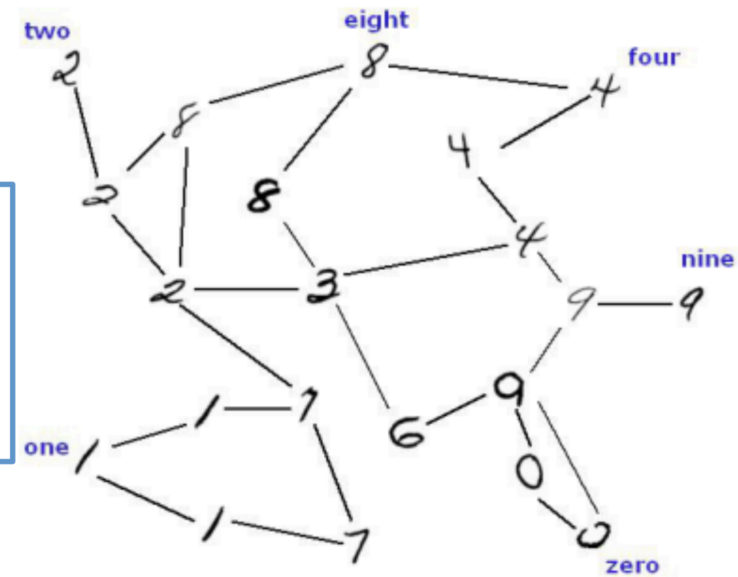
275 categories, 247K candidate glosses, #train=20K, #test=227K

Outline

- The general idea and an example (NELL)
- Some types of SSL
  - Margin-based: transductive SVM
    - Logistic regression with entropic regularization
  - Generative: seeded k-means
  - Nearest-neighbor like: graph-based SSL

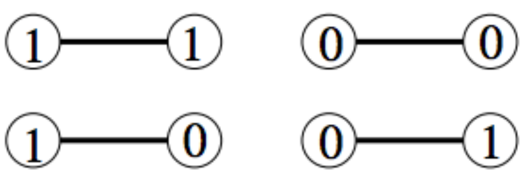
- Idea: construct a graph connecting the most similar examples (k-NN graph)
- Intuition: **nearby points should have similar labels** – labels should "propagate" through the graph
- Formalization: try and minimize "energy" defined as:

Harmonic fields – Gharamani, Lafferty and Zhu



In this example y is a length-10 vector

energy: $E(\mathbf{y}) = \frac{1}{2}\sum_{i,j} w_{ij}\left(y_i - y_j\right)^2$



happy, low energy

unhappy, high energy

Observed label

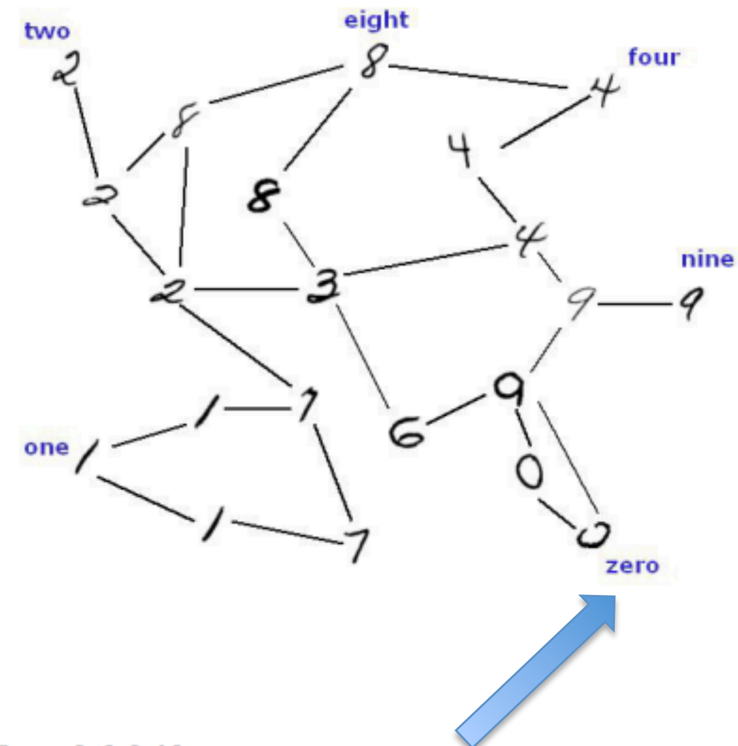
- Result 1: at the minimal energy state, each node's value is a **weighted average of its neighbor's weights**:

