

Predicting code quality: how?

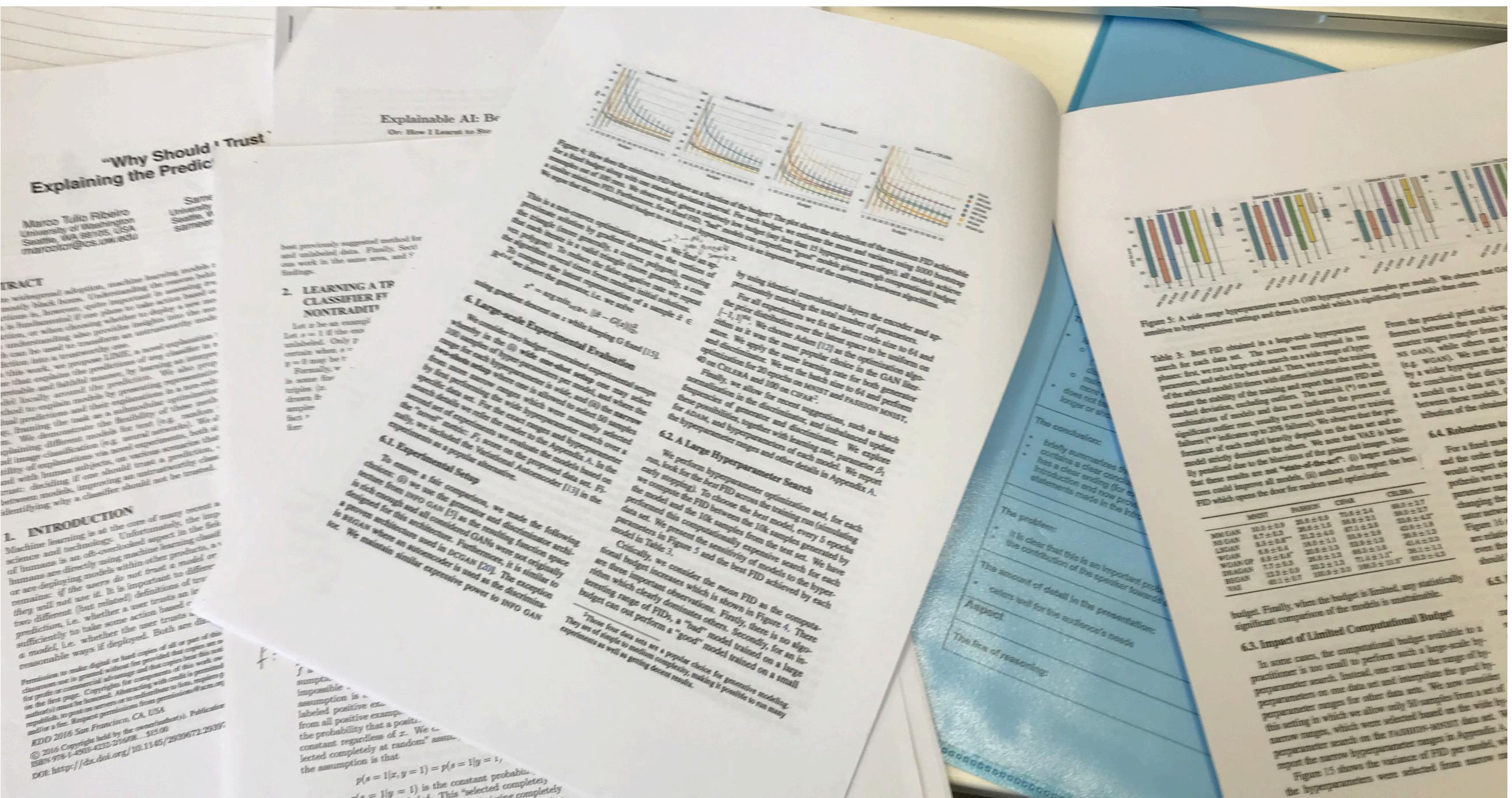
A collage of various code snippets and interfaces, illustrating different types of software development:

- Java Code Snippet:** A snippet of Java code showing a method `arithmetic_average` that calculates the average of elements in a 2D integer array.
- Python Code Snippet:** A snippet of Python code that iterates through a list of users, prints their screen names, and calculates the total number of tweets over a specific time window.
- Python Function:** A snippet of Python code defining two functions: `add5` which adds 5 to its argument, and `dotwrite` which prints a tree structure of nodes.
- Excel VBA Macro:** A snippet of VBA code for Excel, defining a macro named `Macro1` that activates a specific worksheet and changes the fill color of a shape based on a value.
- Terminal Window:** A terminal window displaying a large amount of command-line text, likely a log or a dump of system data.



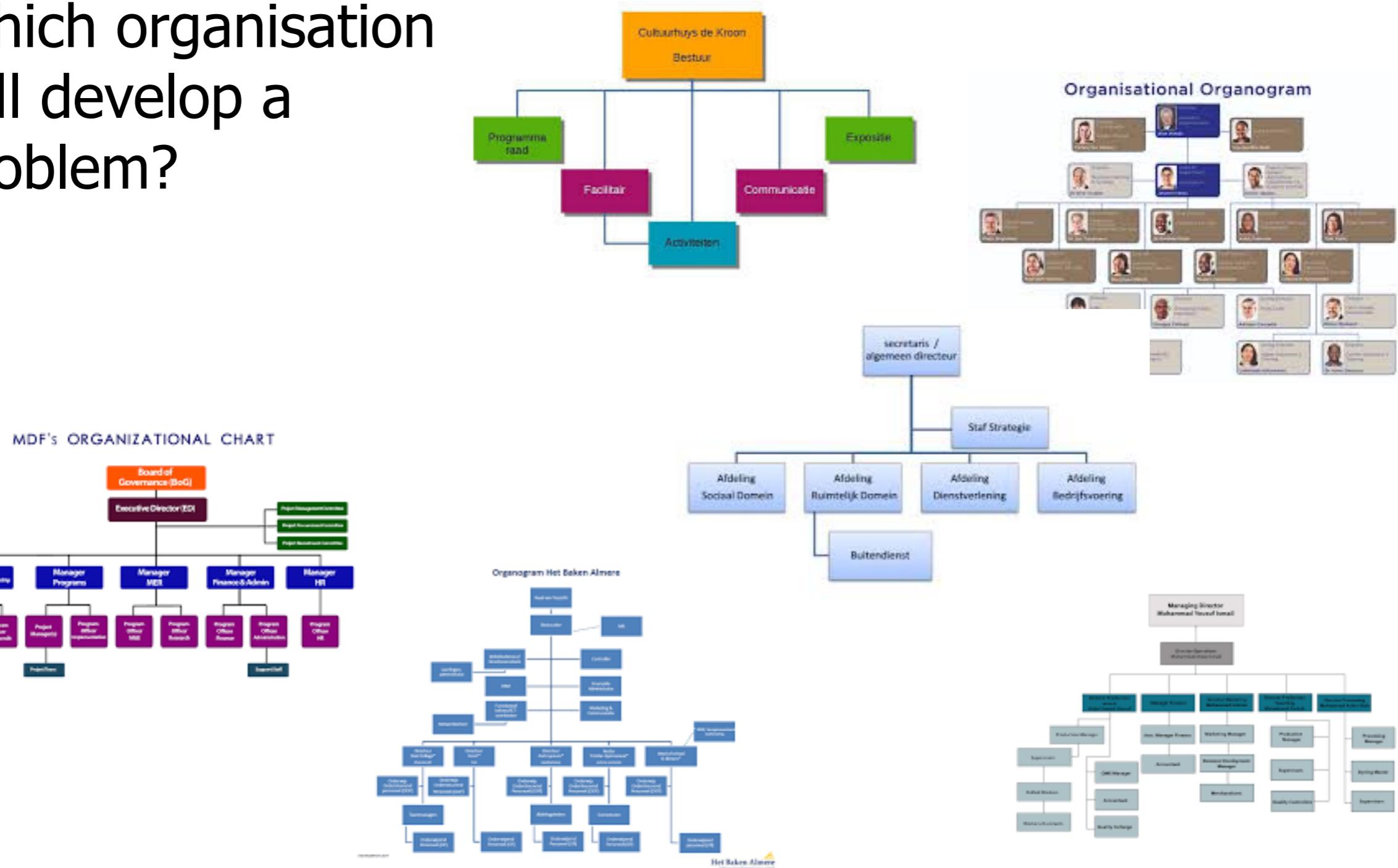
Classify project reports?

- Given that the structure of the texts are, more or less, comparable, how to (automatically) evaluate/classify?



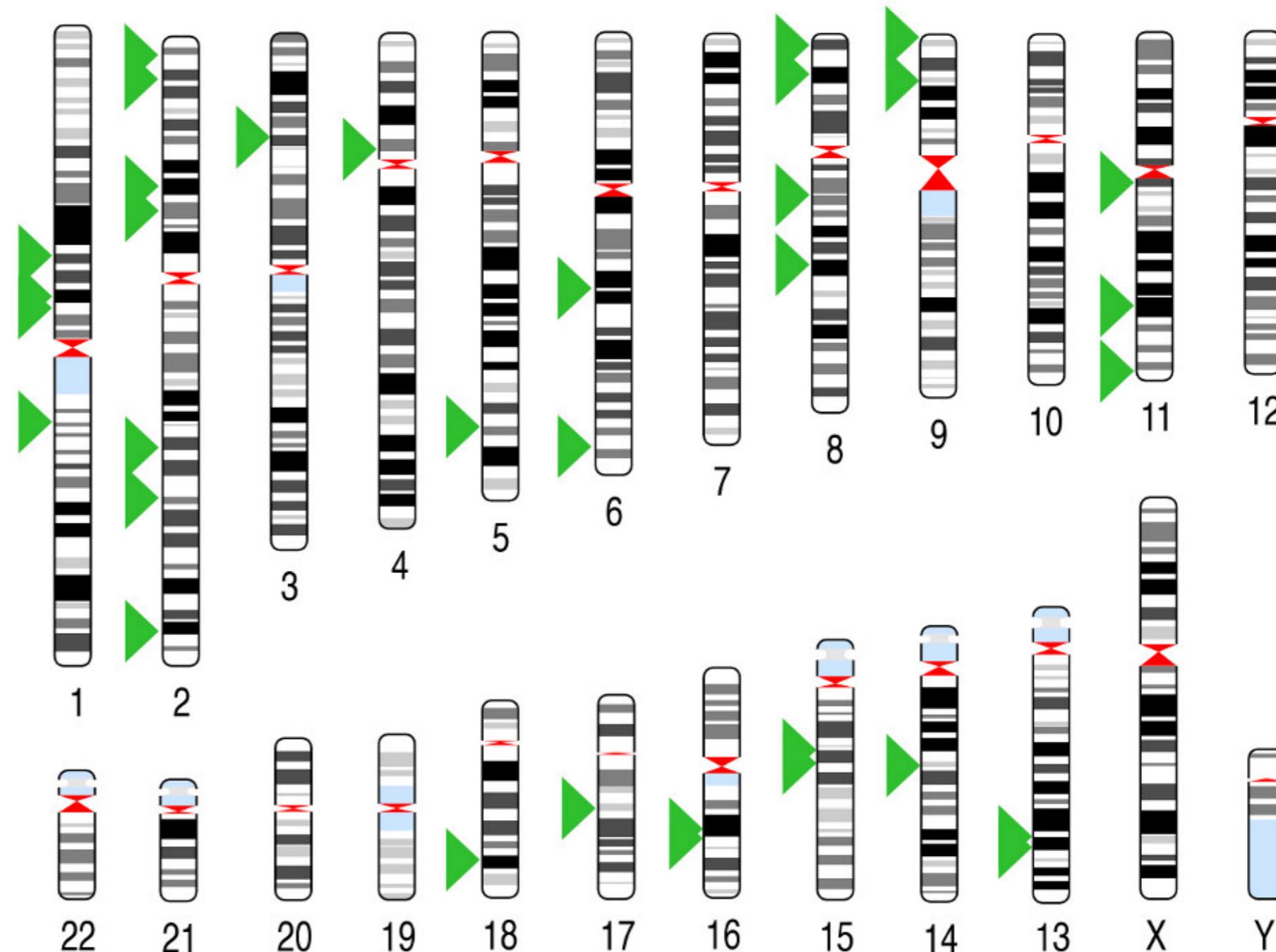
Representation for organisations?

