Homework

- Read chapter 3 in *Introduction to Machine Learning with Python*
- Classification assignment due Friday 3/23

Assignment

- Your task: write a program that can guess from a review what kind of wine is being talked about (*Cabernet Sauvignon, Merlot, Chardonnay, Sauvignon Blanc*)
- See 09-wine-project on github
- Subtask 1:
  - Read texts
  - Tokenize texts using spaCy
- Subtask 2:
  - Build baseline classifier using DummyClassifier
  - Evaluate using 10-fold cross-validation

Assignment

- Subtask 3:
  - Build a logistic regression classifier using LogisticRegression
  - Evaluate using 10-fold cross-validation over the same training/validation splits as you used for the baseline
- Subtask 4:
  - Build the best classifier you can using any method
  - Use GridSearchCV to find optimal settings for hyperparameters
  - Again, evaluate using 10-fold cross-validation over the same training/validation splits as you used for the baseline

Assignment

- Subtask 4:
  - Error analysis
    - What kinds of reviews is your classifier bad at classifying, and why?
  - Discussion
    - What have you learned about the task?
    - Is guessing the wine variety from a review hard or easy? What are the hard parts?
    - What would you need to do to score better than 90% accuracy?
- Turn in notebook via github by next Friday 3/23

Distance metrics

- A distance metric $d(x, y)$ must satisfy:
  - non-negative: $d(x, y) \geq 0$
  - $d(x, y) = 0$ iff x and y are the same
  - symmetric: $d(x, y) = d(y, x)$
  - triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$
- Jaccard distance

$$J(X, Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|}$$

Distance metrics

- Documents as term vectors
- Inner product or 'dot product' of two vectors is defined as

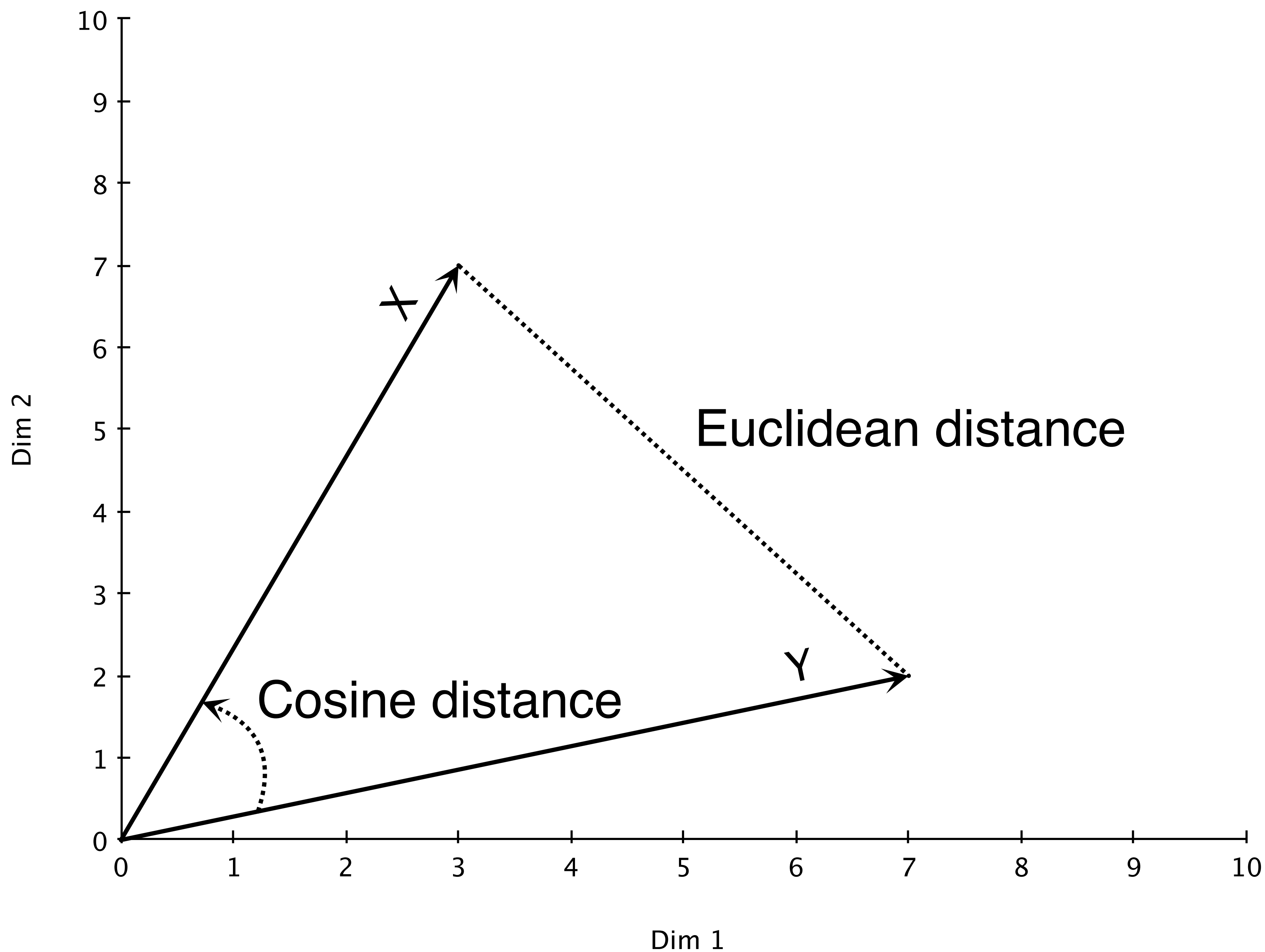
$$X \cdot Y = \sum_{i=1}^{m} x_i y_i$$

- Jaccard distance (for binary term vectors):

$$d(X,Y) = 1 - \frac{X \cdot Y}{X^2 + Y^2 - X \cdot Y}$$

Distance metrics

- Documents as term vectors



Distance metrics

- Euclidean distance

$$d(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- Cosine distance

$$\begin{aligned} d(X,Y) &= 1 - \frac{X \cdot Y}{\|X\| \, \|Y\|} \\ &= 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \end{aligned}$$

Distance metrics

- Euclidean distance
  - ranges from 0 to ∞
  - depends on absolute term frequencies and the total number of words in document
- Cosine distance
  - ranges from 0 to 1
  - only depends on relative word frequencies
- If document vectors are pre-normalized to length 1, then cosine distance is just the dot product $X \cdot Y$
- Distance and similarity are related, but not the same!

