Unsupervised Learning: Clustering

Micha Elsner

March 6, 2014

Supervised learning

So far, we've looked at prediction tasks based on annotated training data:

- Given *hand-tagged text*, predict *tag sequence for new text*
- Given *treebank parsed by linguists*, predict *parse trees for new sentences*
- Given *sentences and human compressions*, figure out *how to compress the way humans do it*
- Given *English sentences*, predict *next word in new English sentence*
  - A bit different since there's no "annotation"

Supervised learning

Training data has examples of the (usually hidden) structure you are learning to predict

Unsupervised learning

- Given *strings of words*, learn *a set of POS tags and how to apply them*
- Given *sentences*, learn *a grammar and how to parse with it*
- Given *sentences*, figure out *how to compress and remain grammatical*

Unsupervised learning

Propose hidden structures for data without any labeled examples

What do you get out?

Unsupervised learning proposes some structures or labels

- These don't have intrinsic semantics

Supervised

DT	NN	VBD	IN	DT	NN
the	cat	sat	on	the	mat

Unsupervised

C0	C1	C2	C3	C0	C1
the	cat	sat	on	the	mat

The *CX* tags don't mean anything on their own

- They gain meaning through context
  - "cat" and "mat" are both *C1*, so they are *the same*
- As linguists, we can interpret C1 as "the class of nouns"
- But not all labels we learn map easily to linguistic concepts

Why would anyone do this?

Several reasons to do unsupervised learning:

- Data exploration: what is the structure of a large dataset?
- Downstream application: learn features to use for another task
- Classification but classes keep changing
- Cognitive models of learning

We'll discuss these in turn

Exploratory analysis

You have a lot of data... and you want to know what's going on in it (in some generic way)

- Can talk about *what instances are similar*
- Or *what features co-occur*
- "It seems like there are three groups of authors. C1 authors like to use words like *oil* and *wheat*, C2 authors use words like *stocks* and *bonds*, C3 authors use words like *kitten* and *ninja*"
  - Can restate as "authors who talk about *oil* also talk about *wheat*"
- "There are a few major groups of Latin nouns. C1 nouns take *a,ae,am,ā* as endings and C2 nouns take *us,i,ō,um* as endings"
  - Can restate as "nouns that end in *ae* also end in *am*"

No direct experimental evaluation; part of a qualitative/quantitative analysis

Downstream application

There's a task you really care about

- Information extraction, coreference, dialogue structure

You wish you had a

- tagger, parser, dictionary, etc

Build one using unsupervised learning!

- Can't have linguistically aware features like "number of nouns"
- But can have "number of C1", "number of C2"...
- Maybe one of these will be *like* number of nouns
- Which will make your information extraction system better

Experimental evaluation in terms of information extraction performance

Entailment

Rimell's entailment system: does "any lingering suspicion that this was a trick was dispelled" entail "suspicion was dispelled"?
Yes, because both dependency trees have a path:

- suspicion (nsubj, $\leftarrow$) was (aux, $\leftarrow$) dispelled

This still works if the trees are labeled arbitrarily:

- suspicion (C1, $\leftarrow$) was (C2, $\leftarrow$) dispelled

Or the arrows are reversed:

- suspicion (C1, $\rightarrow$) was (C2, $\rightarrow$) dispelled

Same path still in both trees

Classification, but classes change

There is a ground truth labeling you want to recover

- You'd use classification
- But the set of classes for each instance is different

For instance coreference:

- Linking "Clinton", "Hillary Clinton", "she"
- The set of known entities keeps changing (new people)
- In a new document, might have to link "Justice Ginsburg", "Ruth Bader Ginsburg", "herself"
  - Can't just learn Clinton-specific features
- Labels *C1*, *C2*... are individual entities
  - So we can label C1: "Clinton", C1: "Hillary Clinton", C2: "Ginsburg"
- Important thing is *which are the same*

Evaluation: how well do the C1, C2... labels match the true entity labels?

- Measuring this is a research question

Cognitive models of learning

There is a ground truth labeling you want to recover

- But how do people learn it?
- They don't see labeled examples...
  - But they still get the right grammar

Part of speech tagging:

- Can you learn a good set of POS tags from just hearing sentences?
- If you need UG constraints, what are they?
- Build a system that learns POS tags (with whatever constraints)
- Compare these tags to human-labeled tags

Evaluation: how well do the C1, C2... labels match the true POS labels?

- Same eval as for changing labels

Not a great motivation

"If we can learn POS tags without labeled data, we can run it on other languages where there isn't a treebank!"

- Unsupervised learning isn't generally a good tool for cross-linguistic robustness
- Performance on your favorite language doesn't predict performance on new languages
- Data isn't as expensive as you think
  - Just pay some people to annotate a little data
- Semi-supervised learning (a bit of labeled data, a lot of unlabeled data) is better
  - Can be a tool for fieldwork/multilingual engineering

Methods

Now we know what unsupervised learning is

- And why we might want to do it

But how do we do it?
There are lots of methods:

- We will focus on the Expectation/Maximization algorithm (EM)
- Popular, flexible, easy to code

Clustering

Like classification:

- Data in the form of feature vectors $F = [f_1, f_2 \ldots f_m]$
- Each datapoint has a tag T
- Task: predict T from F

In classification, tag is one of (eg) *NN, VB, JJ*...