$$\Delta \mathbf{f} = 0 \ \text{ or } \ f_i = \frac{\sum_{j \sim i} w_{ij} f_j}{\sum_{j \sim i} w_{ij}}, \ i \in U$$

energy: $E(\mathbf{y}) = \frac{1}{2} \sum_{i,j} w_{ij} \left(y_i - y_j\right)^2$

1 — 1	0 — 0	happy, low energy
1 — 0	0 — 1	unhappy, high energy

Harmonic fields – Gharamani, Lafferty and Zhu



Observed label

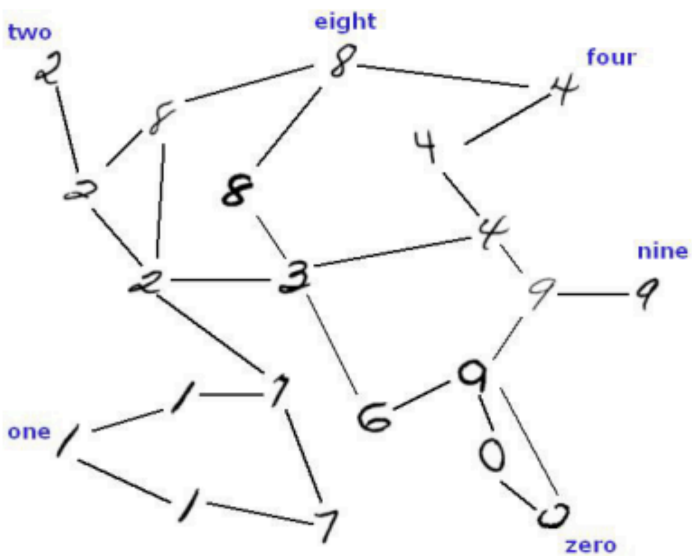
"Harmonic field" LP algorithm

- Result 2: you can reach the minimal energy state with a simple iterative algorithm:
  - Step 1: For each seed example (x_i, y_i):
    - Let $V^0(i,c) = [|\ y_i = c\ |]$
  - Step 2: for t=1,…,T --- *T is about 5*
    - Let $V^{t+1}(i,c)$ =weighted average of $V^{t+1}(j,c)$ for all j that are linked to i, and renormalize

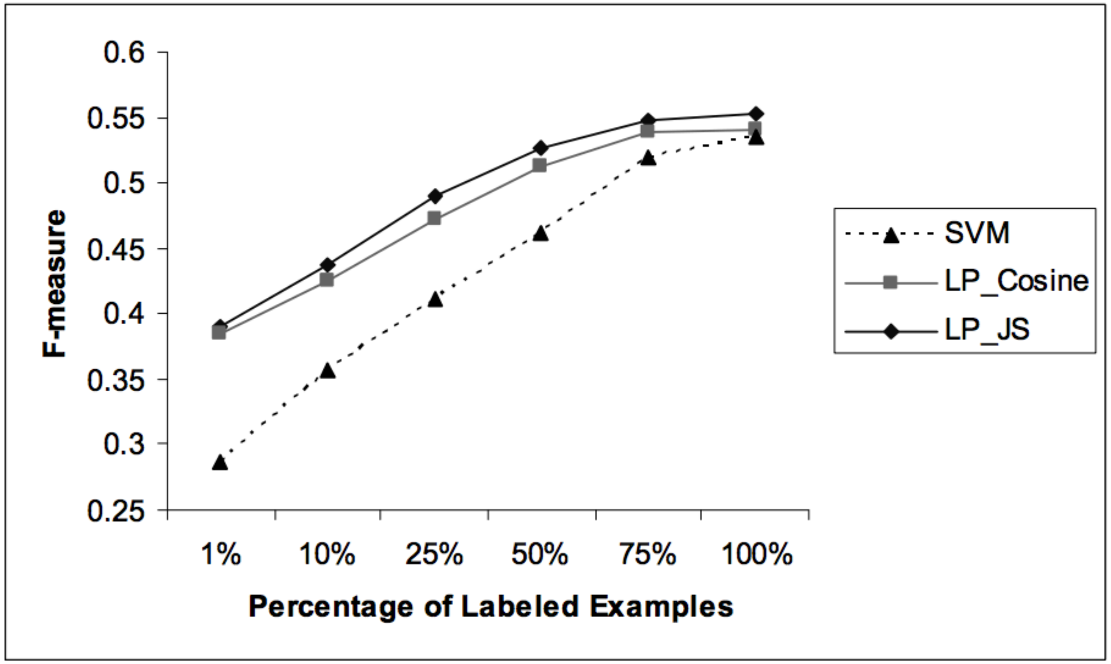
$$V^{t+1}(i,c) = \frac{1}{Z} \sum_j w_{i,j} V^t(j,c)$$