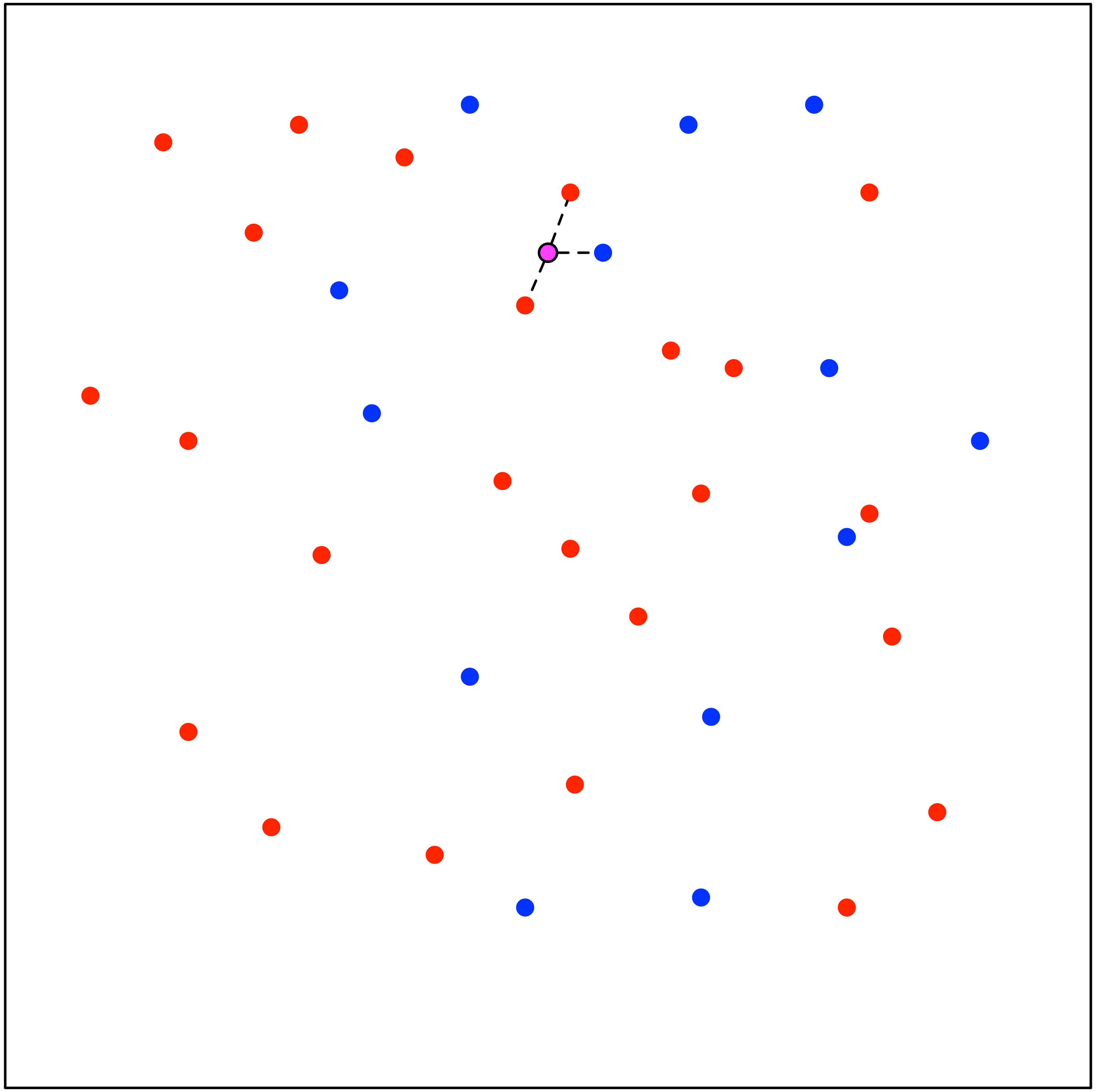
Instance-based learning

- Aka: Memory-based learning, case-based reasoning, lazy learning, . . .
- Relate new instances to the previously seen instances that they most resemble (Cover and Hart 1967, Duda and Hart 1973, Aha and Kibler 1991, Daelemans 1992–now)
- Intuitively attractive, conceptually simple, can be computationally expensive
- Success depends entirely on the choice of representation (as always)

k Nearest Neighbors

- Place training instances and query **vector space**
- The simplest version (IB1) classifies new instances by letting the *k* nearest training instances vote
- Or, for **regression**, it can average the neighbors
- Depends on using the appropriate distance metric

k Nearest Neighbors



k Nearest Neighbors

- IB1 is easily distracted by irrelevant features (imagine one feature is the class, 20 features are random)
- Using mutual information to select a subset of predict features can help
- Another variant weights the contribution of training instances inversely with their distance

Lazy learning

- kNN is **lazy** in that it doesn’t try to model the target function until there is a query
- **Eager** learners (e.g., logistic regressiob) construct an approximation immediately
- A lazy learner only needs to be accurate in the immediate neighborhood of the query, while an eager learner must be accurate everywhere
- Given the same hypothesis space, a lazy learner can represent more complex concepts than an eager learner
- Or, a lazy learner can use a simpler class of hypotheses to learn similar concepts