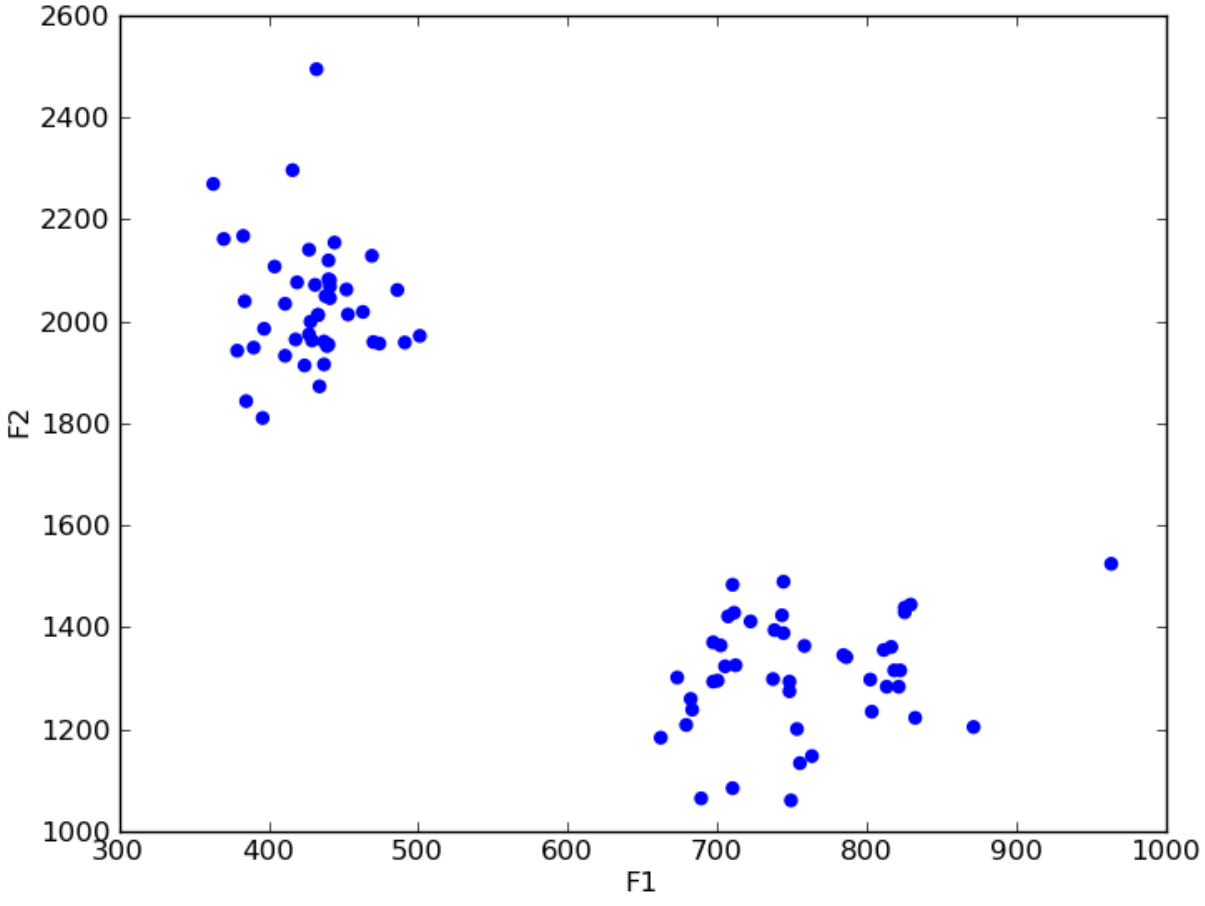
- In clustering, tag is one of $C_1, C_2 \ldots C_k$

What would a good clustering be like?

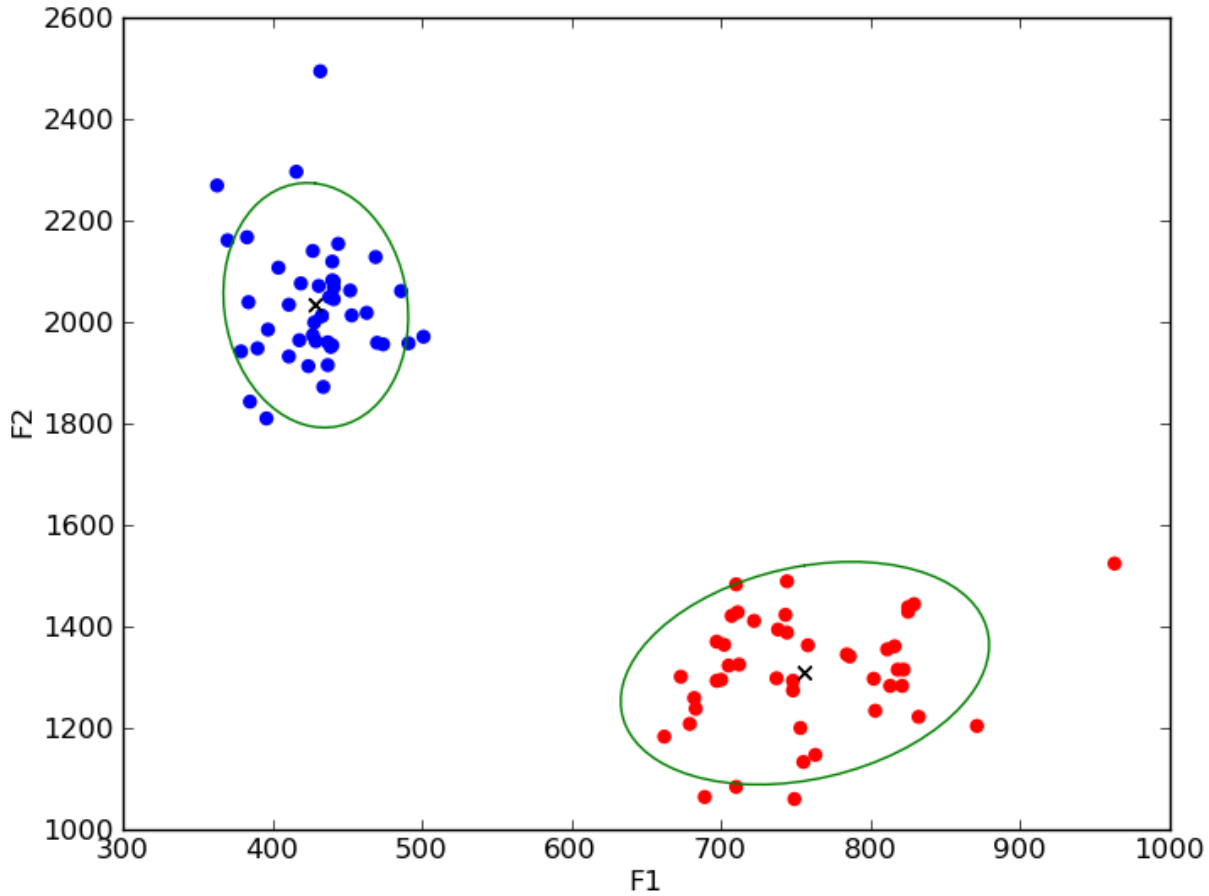
Good clustering

A good clustering groups points that are similar

- For instance, what do you think about this data?
- F1 and F2 for vowel sounds (Getty, Hillenbrand et al)