- Which organisation will develop a problem?

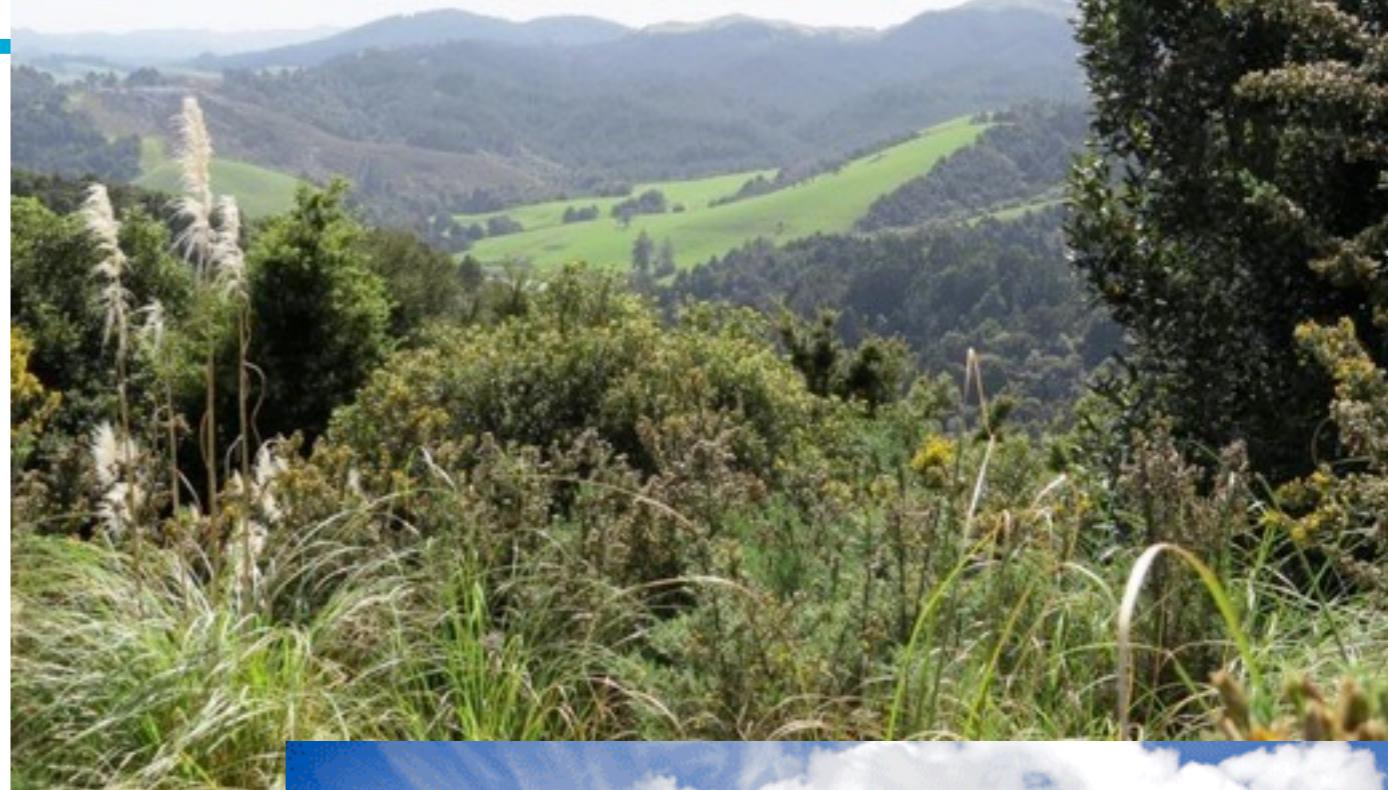


For tumor classification?

- Classify a piece of tumor tissue, given mutations in the genome



Concept to learn?



Multiple-Instance Learning

or:

Set Classification

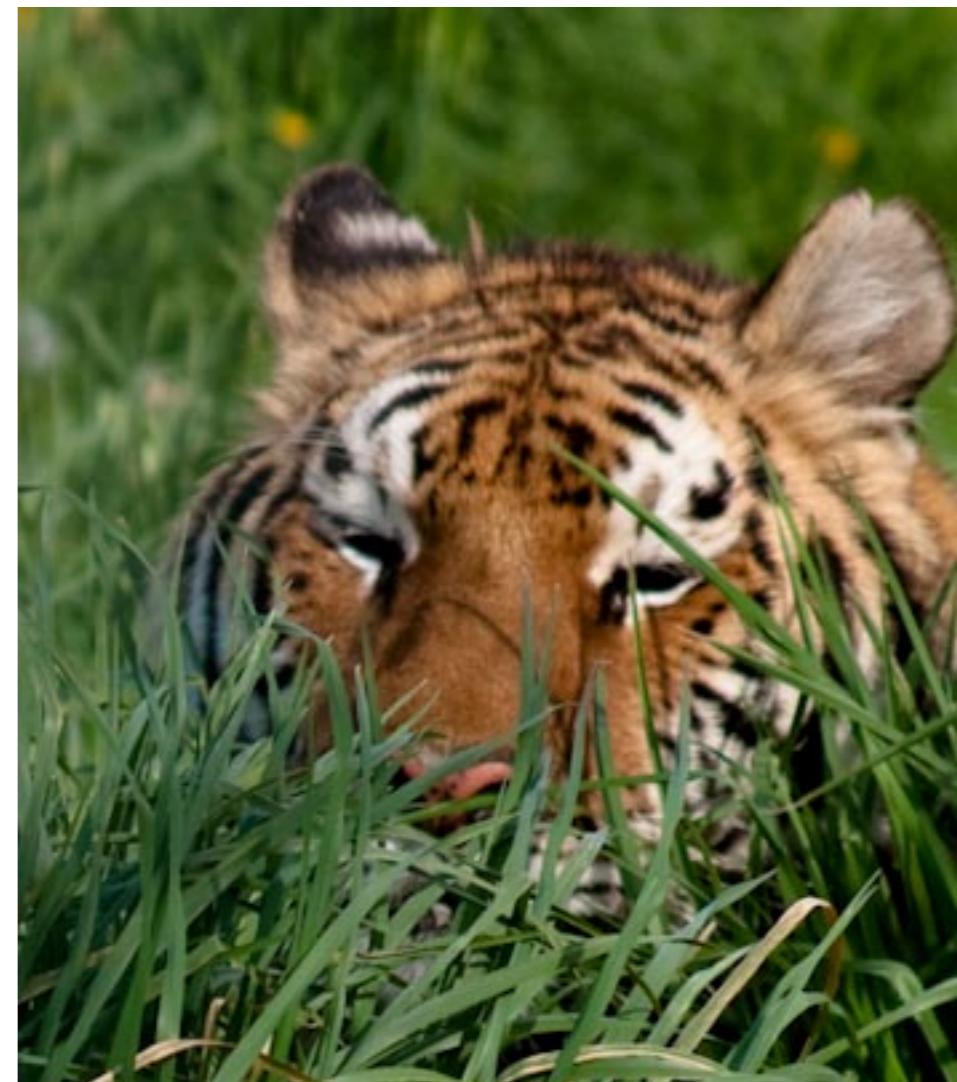
D.M.J. Tax

Pattern Recognition Laboratory
Delft University of Technology

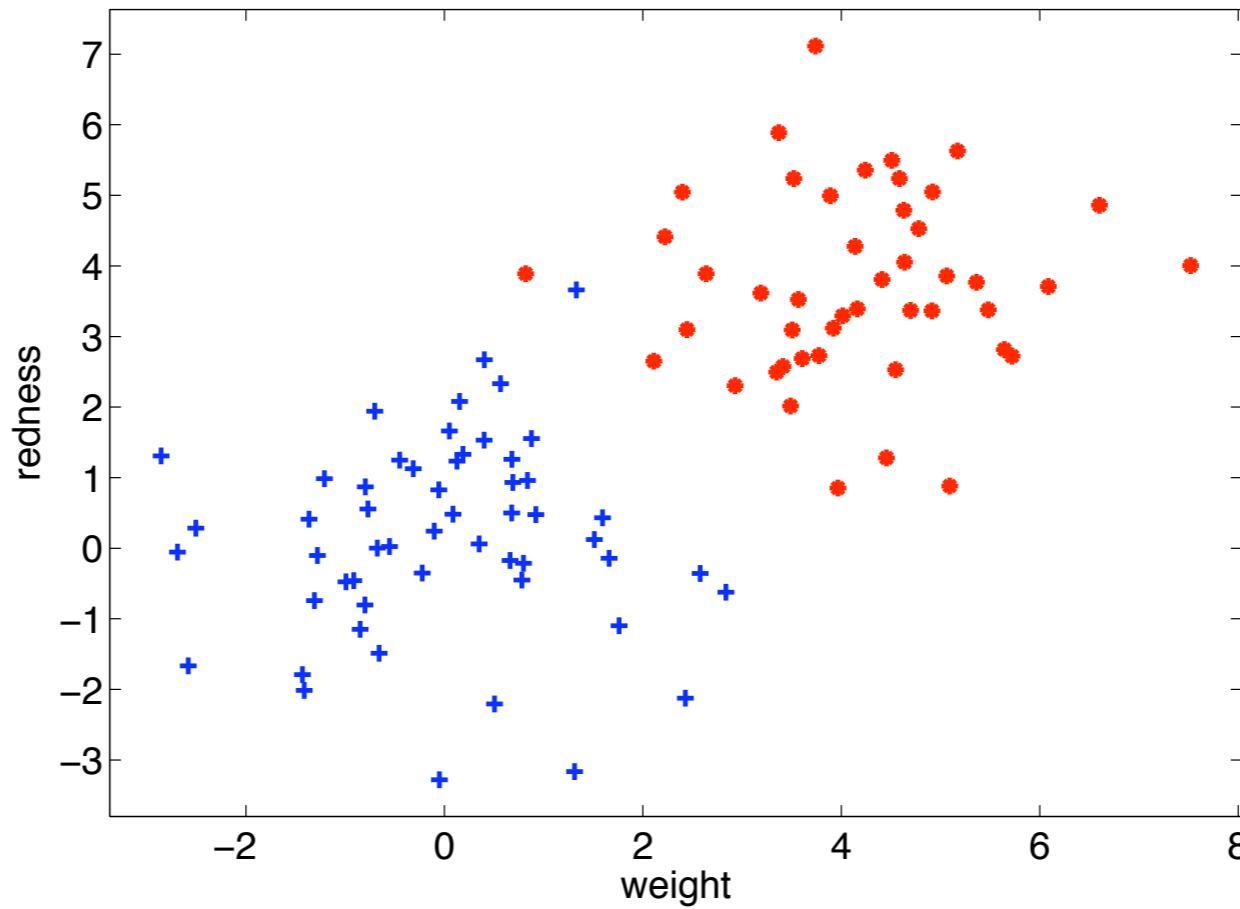


Multiple instance learning (MIL)