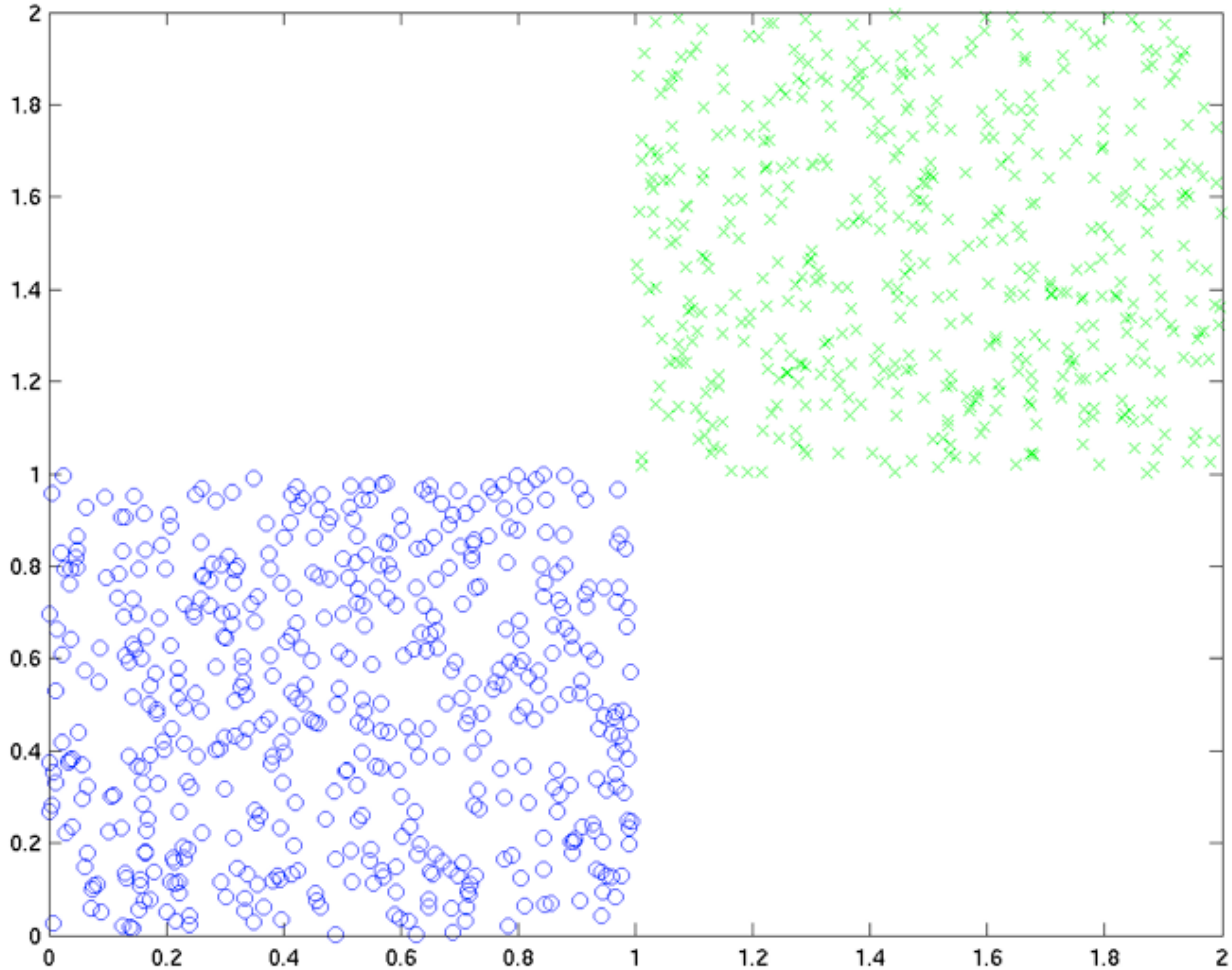
Lazy learning

- Daelemans, et al. (1999) claims lazy learners are particularly well suited to NLP problems
- Language is ‘lumpy’, with lots of different fairly regular patterns: *-ed, -d*
- Even linguistic exceptions include lots of subregularities: *sell*/*sold*, *tell*/*told*
- Even highly exceptional forms need to be remembered: *be*/*was*
- Irregularities can be hard to distinguish from noise
- Analogies