Clustering for vowels



Mathematically

Choose centers of clusters C_1, C_2: $c_{1,x}, c_{1,y}$ and $c_{2,x}, c_{2,y}$

- So that each datapoint is close to one center

Distance from a datapoint $d_i = (d_x, d_y)$ to a cluster center C is written:

$$||d_i - C||_2 = \sqrt{(d_x - c_x)^2 + (d_y - c_y)^2}$$

Distance to the closest center is:

$$\min_{C^* \in [C_1, C_2]} ||d_i - C^*||_2$$

We want to find cluster centers which minimize this distance:

$$\min_{C_1 = (c_{1,x}, c_{1,y}), C_2 = (c_{2,x}, c_{2,y})} \sum_{d_i \in D} \min_{C^* \in [C_1, C_2]} ||d_i - C^*||_2$$

How do we deal with this problem?

Finding the overall solution to the problem isn't easy

- But it has two easy subparts

If we know C_1, C_2, we can find the closest center to a point:

$$\min_{C^* \in [C_1, C_2]} ||d_i - C^*||_2$$

This is a "classification" step

- Like finding the best tag for a word under Naive Bayes—just try both options, pick the lowest

The other easy part

If we had labeled data (each point is labeled $T = C_1$ or $T = C_2$)

- Finding c_x, c_y is easy

$$\min_{C_1=(c_{1,x},c_{1,y}))} \sum_{d_i : T_i = C_1} ||d_i - C_1||_2$$

Just compute mean values:

- Like count-and-divide in Naive Bayes training

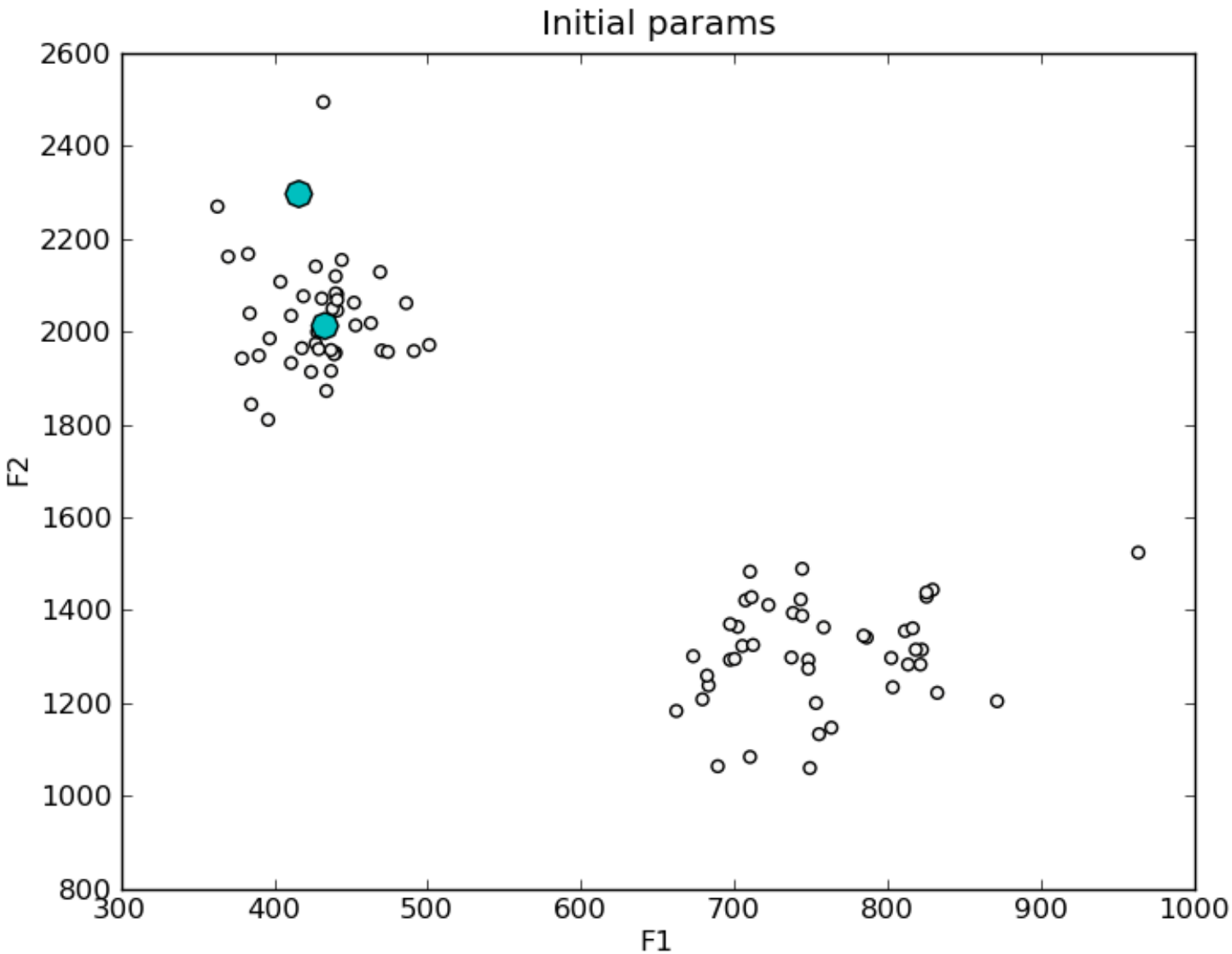
$$c_{1,x} = \frac{1}{\#(T_i = C_1)} \sum_{d_i} d_x$$