- Multiple-instance learning, background history
- 'Pure' MIL Algorithms:
 - Diverse Density
 - naive approach, MI-SVM, MIL-Boost
- MIL bag-representations
 - Bag vectors
 - Bag similarities
 - 'Bag of Words'
 - MILES
- Some applications/future research questions



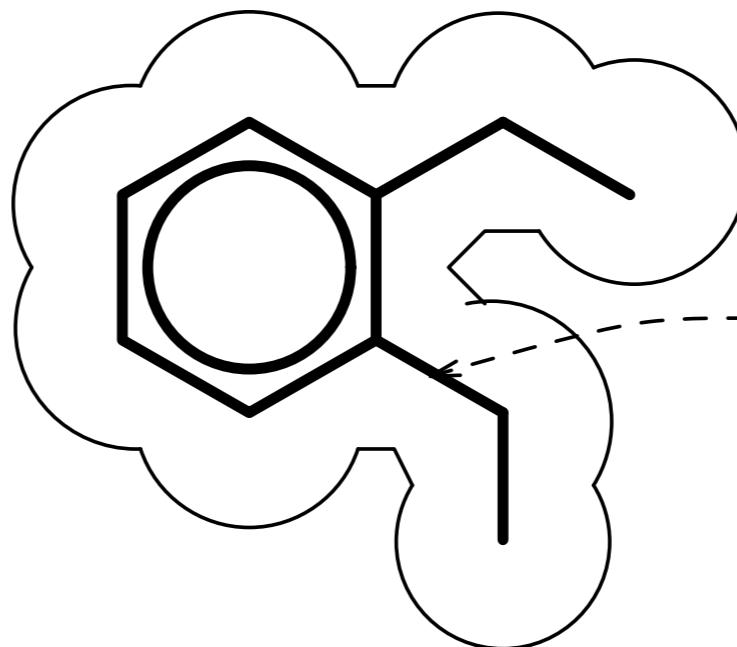
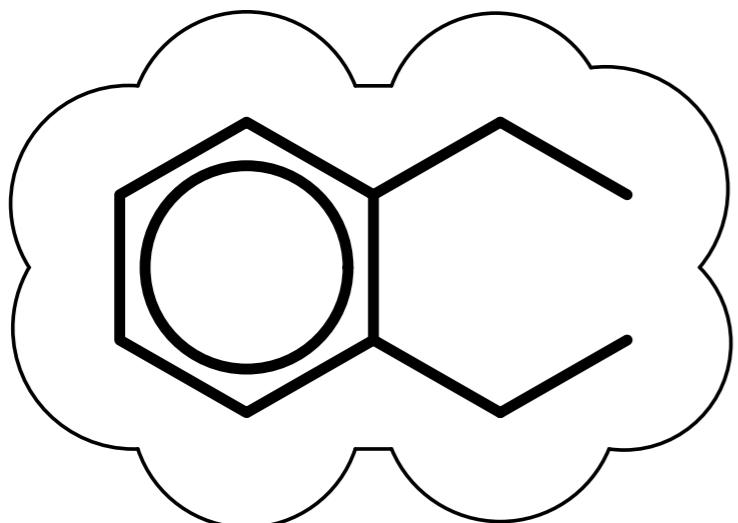
Traditional classification



- Objects are represented by a (fixed) set of features
- Objects have (one) label
- Confusion between classes caused by noisy measurements and poor features



Origin: Drug discovery, MUSK



Rotated Bond

- Predict if a molecule smells 'musky'
- Chemical formula is fixed, physical shape can change: different **conformations** are possible.
- When one conformation smell musky, the molecule smells musky
- Each conformation is an 'instance'



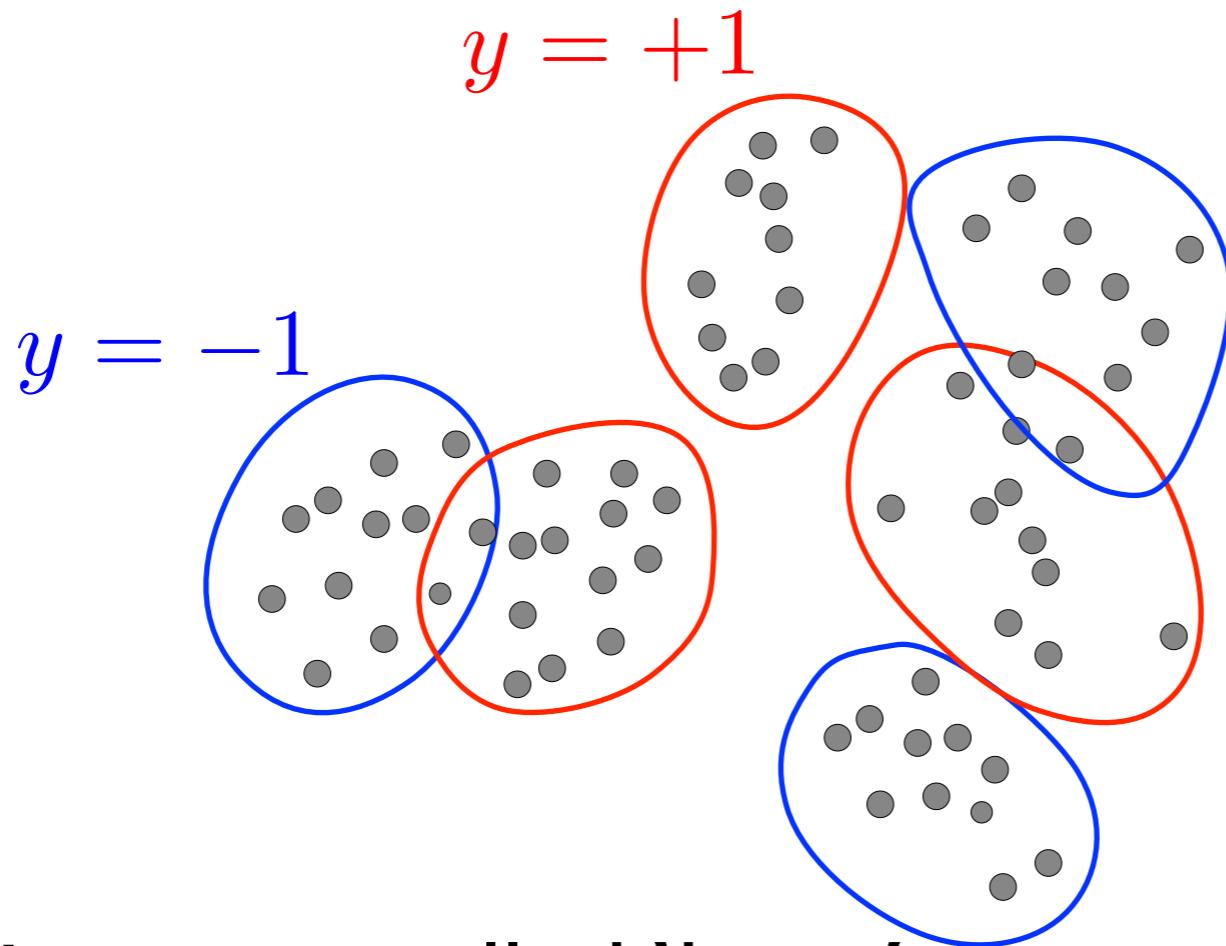
Multi(ple)-instance learning



- In multi-instance learning, it is assumed that a set (or **bag**) of objects (**instances**) is labeled
- All instances are represented by feature vectors in the same feature space
- It is a two-class problem:
 - In a negative bag, all instances are assumed to be negative
 - In a positive bag, some (or, at least one) instance is assumed to be positive



Multiple-instance learning



- Objects are called 'bags'
- Feature vectors are called 'instances'
- The bags are labeled, **not** the instances



Notation

- Assume we have N bags of instances
- Each bag B_i has n_i instances

$$B_i = \{\mathbf{x}_{i1}, \dots, \mathbf{x}_{ij}, \dots, \mathbf{x}_{in_i}\}$$

- In training, each bag is labeled

$$\{(B_i, y_i), \quad i = 1, \dots, N\}$$

where

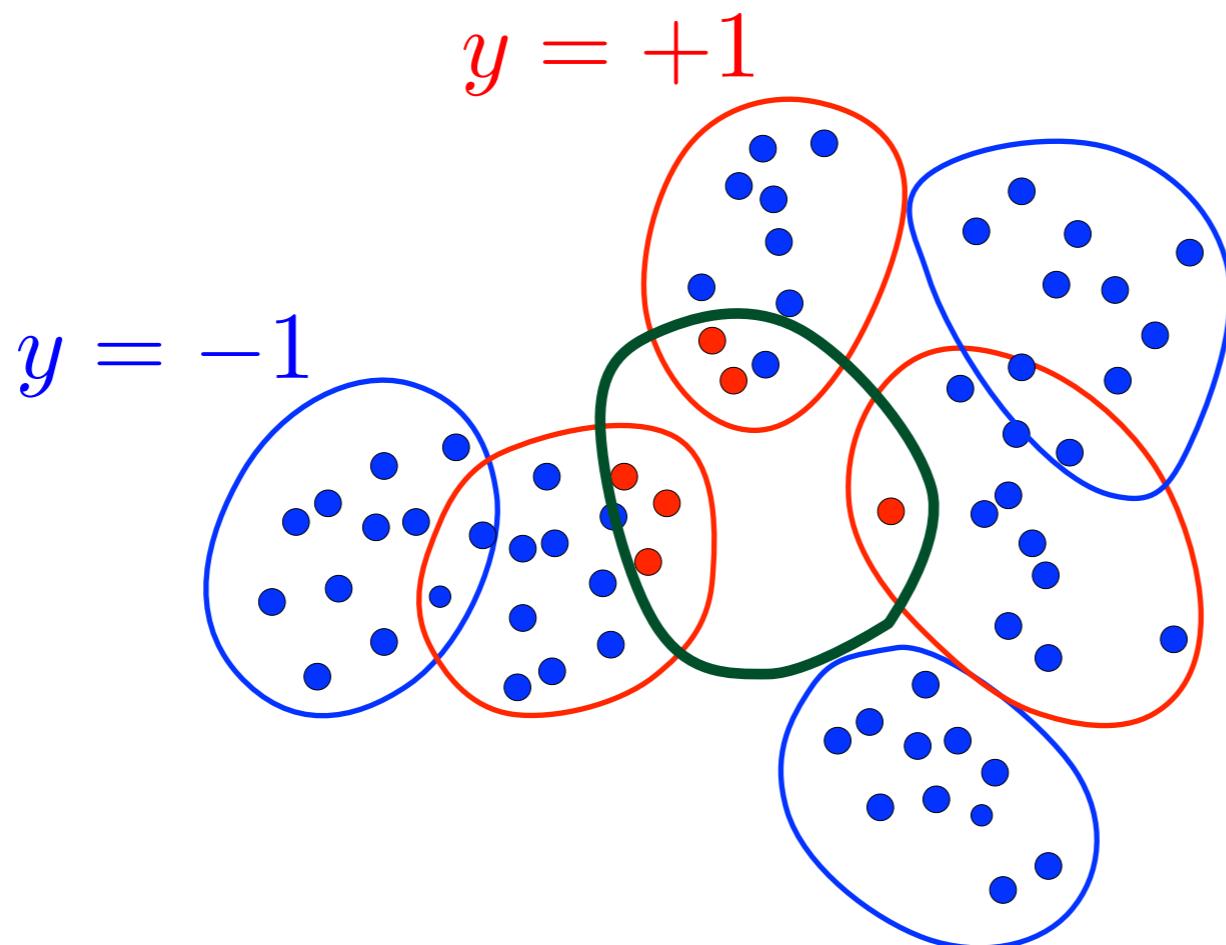
$$y \in \{-1, +1\}$$

- We are asked to find a bag-classifier:

$$\hat{y}_i = f(B_i)$$



Classical: find the concept

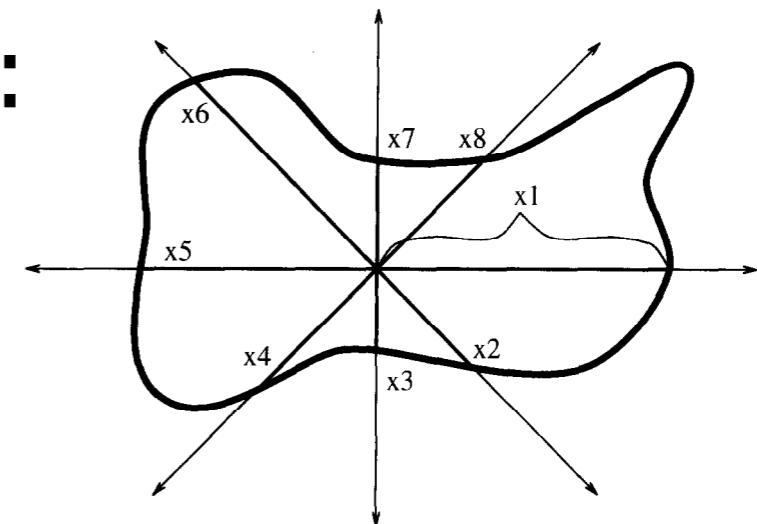


- Classically: some instances belong to a 'concept', or not
- Learning means finding the 'concept'



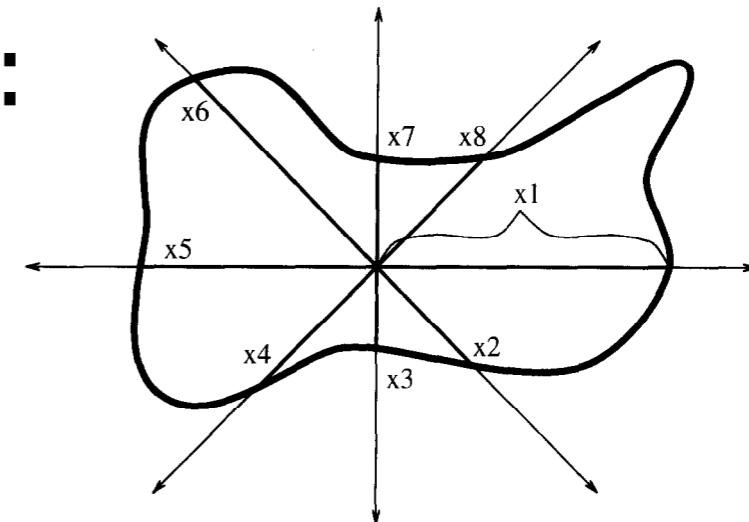
Original solution drug discovery

- Make a 3D model of the molecule: each conformation is another model
- Describe the shape by 'rays'

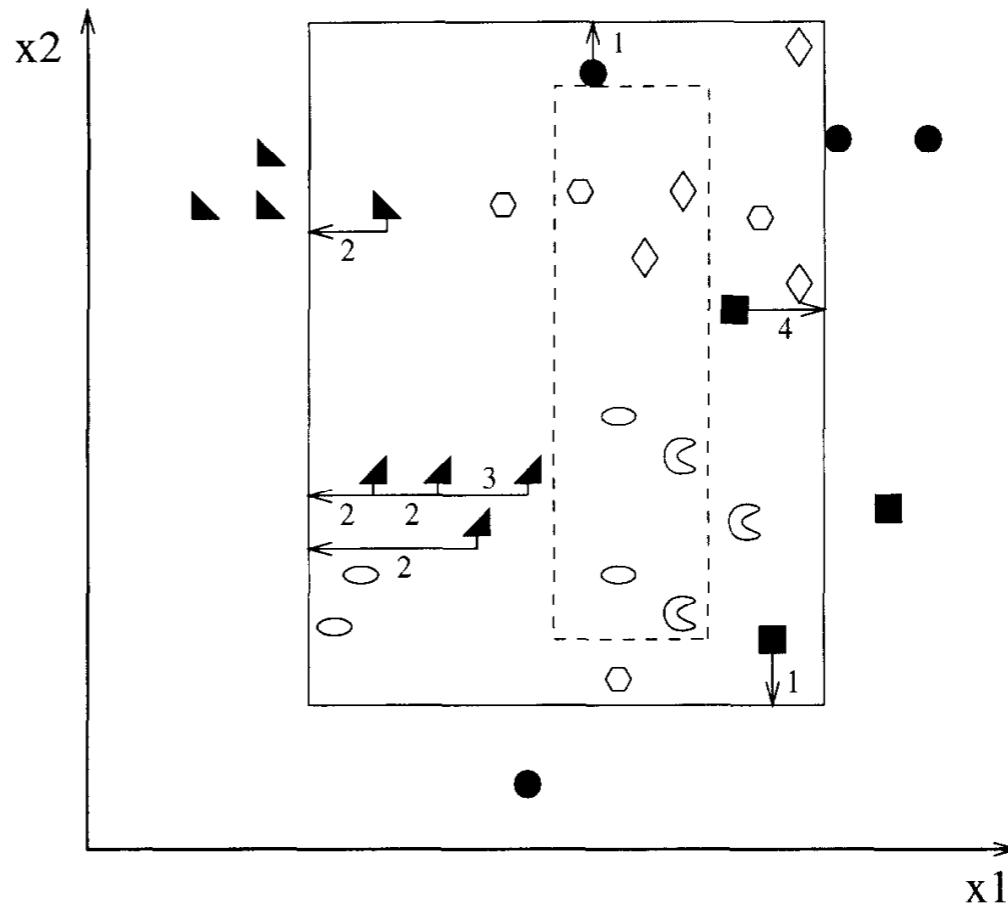


Original solution drug discovery

- Make a 3D model of the molecule: each conformation is another model
- Describe the shape by 'rays'



- Now fit the axis-parallel rectangle (see lecture on computational learning theory)
- Take the MIL assumption into account

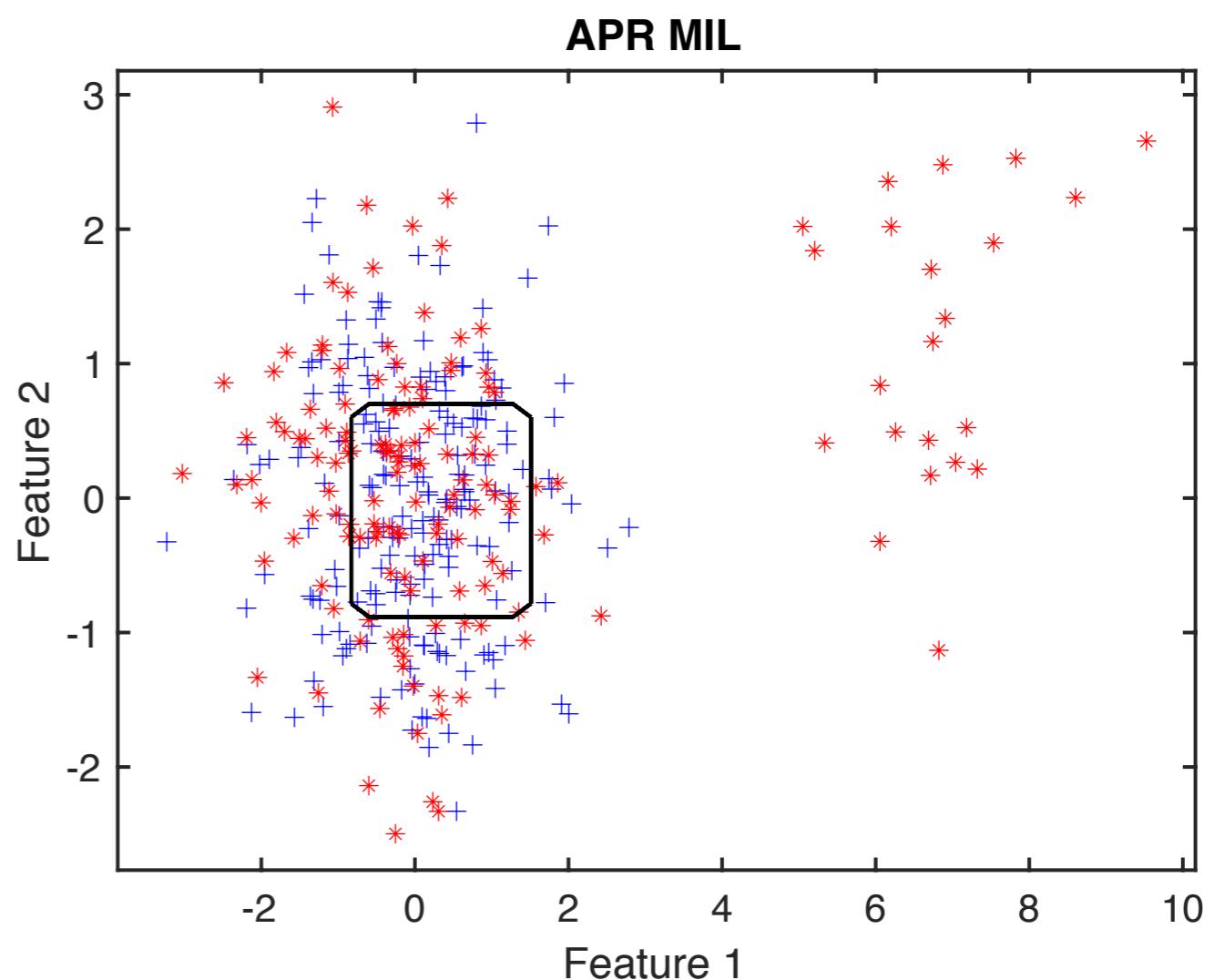


Solving the multiple instance problem with axis-parallel rectangles
Thomas G. Dietterich, Richard H. Lathrop, Tomás Lozano-Pérez Artificial Intelligence, 1997



On artificial data

- 10 positive, 10 negative bags
- Instances are drawn from two Gaussian distributions
- Positive bags have instance in the right Gaussian (the concept)
- How can you plot the decision boundary?