    - For seeds, reset $V^{t+1}(i,c) = [|\ y_i = c\ |]$

Harmonic fields – Gharamani, Lafferty and Zhu



This family of techniques is called "Label propagation"

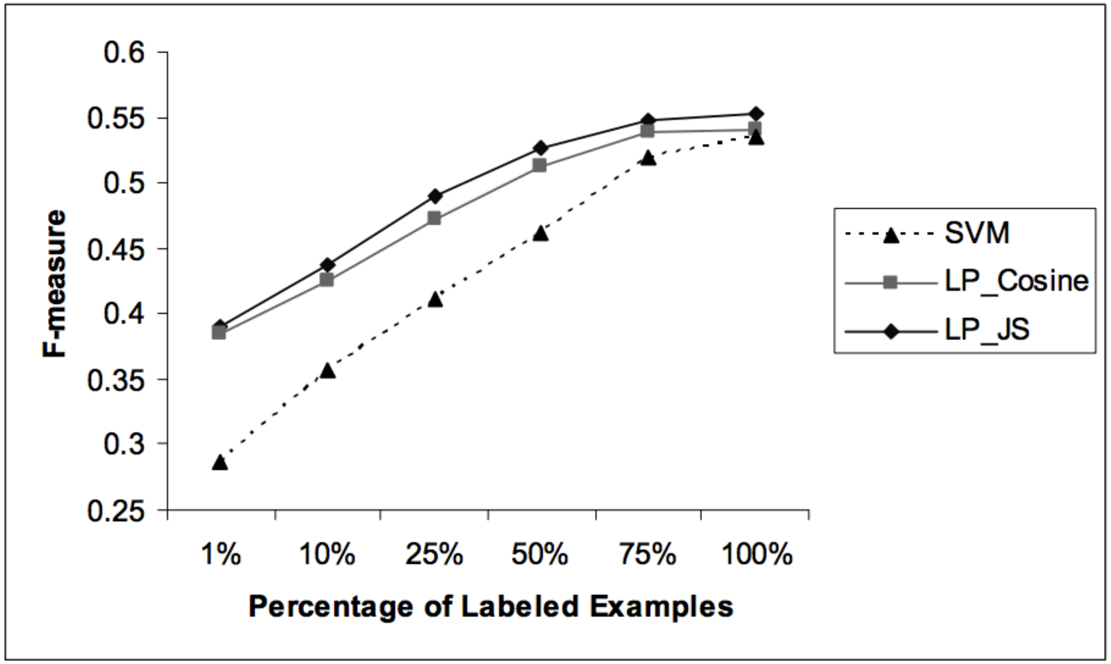


Harmonic fields – Gharamani, Lafferty and Zhu

This family of techniques is called "Label propagation"

This experiment points out some of the issues with LP:

1. What distance metric do you use?
2. What energy function do you minimize?
3. What is the right value for K in your K-NN graph? Is a K-NN graph right?
4. If you have lots of data, how expensive is it to build the graph?