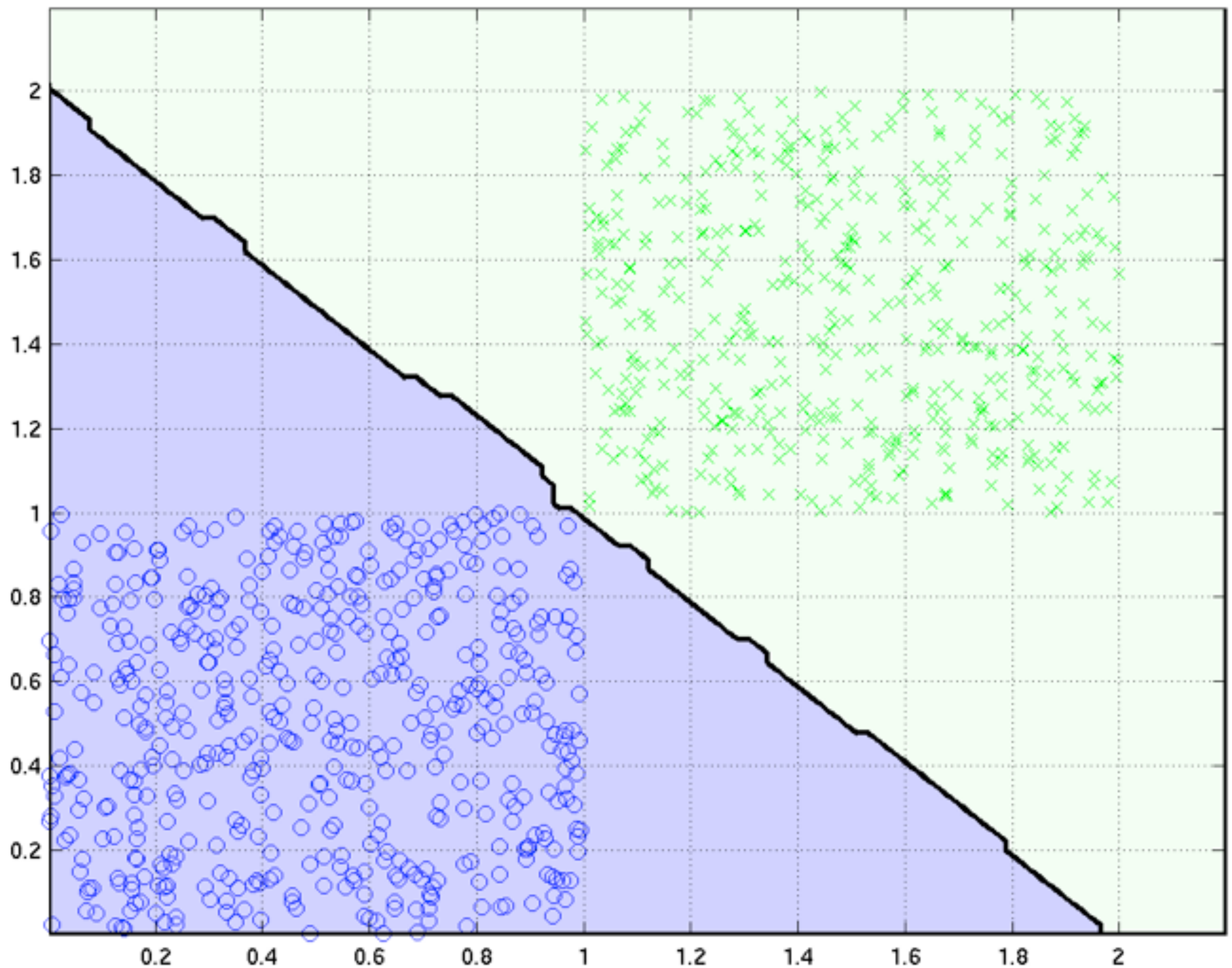
Instance indexing

- The computational cost of the training phase in IBL is minimal
- Classification can be expensive, if the training set is very large, but careful indexing can help
- Database of instances can be organized into a tree, ordered by gain ratio – this gives compact storage and quick access to similar examples
- An **inverted index** lists training instances which are relevant by feature – this allows quick access to relevant examples