MIL methods

- Dietterich's approach did not work so well
- Many different approaches:
 - Often quite heuristic (but it works)
 - Sometimes nicely founded (but hard to optimise)
- Naive approach
- Diverse Density
- MI-SVM
- Boosting approach
- MILES and friends



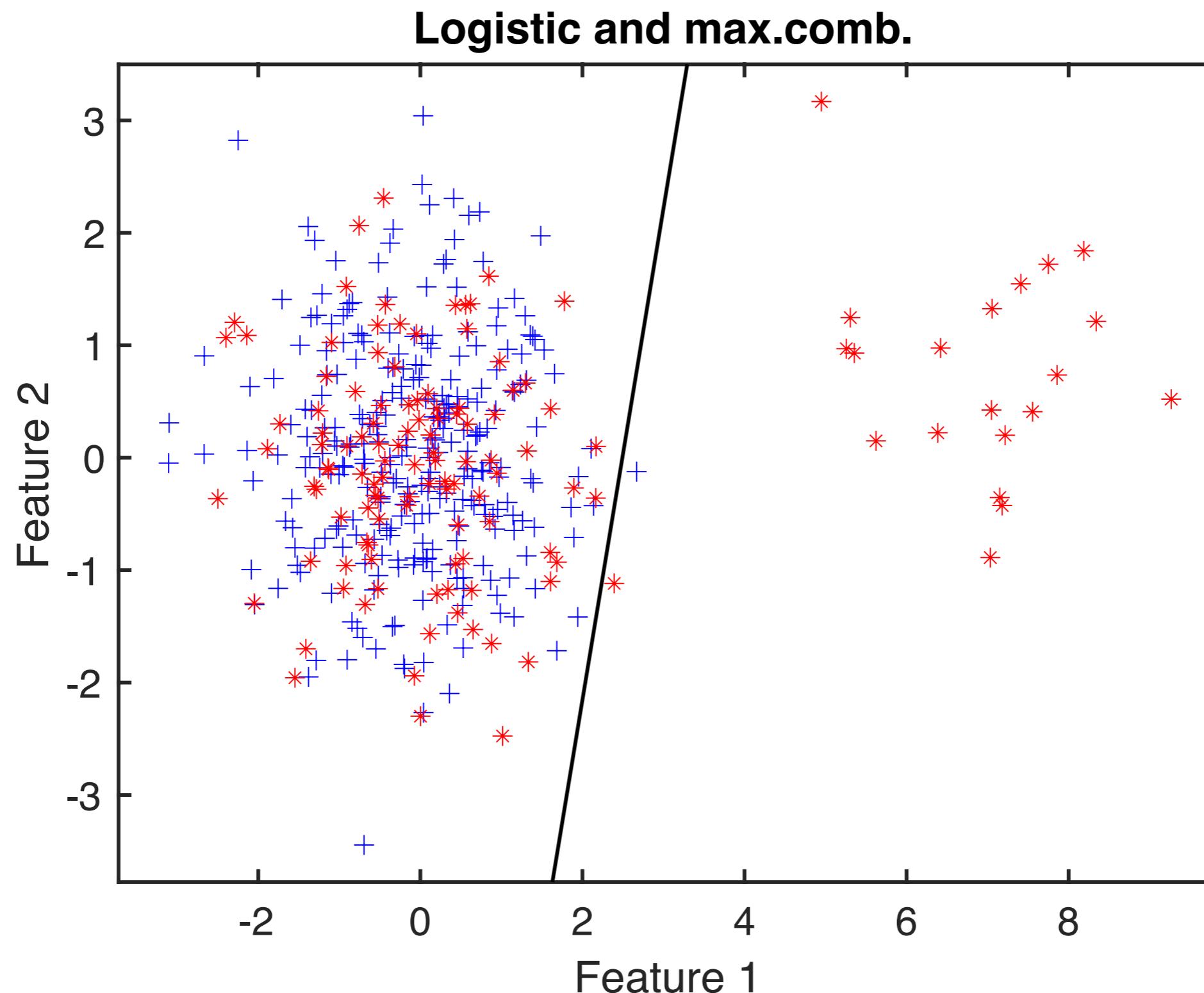
The 'naive' approach

- Many people do multiple-instance learning, but do not realize it
- Often the straightforward approach is followed:
 1. Copy the bag-labels to the instances $y_{ij} = y_i, \quad \forall j$
 2. Train a standard classifier $\hat{y}_{ij} = h(\mathbf{x}_{ij})$
 3. Combine the classifier outcomes on the instance using a combining rule $\hat{y}_i = C(h(\mathbf{x}_{i1}), \dots, h(\mathbf{x}_{in_i}))$
- Often the max- or mean-combining rule is used
- Suffers from heavily overlapping classes



Simple combination of logistic cl.

- Max-combining of logistic:



Diverse density

- A probabilistic approach to MIL
- Idea:
if none of the instances are from the concept: negative bag

if one, **or more**, instances are from concept: positive bag



Diverse density

- For positive bags, we want to optimize

$$Pr(+|B) = 1 - Pr(-|B)$$

and for negative bags

$$Pr(-|B)$$

- For negative bags, **all** instances should have high prob. of negative class:

$$Pr(-|B_i) = \prod_{j \in i} Pr(-|\mathbf{x}_{ij})$$

Noisy-OR
combination

- The negative class posterior is modeled:

$$Pr(-|\mathbf{x}_{ij}) = 1 - Pr(+|\mathbf{x}_{ij}) = 1 - \exp\left(-\frac{(\mathbf{x}_{ij} - \mathbf{t})^2}{s^2}\right)$$

- For positive bags:

$$Pr(+|B_i) = 1 - \prod_{j \in i} \left(1 - \exp\left(-\frac{(\mathbf{x}_{ij} - \mathbf{t})^2}{s^2}\right)\right)$$



Diverse Density

- How to optimize this? Maximize the likelihood:

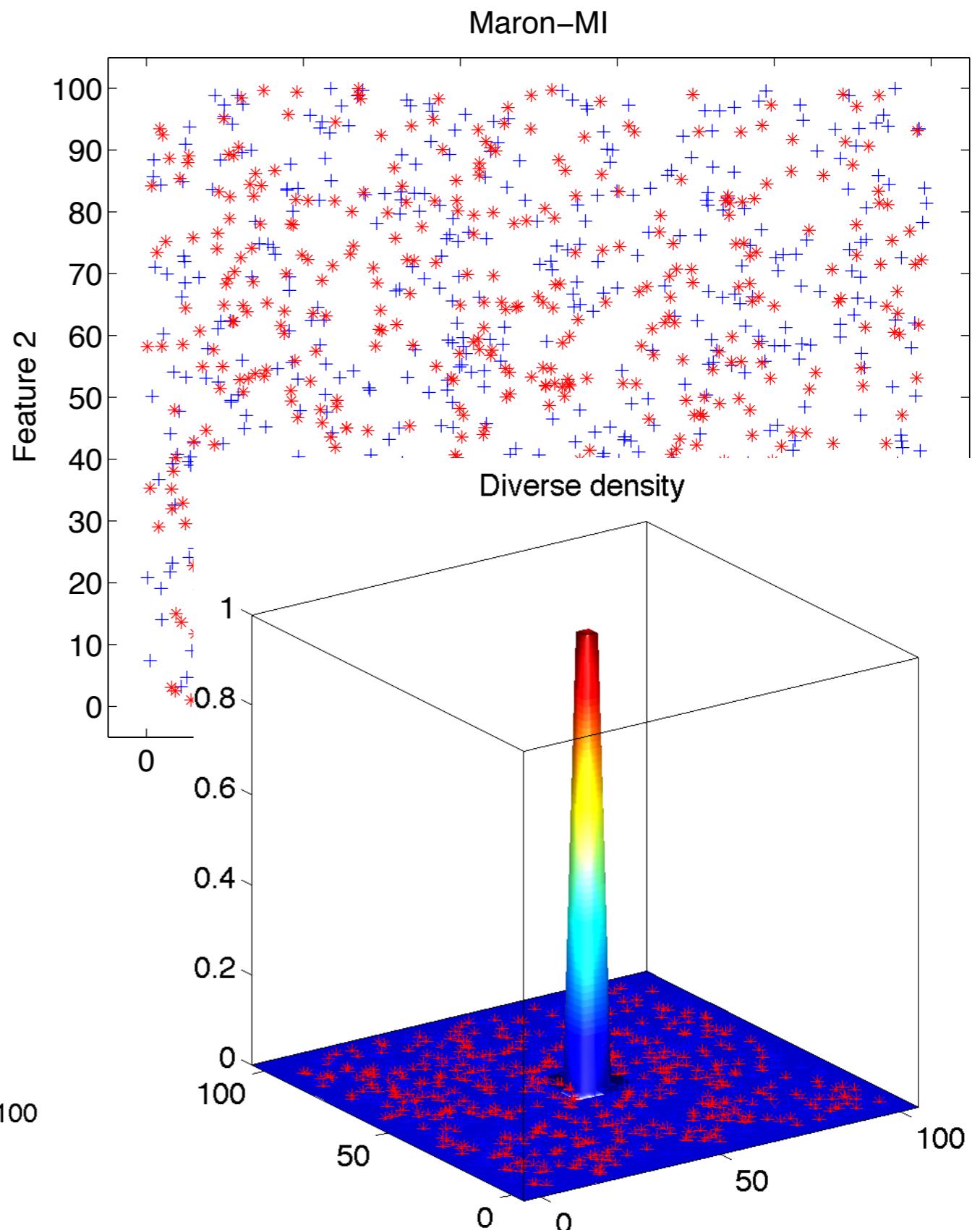
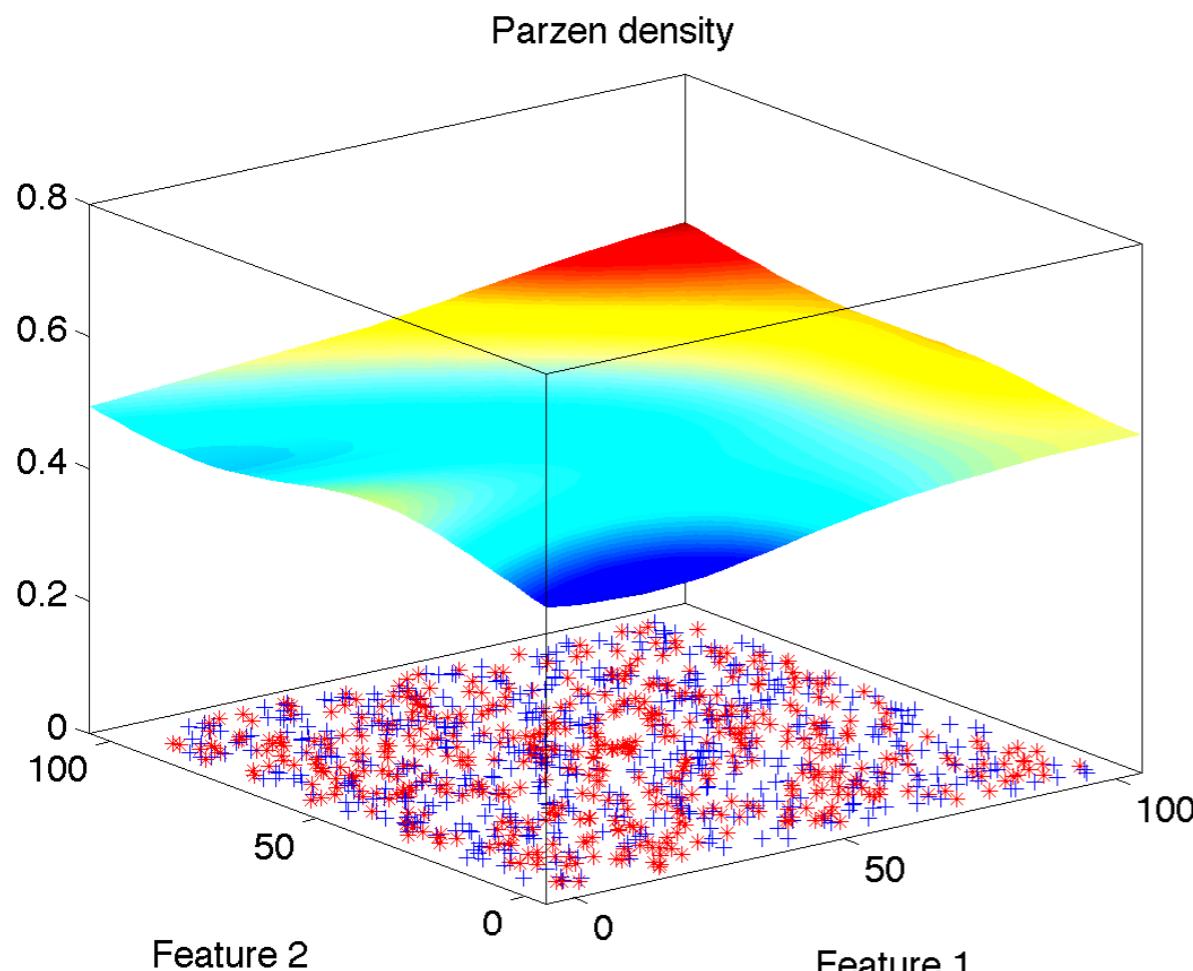
$$L = \prod_{i \in +} Pr(+) | B_i) \prod_{i \in -} Pr(- | B_i)$$