$$c_{1,y} = \frac{1}{\#(T_i = C_1)} \sum_{d_i} d_y$$

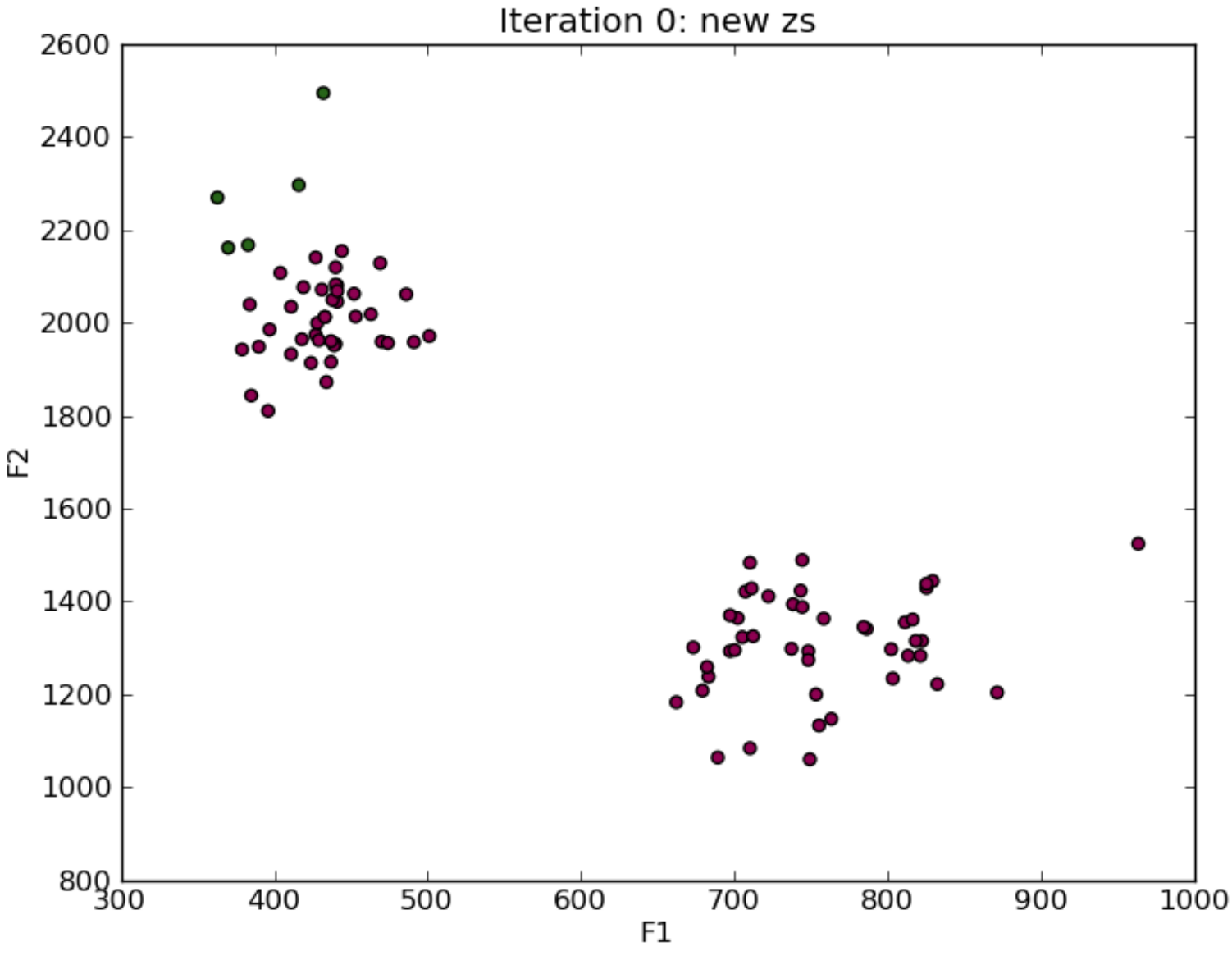
The k-means algorithm

- Randomly choose two points as cluster centers
- Until bored (or assignments don't change)
  - Assign all points to closest cluster
  - Move cluster center to mean of points