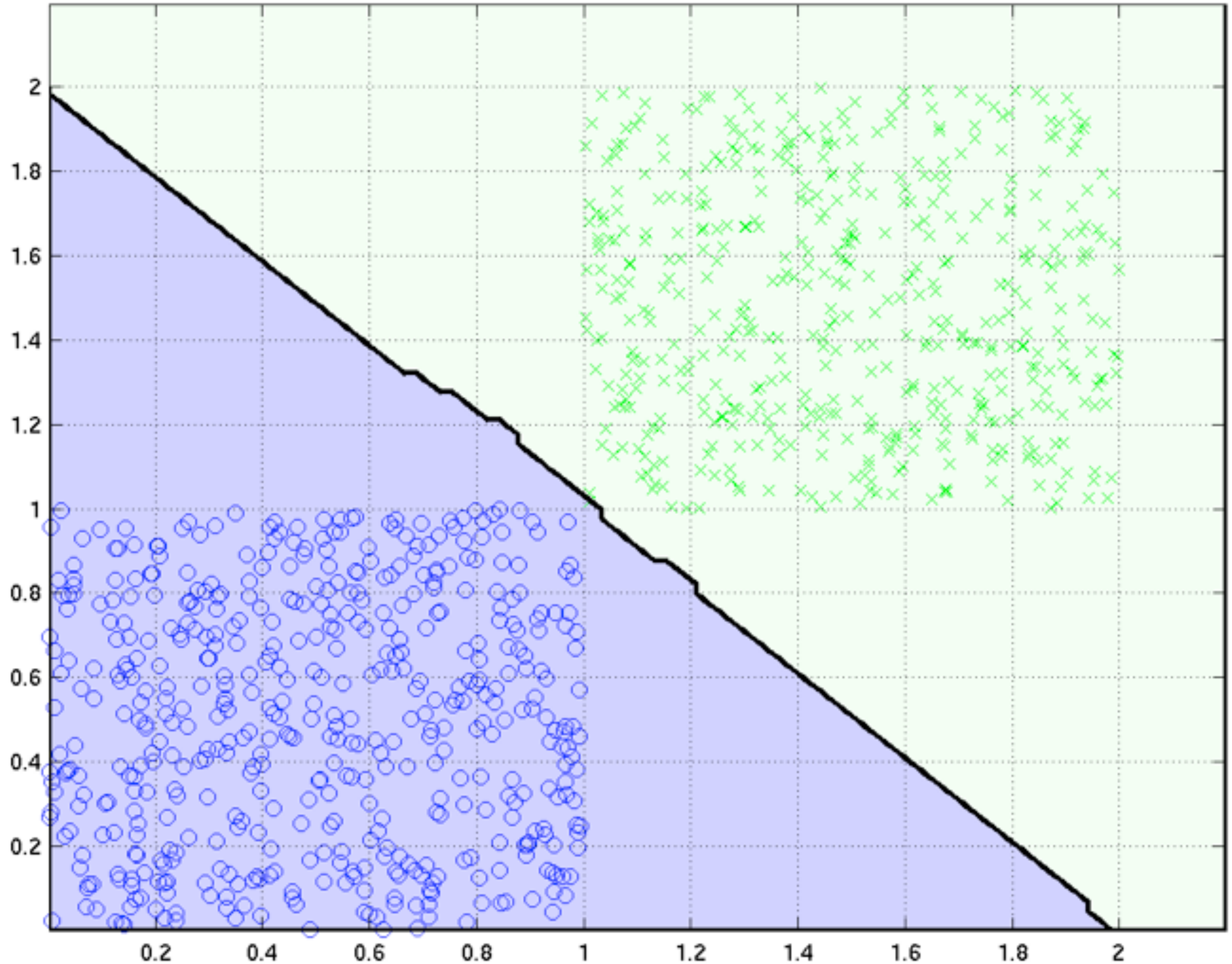
Decision boundaries



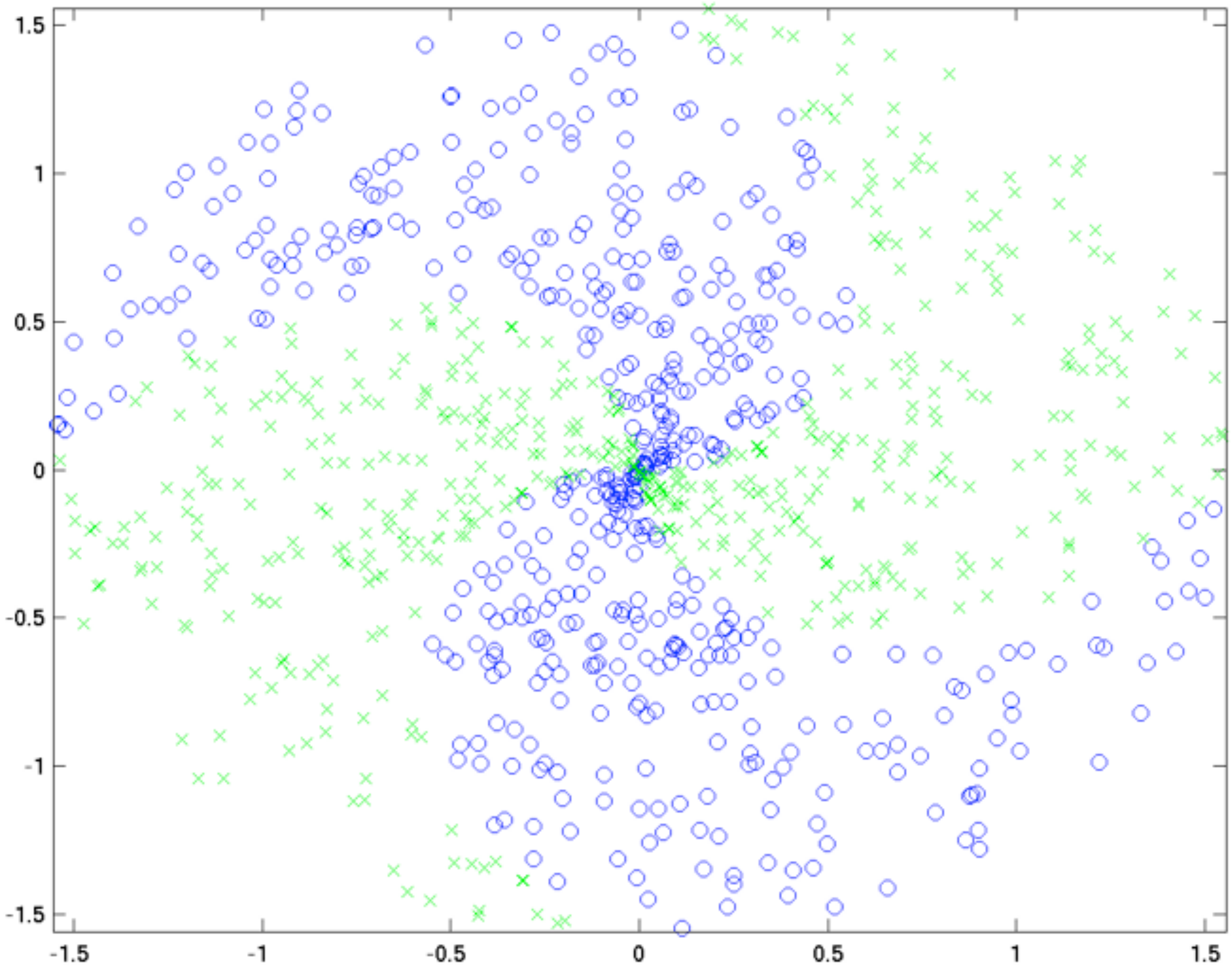
Decision boundaries k=1



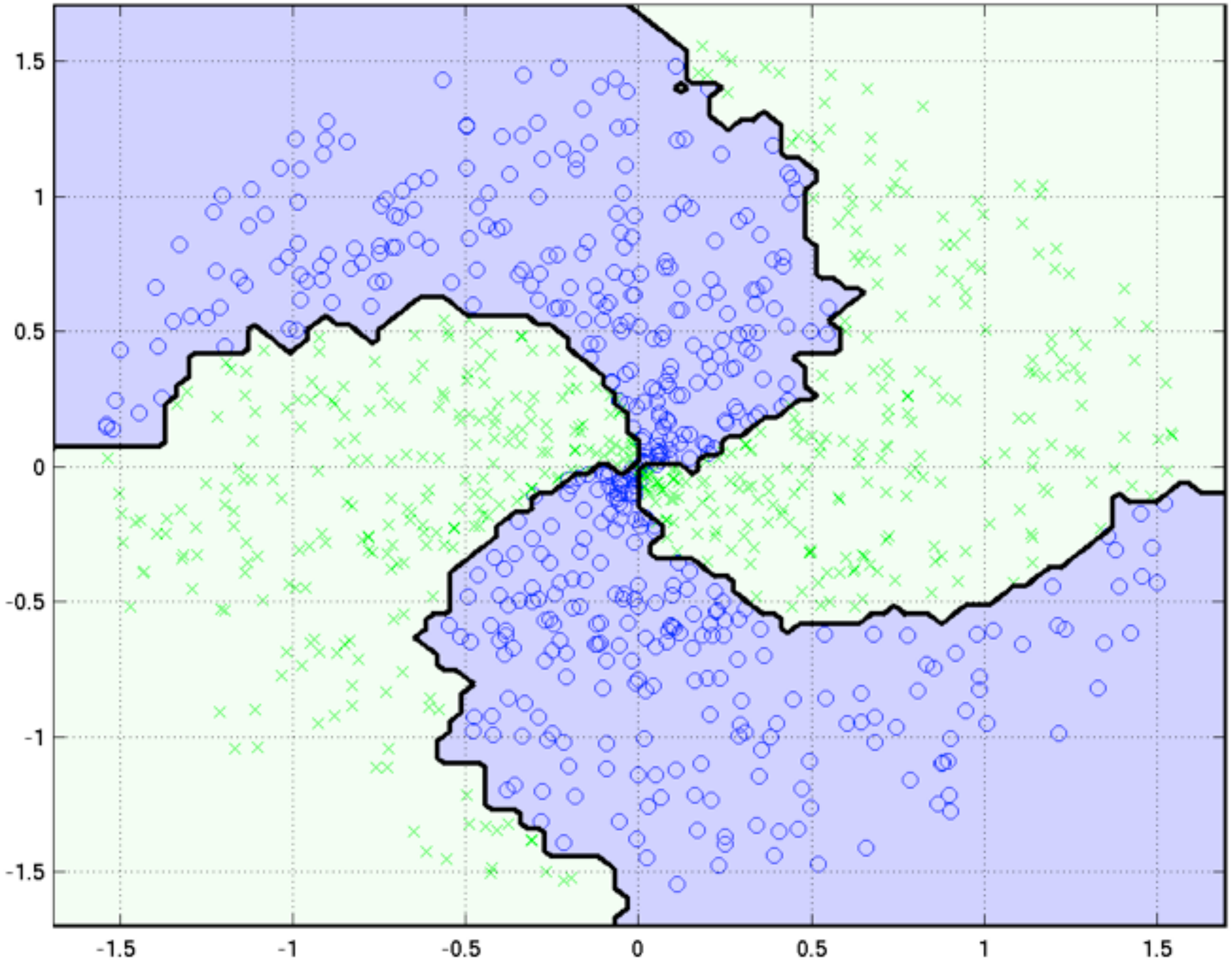
Decision boundaries k=5



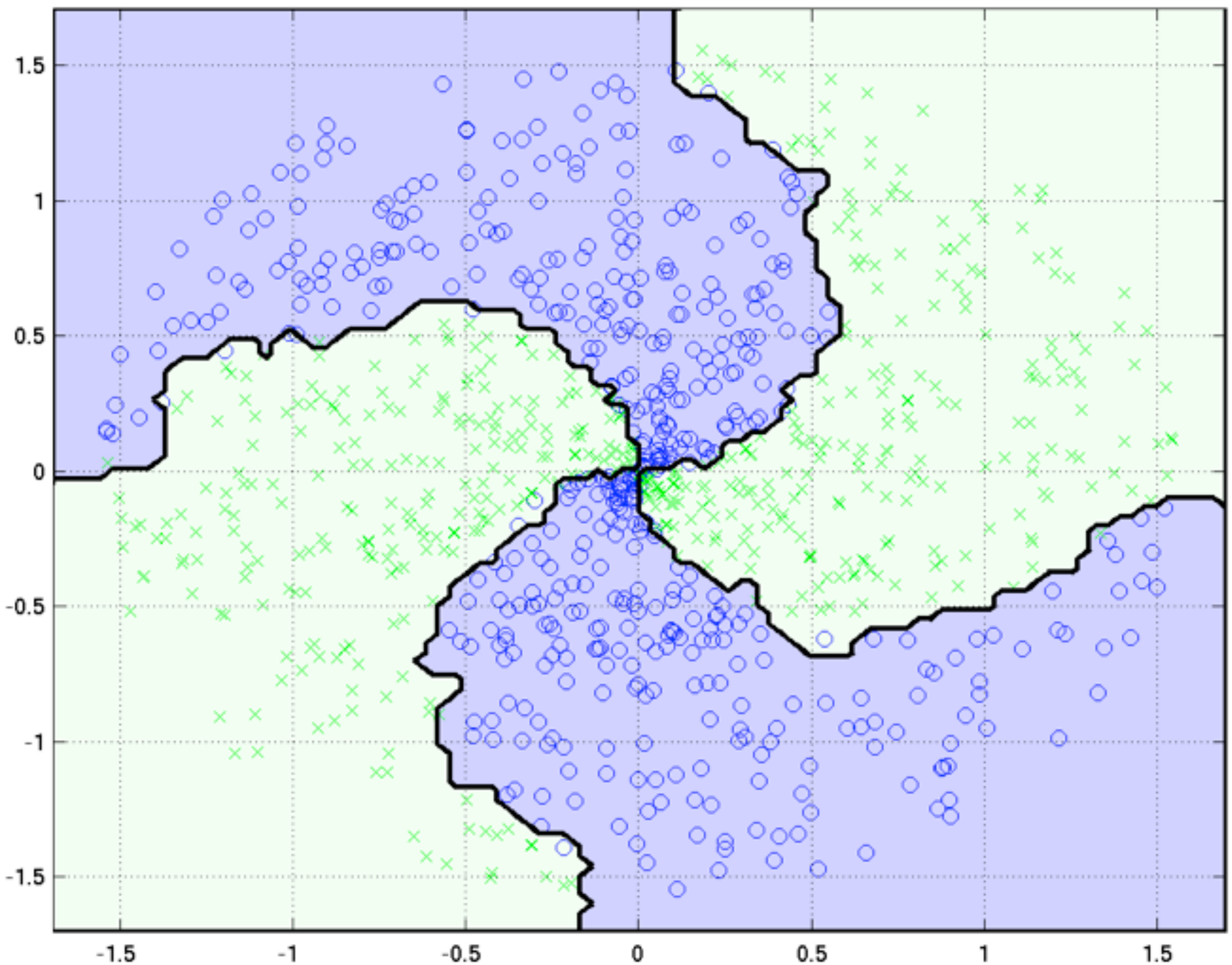
Decision boundaries



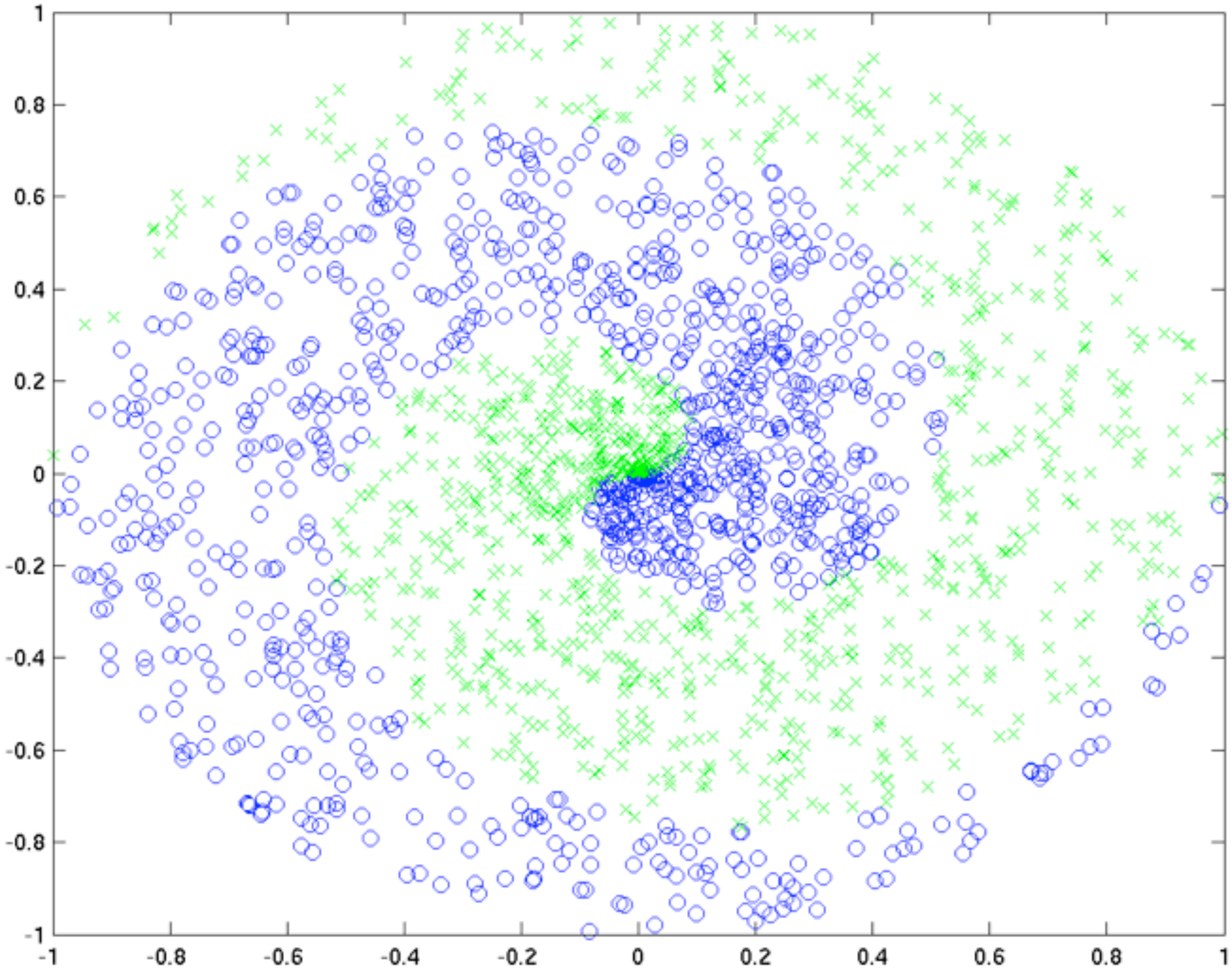
Decision boundaries k=1



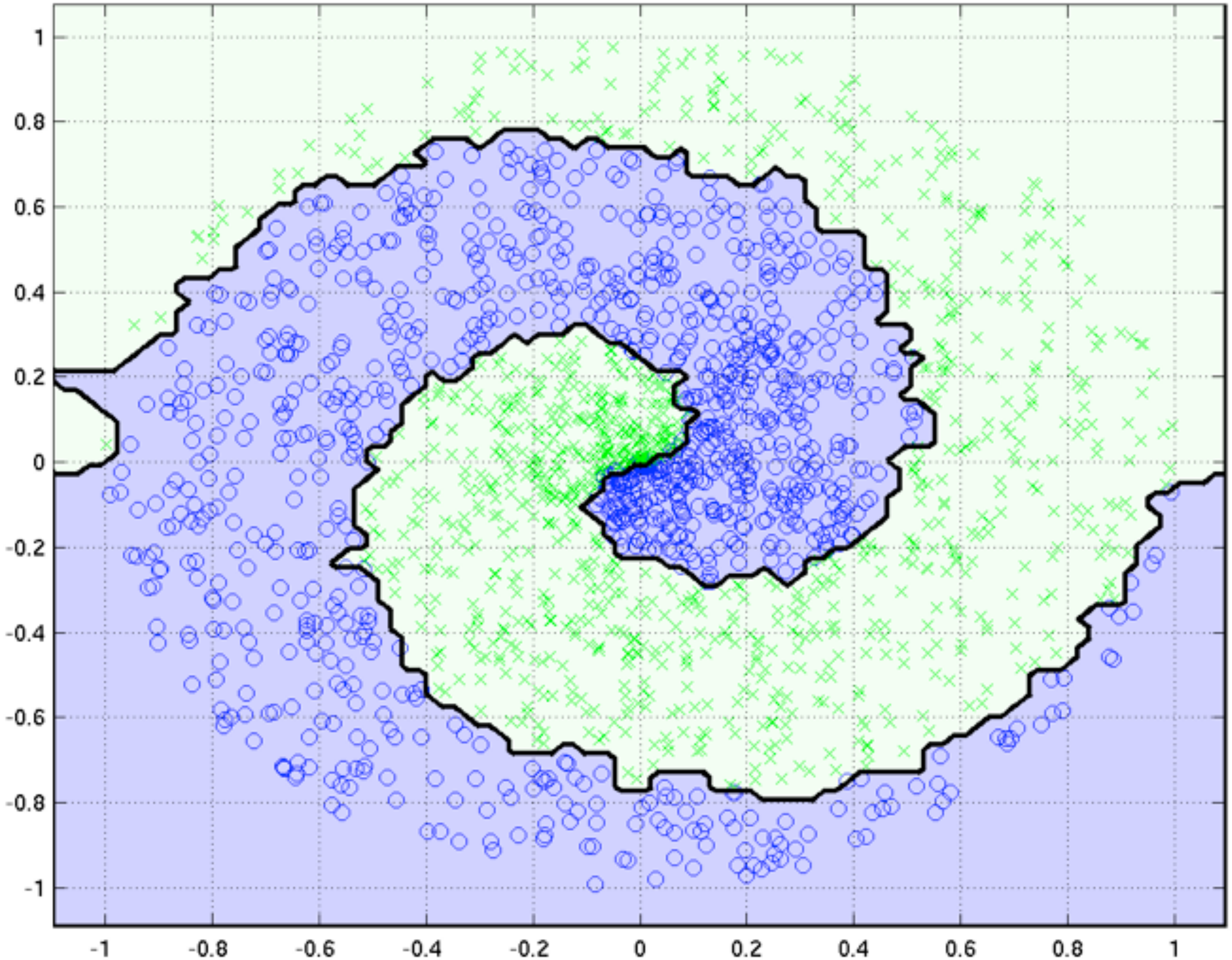
Decision boundaries k=5



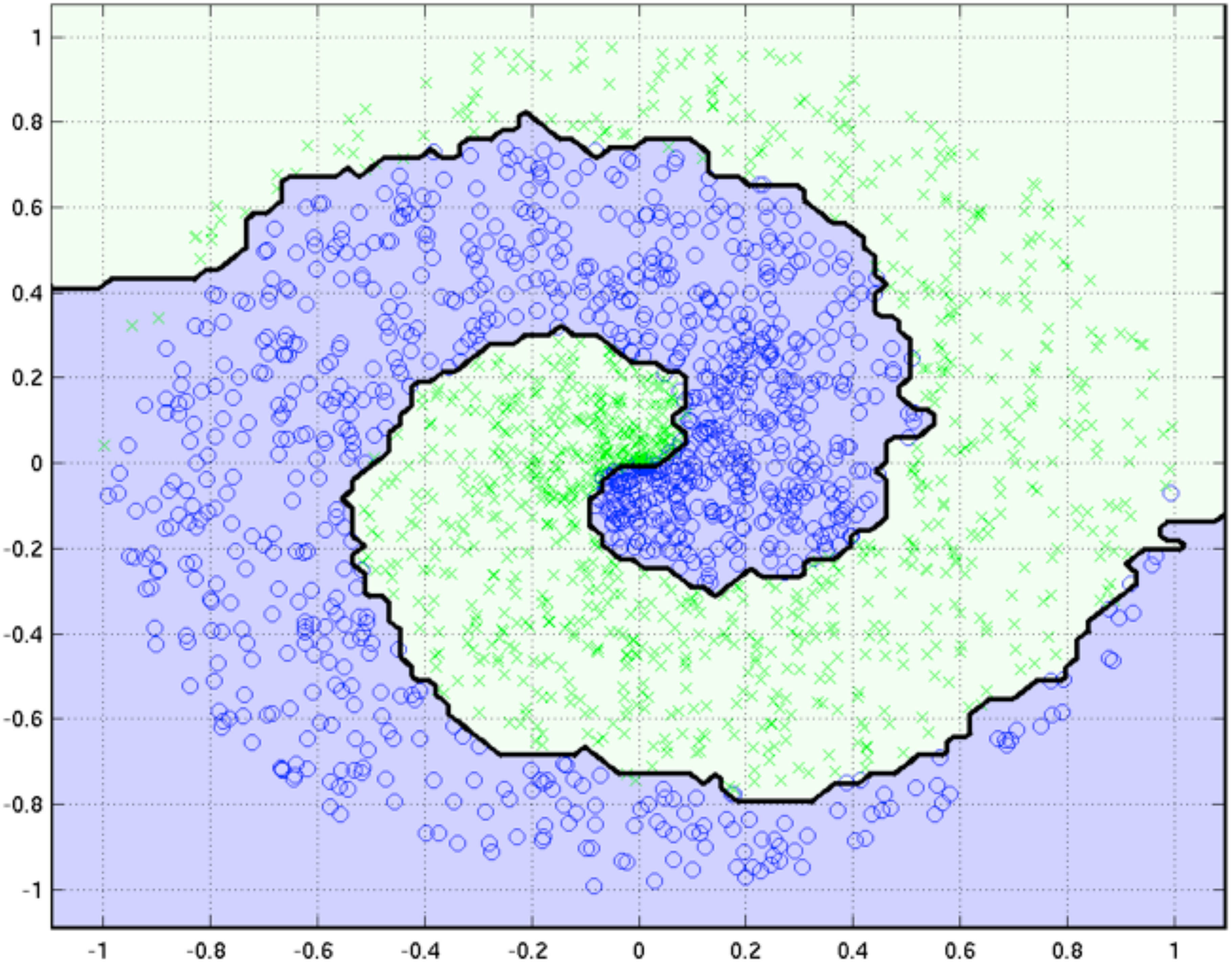
Decision boundaries



Decision boundaries k=1



Decision boundaries k=5



Clustering

- Clustering is the unsupervised analog of classification: group texts into categories
- Evaluation is hard, since accuracy, precision, recall are meaningless without ground truth
- ‘Ecological’ evaluation
- Silhouette Coefficient

Clustering

- Silhouette Coefficient
  - *a* = the mean distance between a sample and all other points in the same class.
  - *b* = The mean distance between a sample and all other points in the next nearest cluster.
  - The Silhouette Coefficient *s* for a single sample is then given as:

$$s = \frac{b - a}{max(a, b)}$$

  - The Silhouette Coefficient for a set of samples is given as the mean of the Silhouette Coefficient for each sample

K Means

- K Means clustering organizes items into *k* clusters represented by centroids
- Each item is in the cluster with the closest centroid
- Start with randomly distributed centroids
- http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html

K Means

- Lloyd's algorithm
  1. Choose *k* centroids at random
  2. Repeat until converged:
     1. Assign documents to cluster whose centroid is closest
     2. Recompute cluster centroids
- Results depend (a lot) on initial guess
  - Not guaranteed to converge
  - Won't find an optimal solution
  - Run multiple times and average the solutions?