- We need to optimize the location of the concept t , and the scale per feature s
- It is hard to optimize. A gradient ascent procedure is proposed. L has many local optima, so multiple initializations are necessary to find a good one
- In practice this is **very** slow, but it works



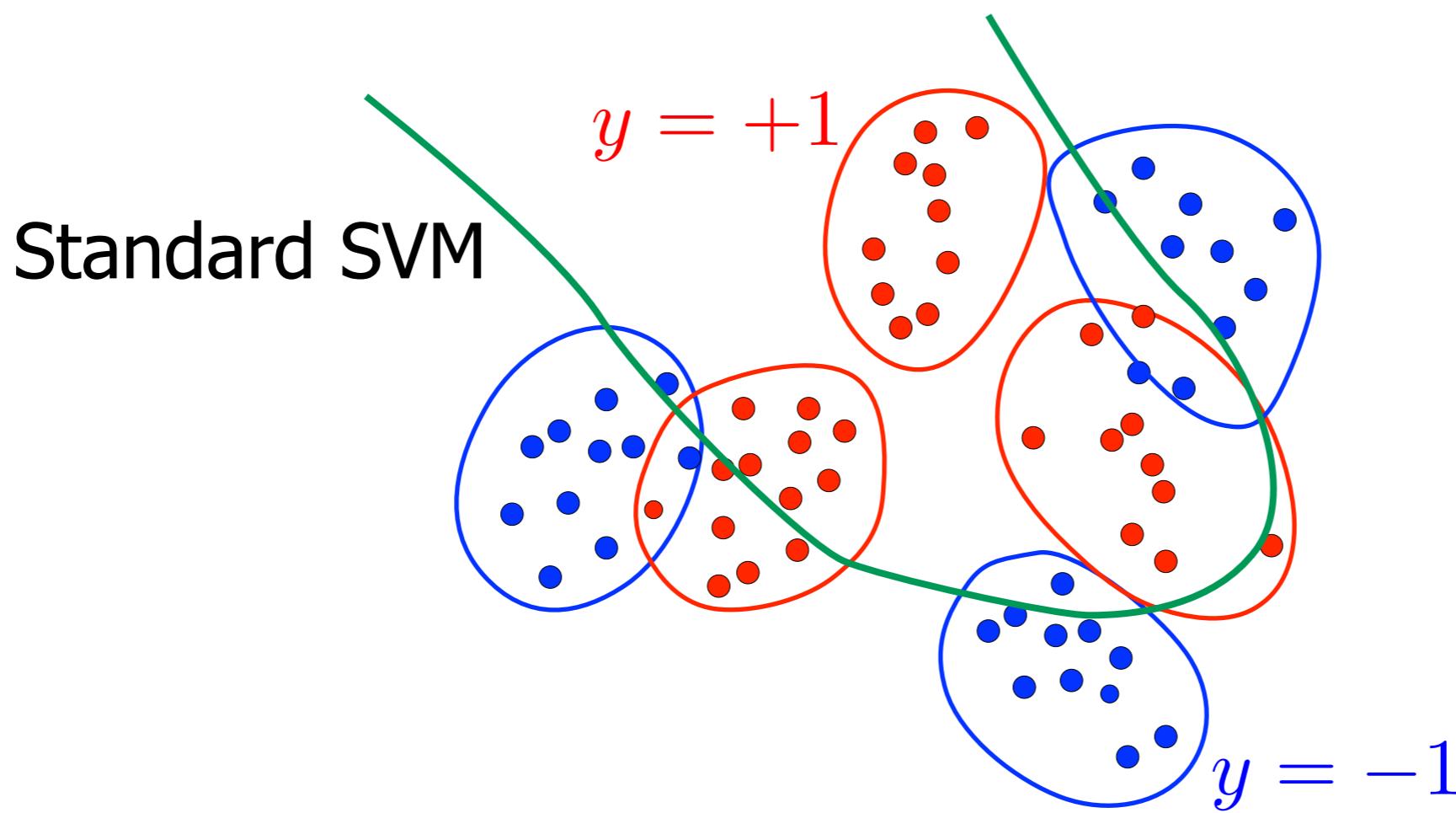
Artificial example

- Generate 10 positive and 10 negative bags
- The positive bags have at least one instance in a small square in the middle (the concept)



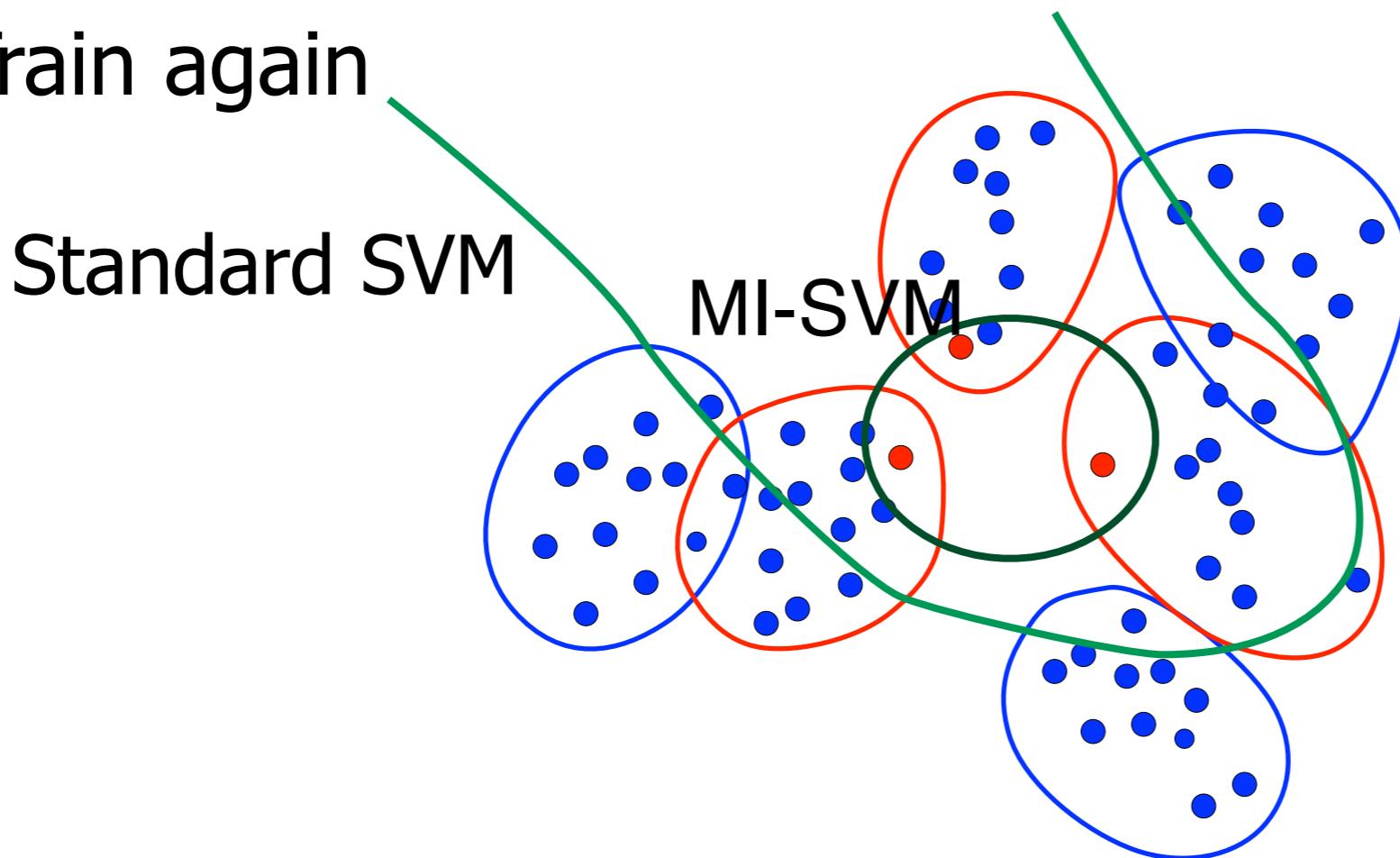
MI-SVM

- Extension of a standard Support Vector machine
- In the first iteration, copy the bag label to the instance label
- Train a standard SVM



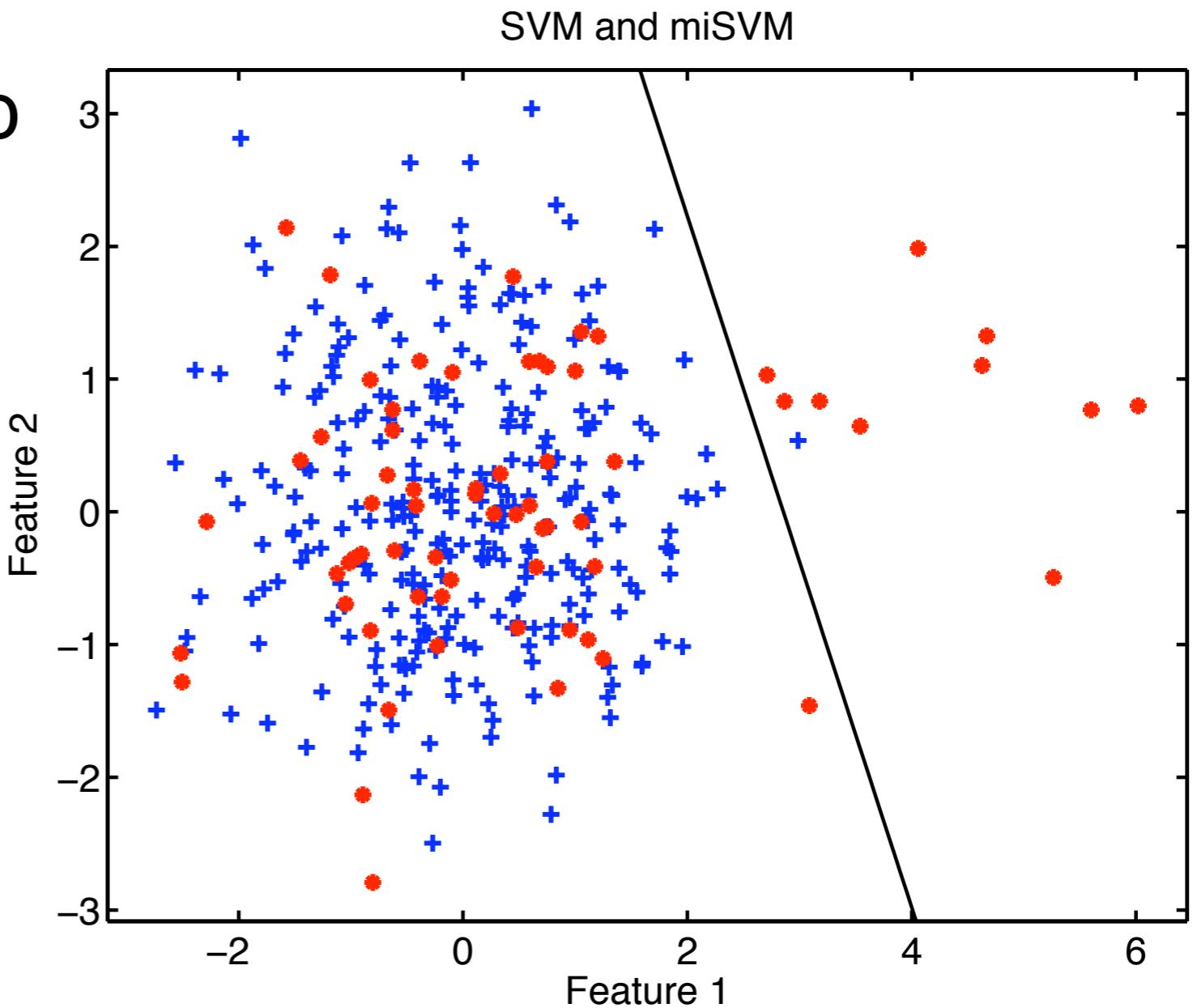
MI-SVM

- Extension of a standard Support Vector machine
- After the first iteration, allow that just a **single** instance in a positive bag is on the correct side of the decision boundary
- Train again