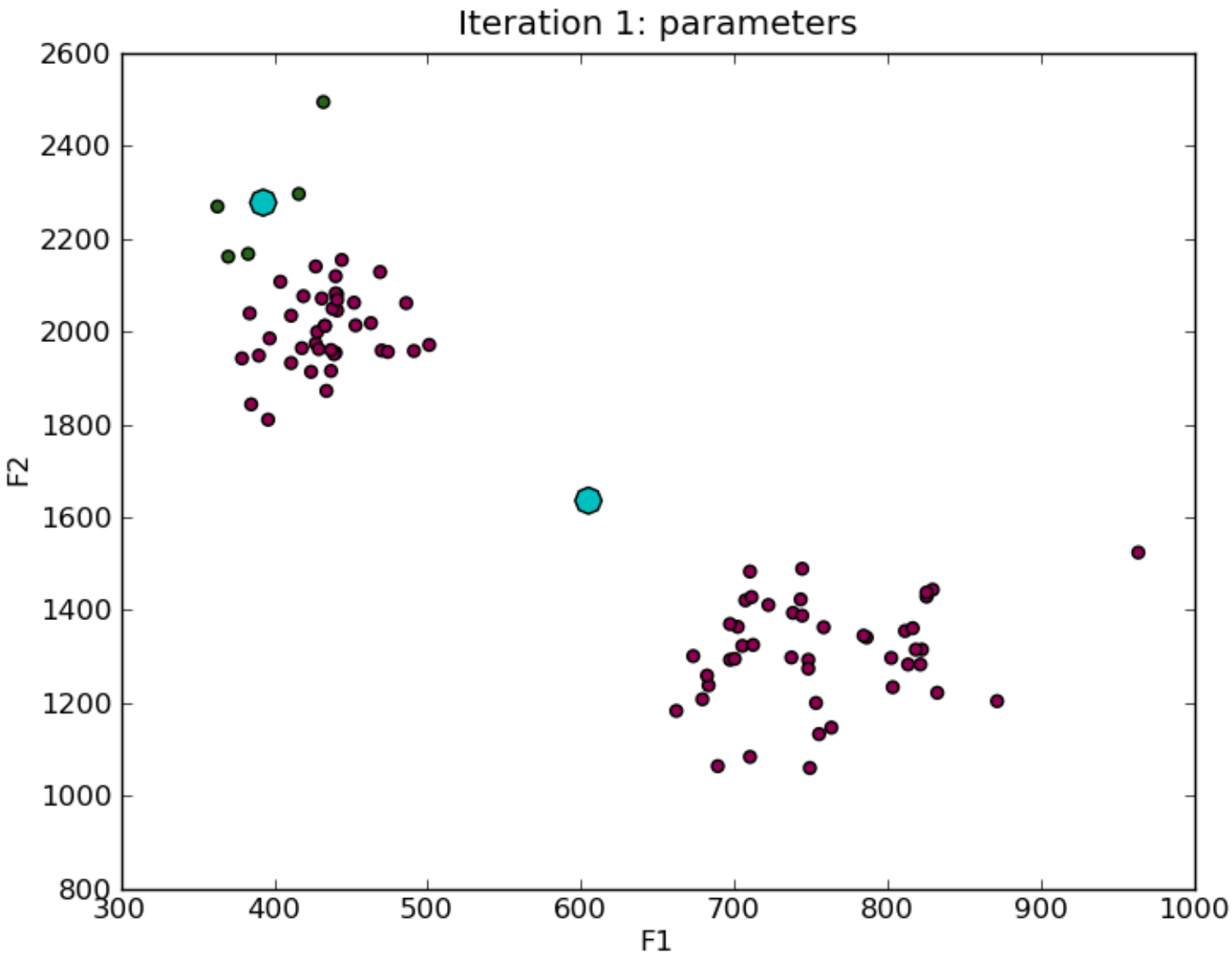
Starting state



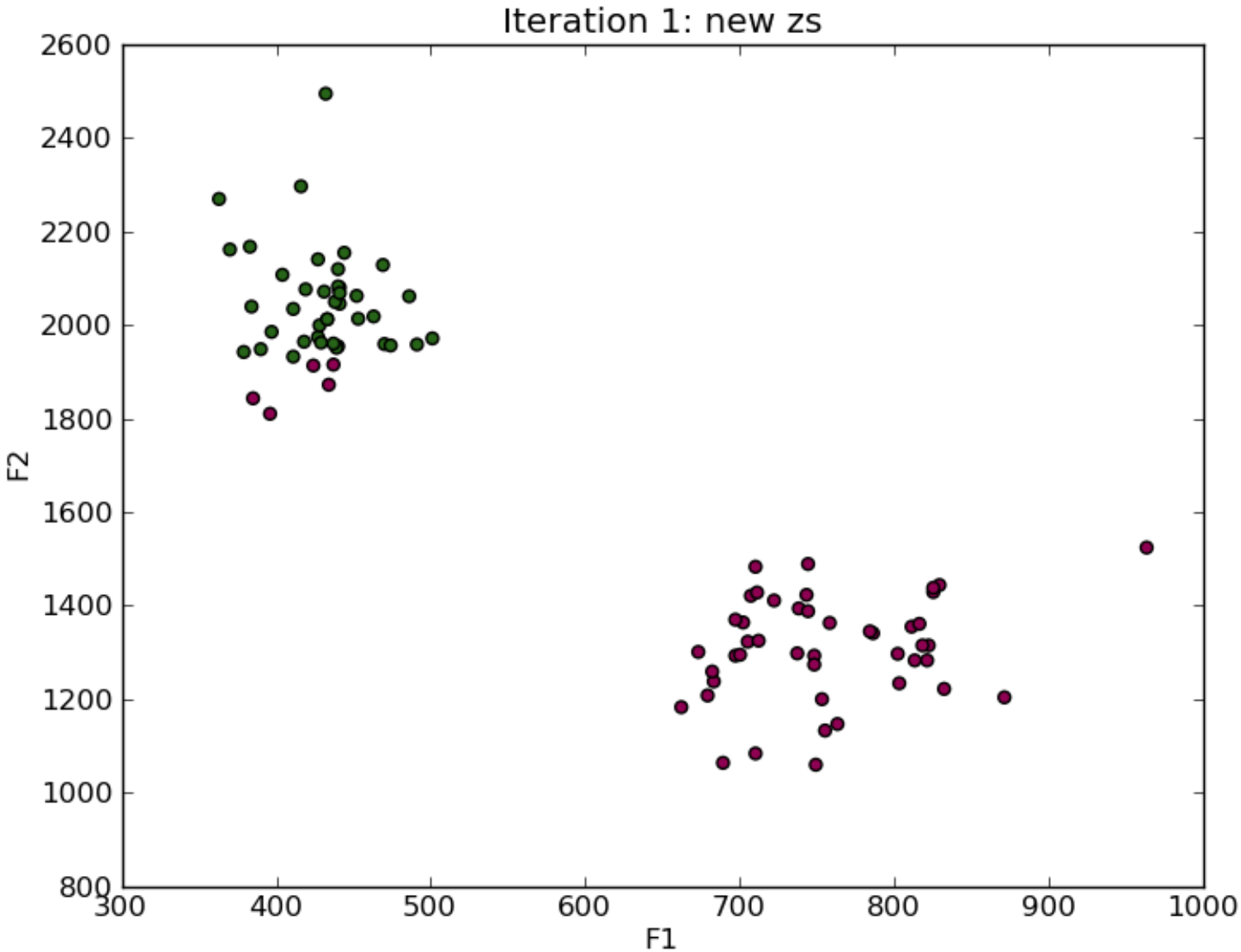
Classify points (assign to closest center)