NELL: Uses Co-EM ~= HF

Extract cities:

Examples

Paris
Pittsburgh
Seattle
Cupertino

San Francisco
Austin
denial

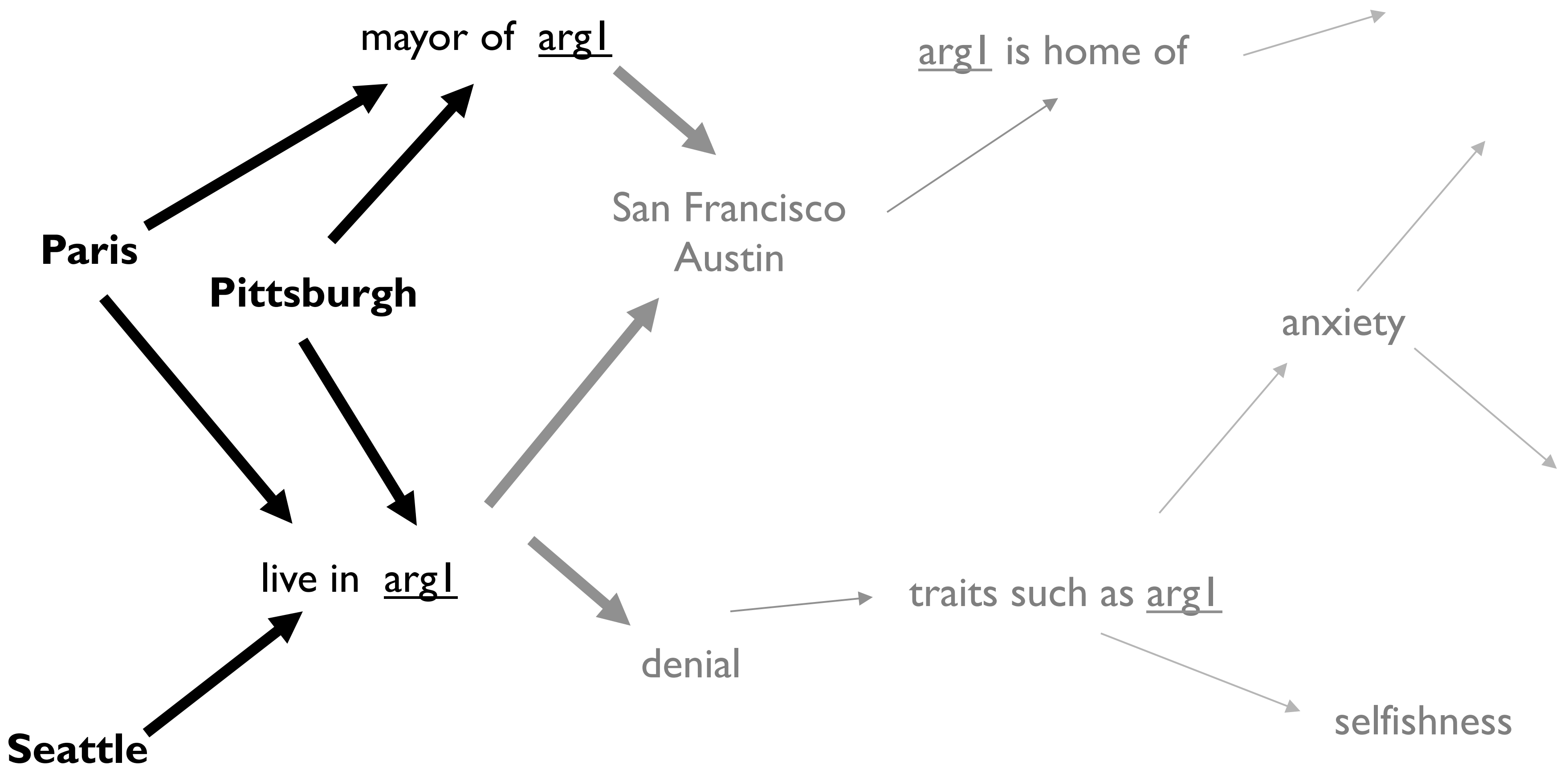
anxiety
selfishness
Berlin

mayor of arg1
live in arg1

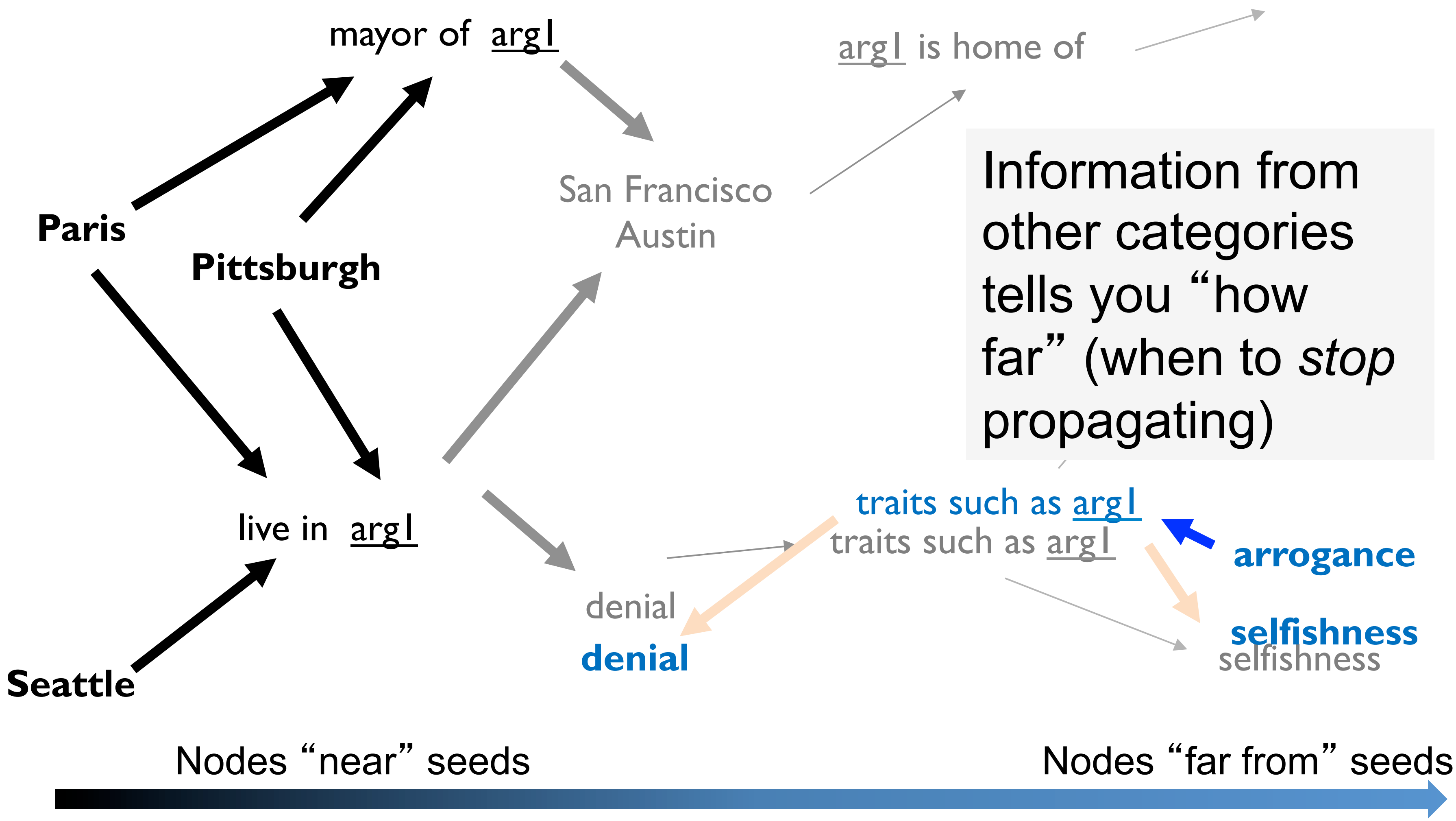
arg1 is home of
traits such as *arg1*

Features

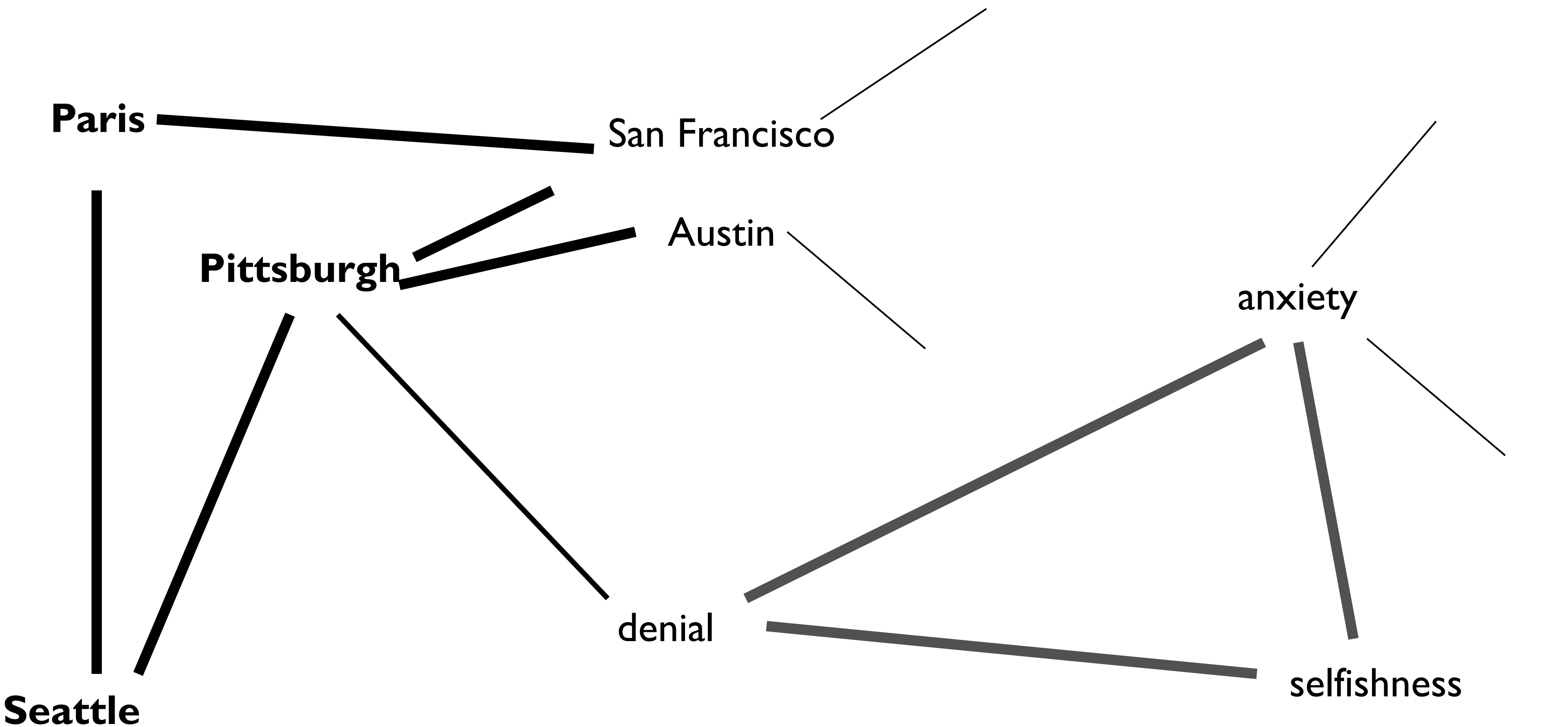
Semi-Supervised Bootstrapped Learning via Label Propagation



Semi-Supervised Bootstrapped Learning via Label Propagation



Difference: graph construction is not instance-to-instance but instance-to-feature



Some other general issues with SSL

- How much unlabeled data do you want?
  - Suppose you're optimizing $J = J_L(L) + J_U(U)$
  - If |U| >> |L| does J_U dominate J?
    - If so you're basically just clustering
  - Often we need to balance J_L and J_U
- Besides L, what other information about the task is useful (or necessary)?
  - Common choice: relative frequency of classes
  - Various ways of incorporating this into the optimization problem

Key and not-so-key points

- The general idea : what is SSL and when do you want to use it?
  - NELL as an example of SSL
- Different SSL methods:
  - margin-based approach: start with a supervised learner
    - transductive SVM: what's optimized and why
    - logistic reg with entropic regularization
  - k-means versus seeded k-means: start with clustering
    - The core algorithm and what
    - Extension to hierarchical case and GLOFIN
  - nearest-neighbor like: graph-based SSL and LP
    - The HF algorithm and the energy function being minimized