On artificial data

- The miSVM manages to detect the 'concept'



- By the overwhelming presence of the negative class, the standard SVM classifies everything as negative



Generalising MI-SVM?

- Why would you only use an SVM here?
- Can we generalise this?
- How...?



MIL-Boost

- Assume we have a model

$$\mathbf{h}(x) = \sum_{t=1}^T \alpha_t h_t(x)$$

- Assume we have a performance

where

$$\mathcal{L} = \prod_{i \in +} p(+) | B_i \prod_{i \in -} p(-) | B_i$$

$$p(+) | B_i = \text{softmax}_i(\text{sigmoid}(\mathbf{h}(\mathbf{x}_{ij})))$$

- Softmax can be defined in different ways, for instance

$$\left(\frac{1}{m} \sum_{\ell} v_{\ell}^r \right)^{\frac{1}{r}}$$

- Then optimising the loss iteratively



MIL Boost

Input: Dataset $\{X_1, \dots, X_n\}, \{y_1, \dots, y_n\}, y_i \in \{-1, 1\}$

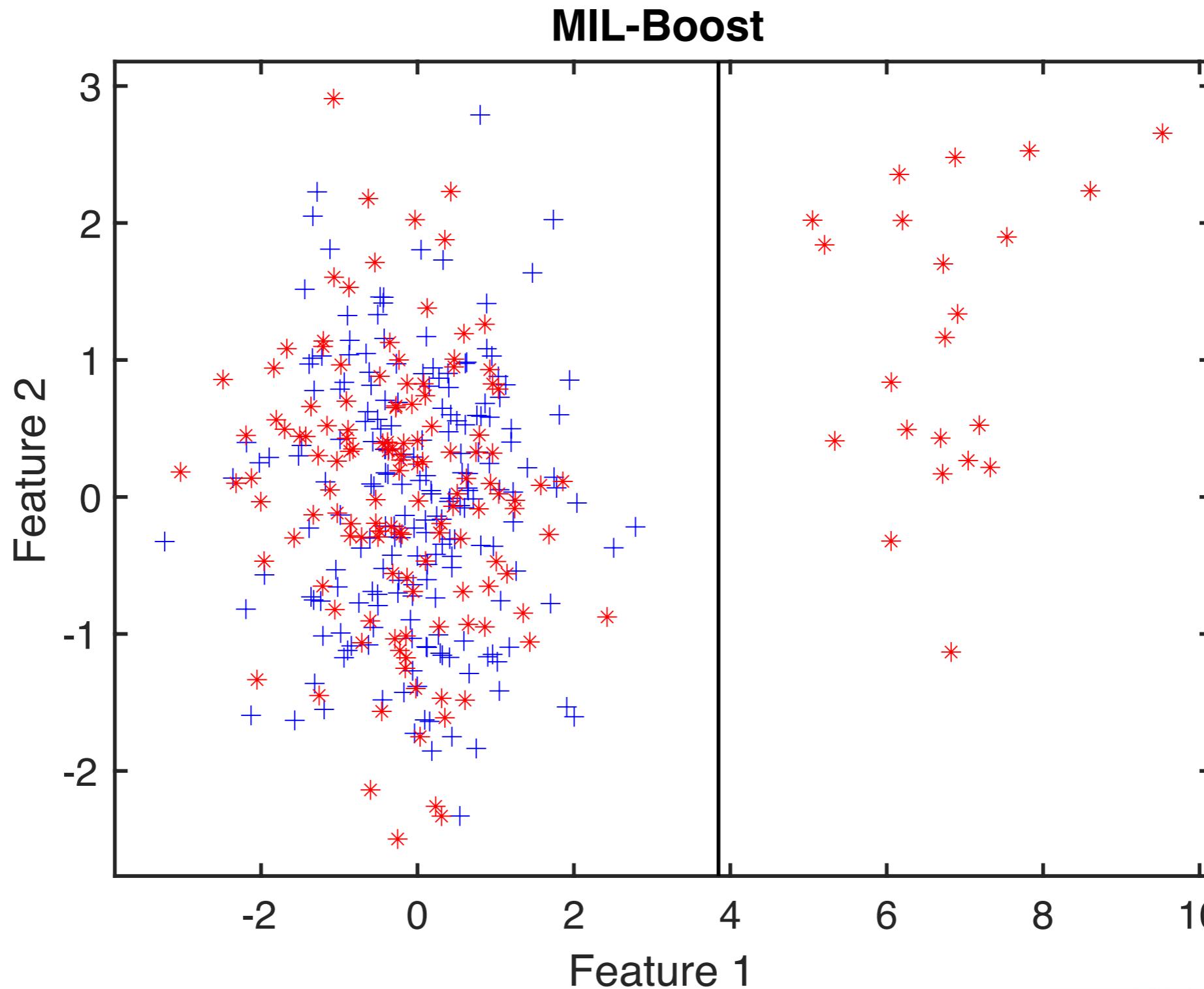
- 1: **for** $t = 1$ to T **do**
- 2: Compute weights $w_{ij} = -\frac{\partial \mathcal{L}}{\partial \mathbf{h}_{ij}}$
- 3: Train weak classifier h_t using weights $|w_{ij}|$
$$h_t = \operatorname{argmin}_h \sum_{ij} \mathbf{1}(h(x_{ij}) \neq y_i) |w_{ij}|$$
- 4: Find α_t via line search to minimize $\mathcal{L}(\mathbf{h})$
$$\alpha_t = \operatorname{argmin}_\alpha \mathcal{L}(\mathbf{h} + \alpha h_t)$$
- 5: Update strong classifier $\mathbf{h} \leftarrow \mathbf{h} + \alpha_t h_t.$
- 6: **end for**

- Here, the tough part is to derive $-\frac{\partial \mathcal{L}}{\partial h(x_{ij})}$
- Chain-rule after chain rule...



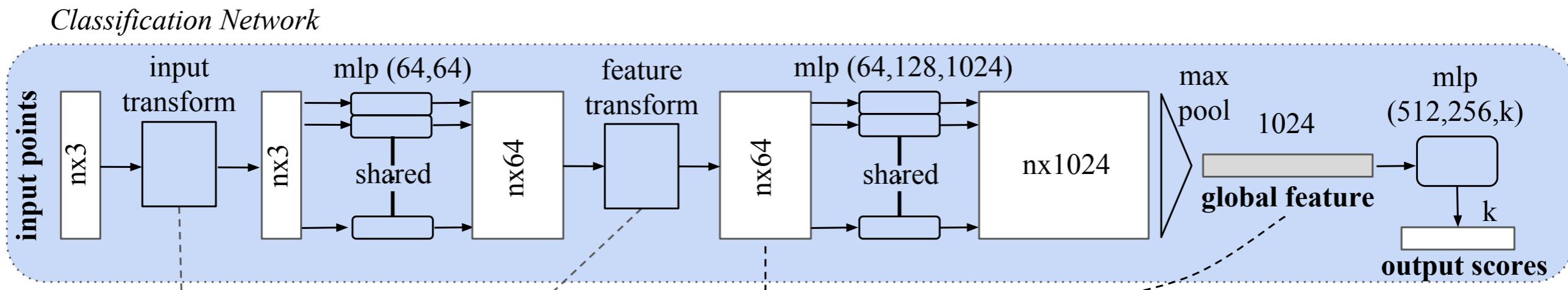
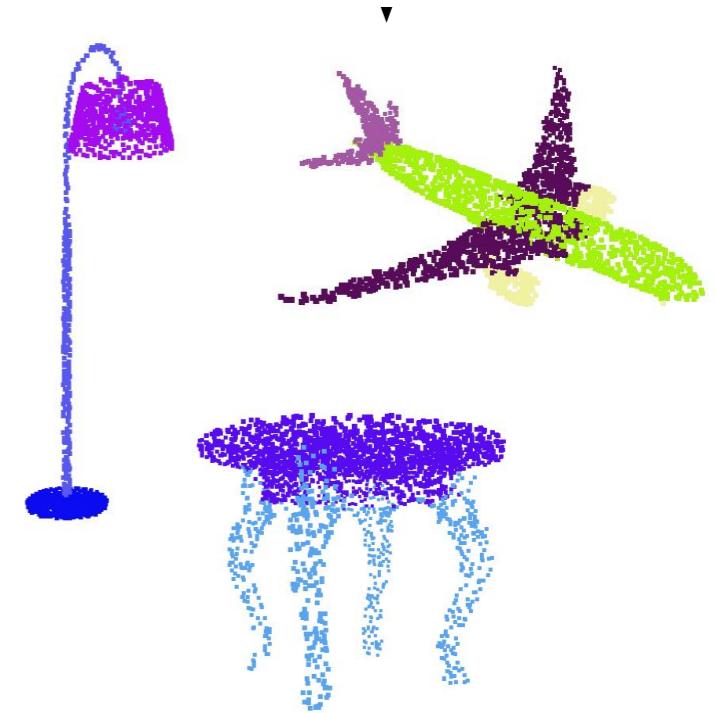
On artificial data

- Weak classifier is a decision stump:



PointNet

- Deep learning approach to classify/segment point clouds
- Each 3D point is an instance
- The n instances are combined using a max-pooling/combining
- Does **one** instance determine the bag label?