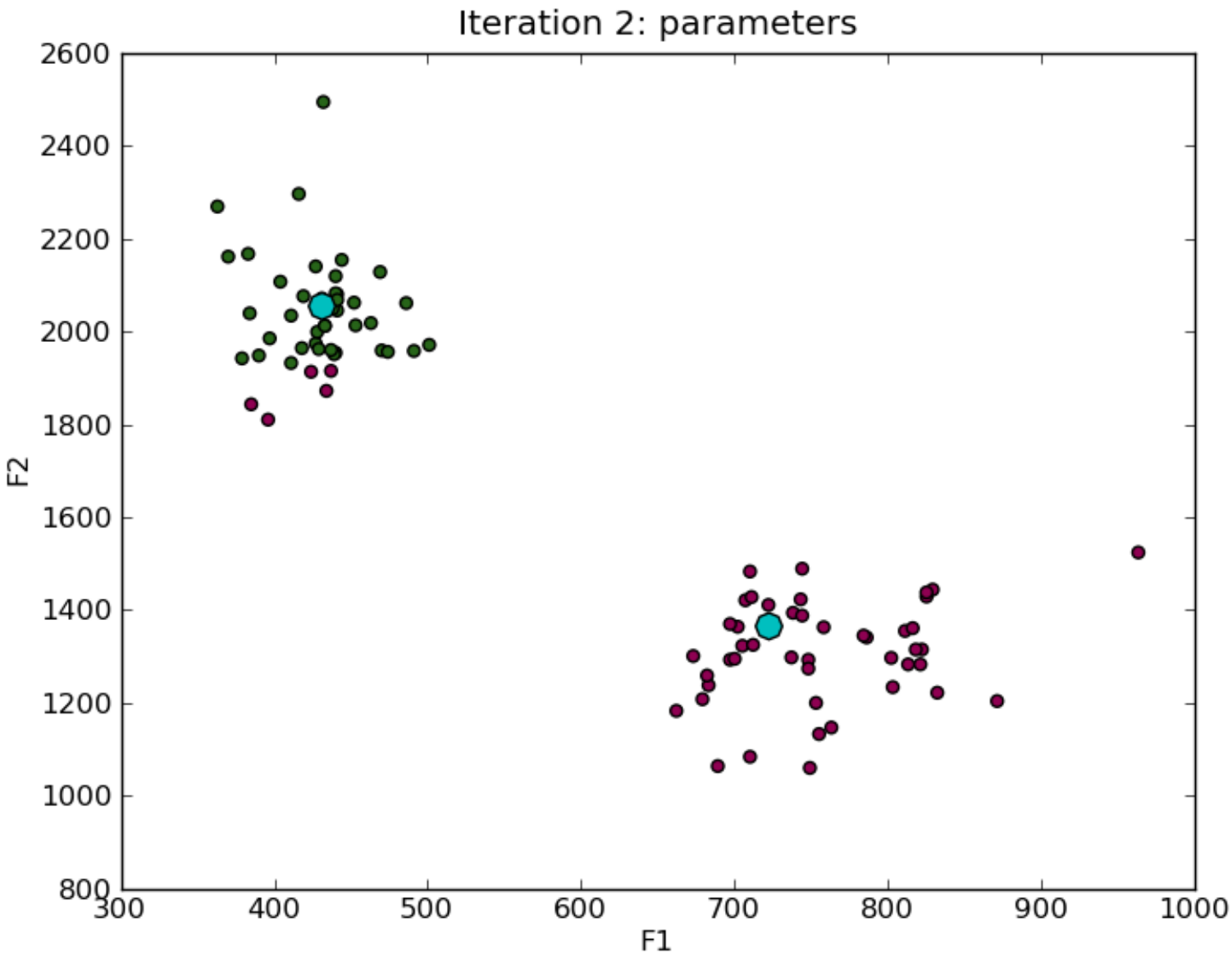
Estimate centers (mean of points)



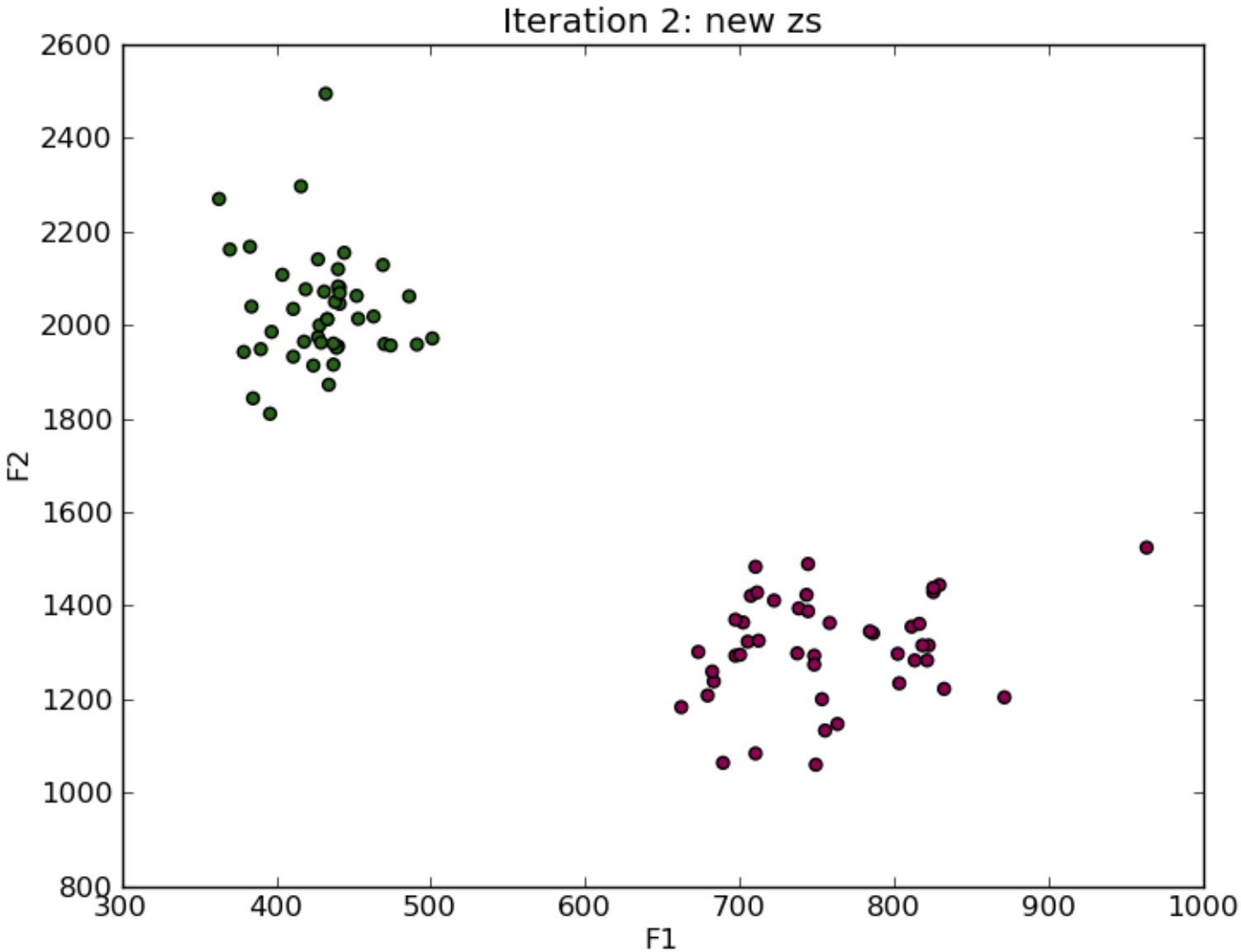
And again (classify)



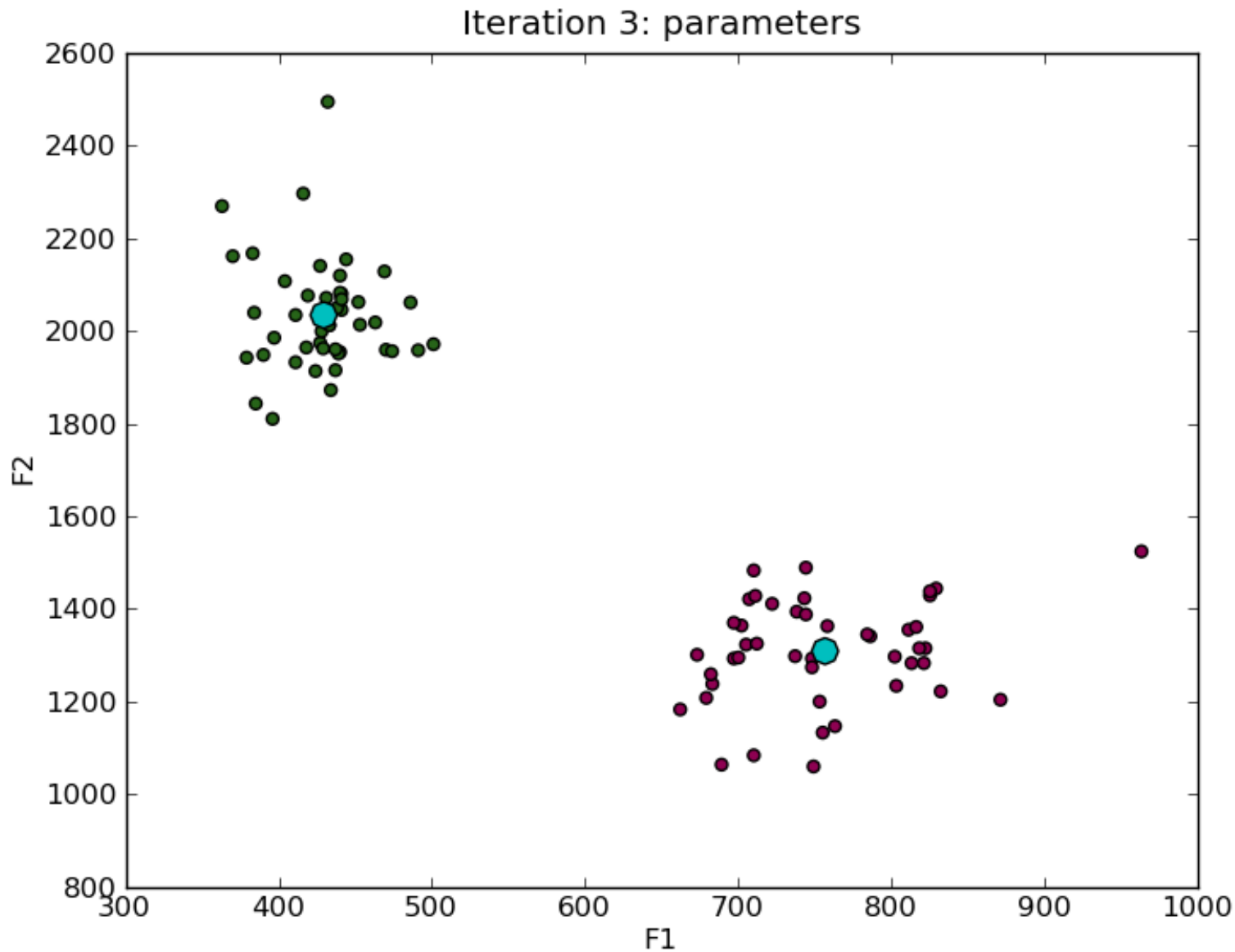
And again (estimate)



And again (classify 2)



And we're done (estimate 2)



Iterative optimization

Notice that each step improves our objective:

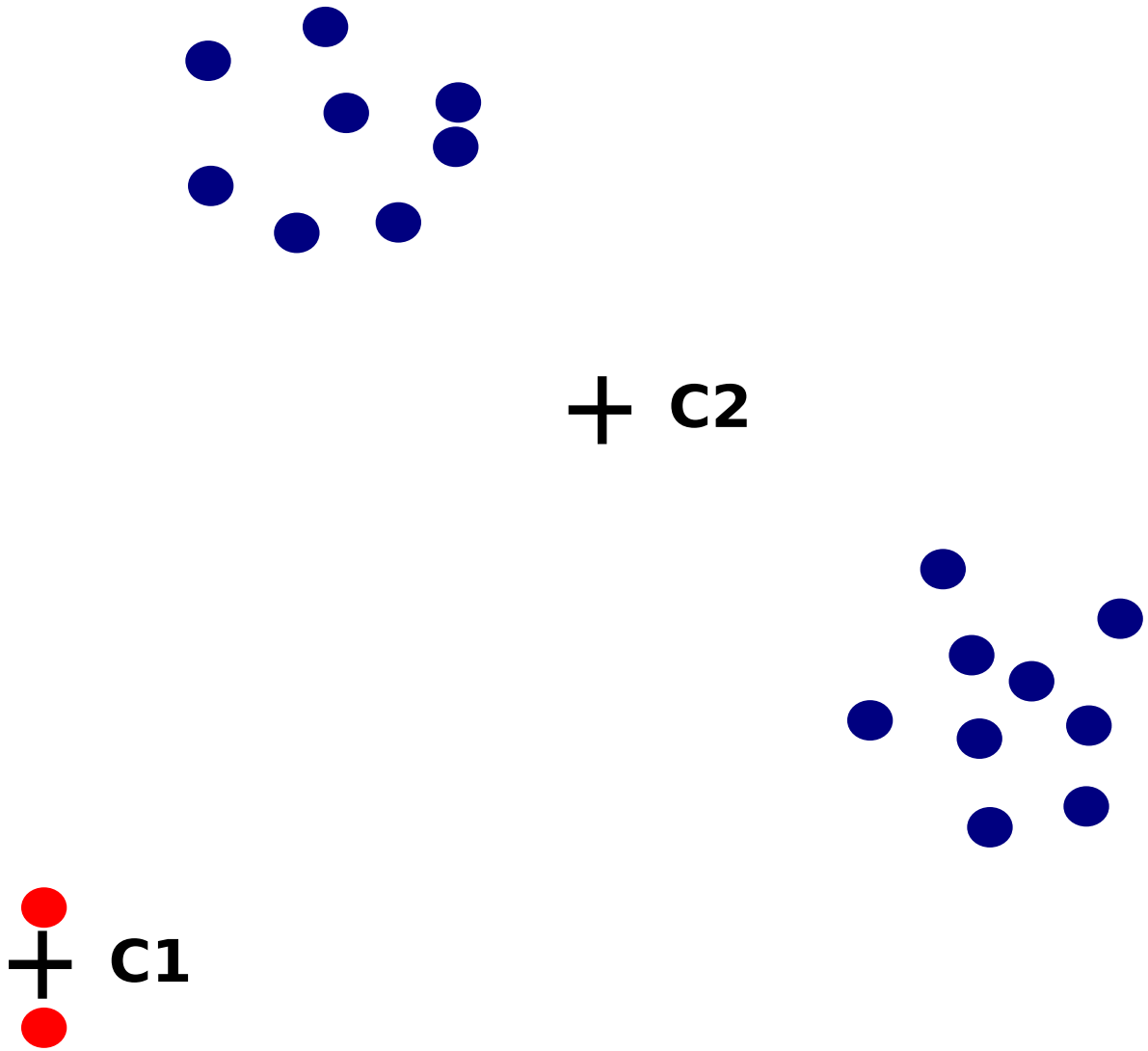
$$\min_{C_1=(c_{1,x},c_{1,y}),C_2=(c_{2,x},c_{2,y})} \sum_{d_i \in D} \min_{C^* \in [C_1,C_2]} ||d_i - C^*||_2$$

By reassigning a point d_i to a new C^*, we reduce $||d_i - C^*||_2$ for that point
By moving cluster C_1, we reduce:

$$\min_{C_1=(c_{1,x},c_{1,y})} \sum_{d_i : T_i = C_1} ||d_i - C_1||_2$$

Running the algorithm never makes things worse...
This does not mean we must converge to the best answer!

Local minimum



Theory of k-means

We can talk about probabilities instead of distances:
Suppose that probability of a point is inversely proportional to distance from the center:

$$P(d|C) \propto \exp - \left((d_x - c_x)^2 + (d_y - c_y)^2\right)$$

$\propto$ means "up to a constant"

- Remember, denominators are boring

Instead of thinking about $||d_i - C||_2$ we can think about $P(d_i|C)$

Comparison to Naive Bayes

Our objective was:

$$\min_{C_1=(c_{1,x},c_{1,y}),C_2=(c_{2,x},c_{2,y})} \sum_{d_i \in D} \min_{C^* \in [C_1,C_2]} ||d_i - C^*||_2$$

Replacing the distance, we get:

$$\max_{C_1=(c_{1,x},c_{1,y}),C_2=(c_{2,x},c_{2,y})} \sum_{d_i \in D} \max_{C^* \in [C_1,C_2]} \log P(d_i|C^*)$$

If we assume $P(C^*)$ is uniform:

- With two clusters, it's 1/2

We can put it in without changing the maximum:

$$\max_{C_1=(c_{1,x},c_{1,y}),C_2=(c_{2,x},c_{2,y})} \sum_{d_i \in D} \max_{C^* \in [C_1,C_2]} \log P(d_i|C^*)P(C^*)$$

So at this point...

This is a joint probability...

$$P(d_i|C^*)P(C^*) = P(d_i, C^*)$$

And therefore:

$$\max_{C_1=(c_{1,x},c_{1,y}),C_2=(c_{2,x},c_{2,y})} \sum_{d_i \in D} \max_{C^* \in [C_1,C_2]} \log P(d_i, C^*)$$

Under the assumptions that:

- $P(d|C)$ is proportional to $\exp(-dist)$
- $P(C)$ is uniform

k-means is trying to maximize joint likelihood of the data and classification decisions

- For a naive-Bayes-like model
- k-means is similar to "find labels such that a Naive Bayes classifier is a good model of the data"

Conclusion

- Unsupervised learning allows us to learn labels for our data without training annotations
- The labels gain meaning through “same/different” distinctions
- Various reasons to do unsupervised learning
- Clustering: classification with no labels
- K-means objective is to find centers that points are close to
- This is like finding Naive Bayes classifier that describes data points well
- Iterative EM algorithm alternates between classifying and estimating
  - Can get stuck