Conclusion first approaches MIL

- General idea: detect concept
- Diverse Density: probabilistic implementation, works, but (very) slow
- MI-SVM, naive approach: heuristic approach, very fast, often works reasonably well
- MIL-Boost: more flexible, also using the noisy-OR idea of Diverse Density
- Point-net: a deep learning approach for point clouds

- Concept detection allows for interpretation of the classification
- Competing idea: general bag classification, avoid concept detection



Bag representation

- Bag approaches:
make new representations that capture all bag info

- Extract features from a bag of instances:

$$\mathbf{x}(B) = (\text{mean}(B), \min(B), \max(B), \text{cov}(B), \dots)$$

- Define a (dis-)similarity measure between bags:

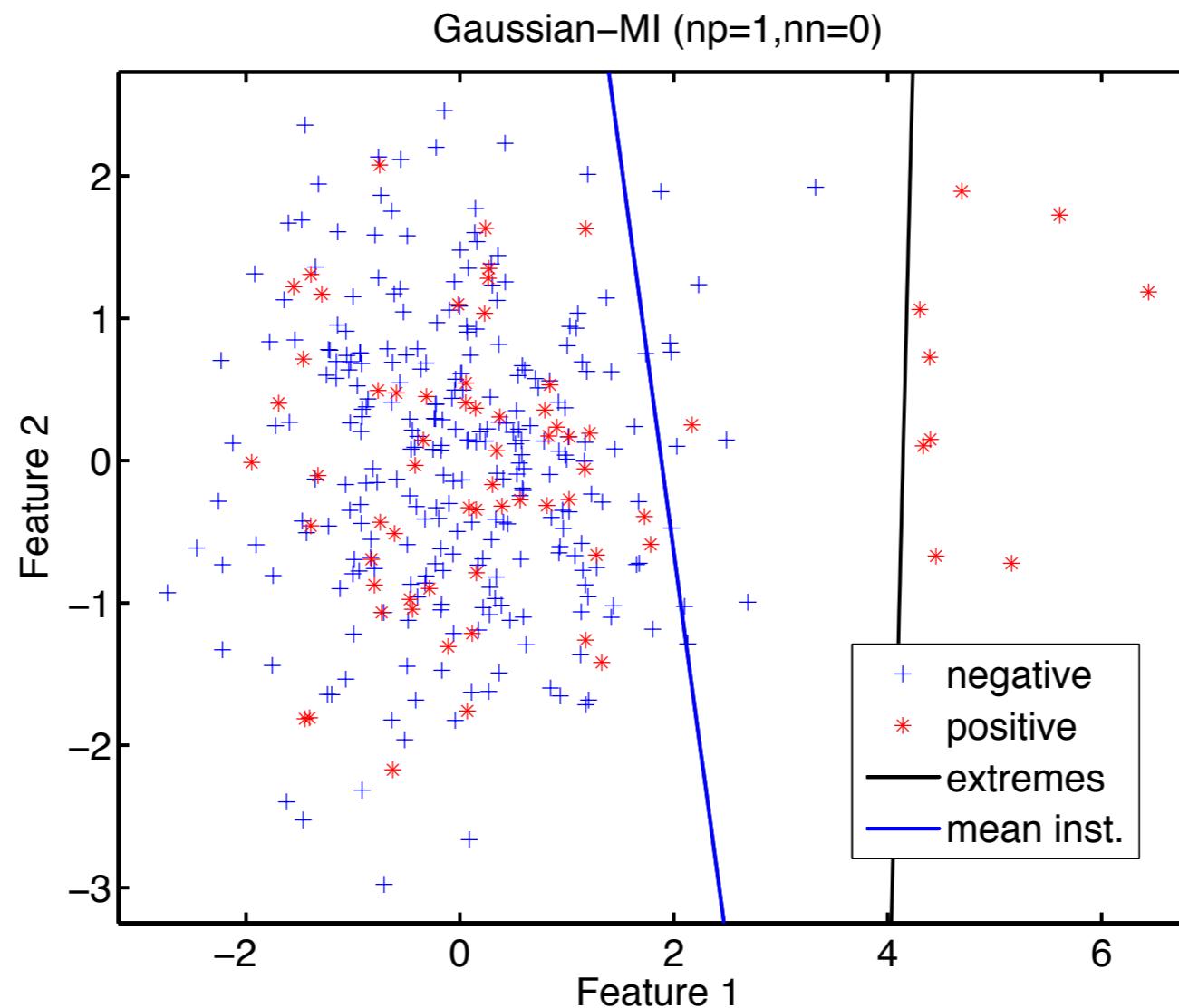
$$K(B_i, B_j) = (m(B_i) - m(B_j))^T \Sigma^{-1} (m(B_i) - m(B_j))$$
$$m(B_i) = \text{mean}(B_i)$$

- A Bag-of-Words approach: ...

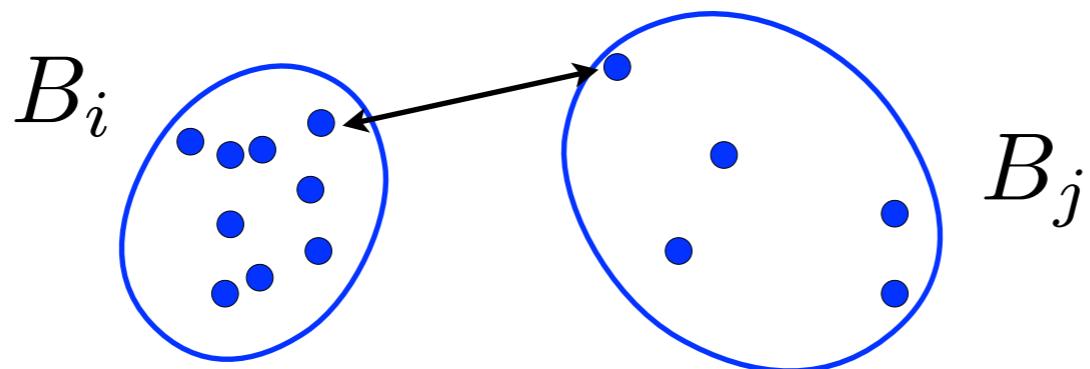


Simple feature representation

- Use the mean-instance, or the min.and max. values per bag (or any statistic), and train an SVM
- Not perfect, but very fast and simple:



Bag dissimilarities



- Define a distance between bags:

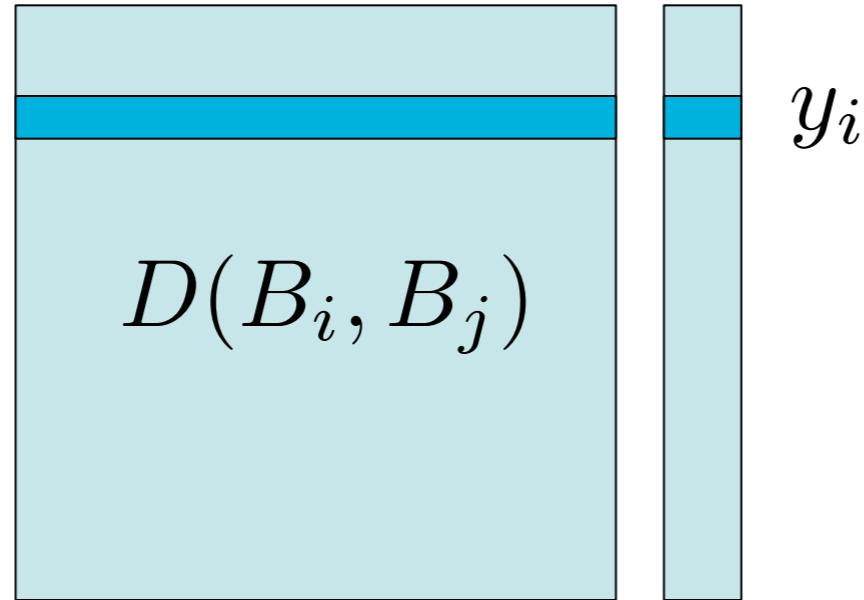
$$D(B_i, B_j) = \min_{k,l} \|\mathbf{x}_{ik} - \mathbf{x}_{jl}\|^2$$

- or

$$D(B_i, B_j) = \max \left(\min_l \|\mathbf{x}_i - \mathbf{x}_{jl}\|^2, \min_l \|\mathbf{x}_j - \mathbf{x}_{il}\|^2 \right)$$

- or ...

Bag dissimilarities



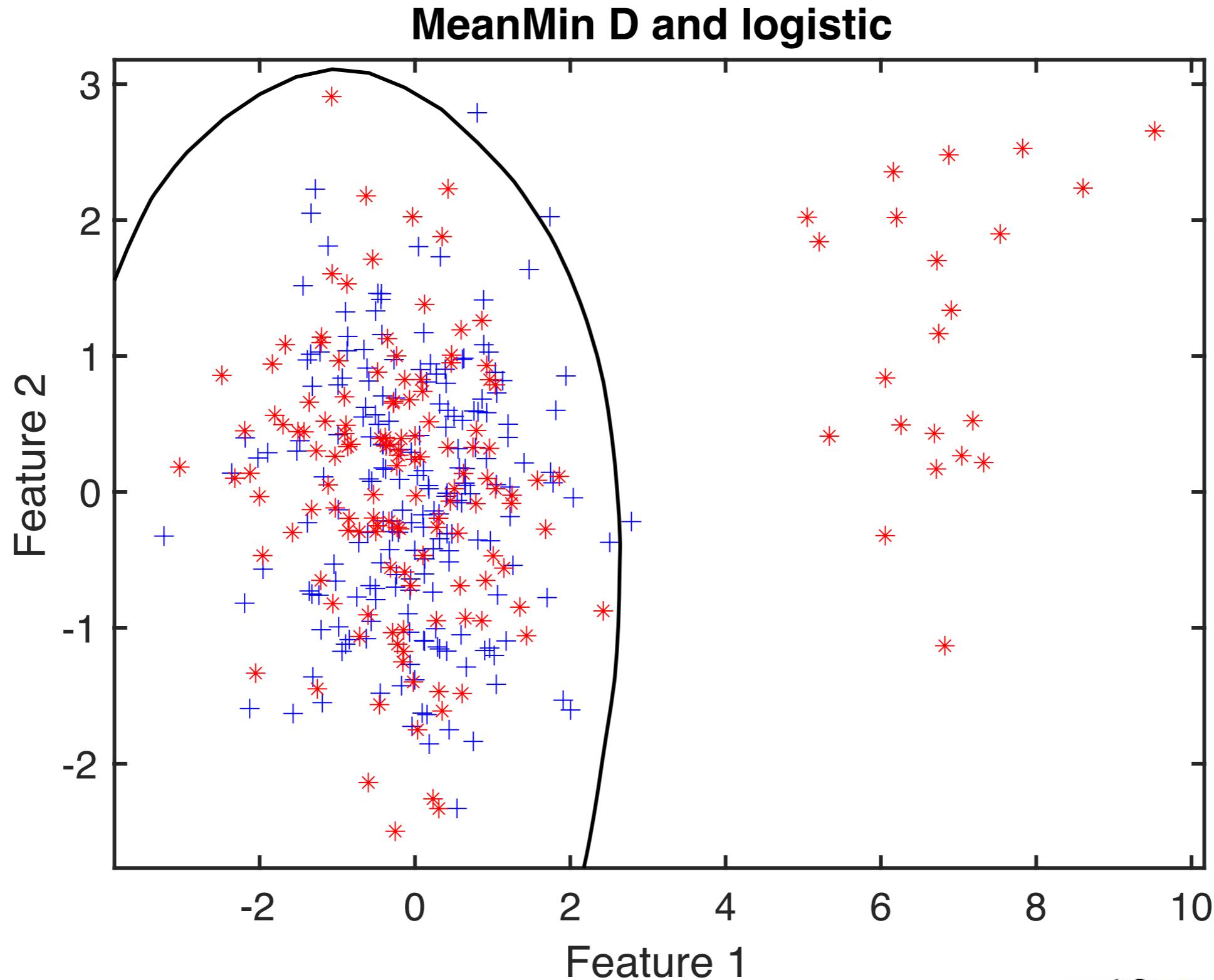
- Given a bag-distance matrix, train any classifier you want!

$$\hat{y}_i = f([D(B_i, B_1), \dots, D(B_i, B_N)])$$

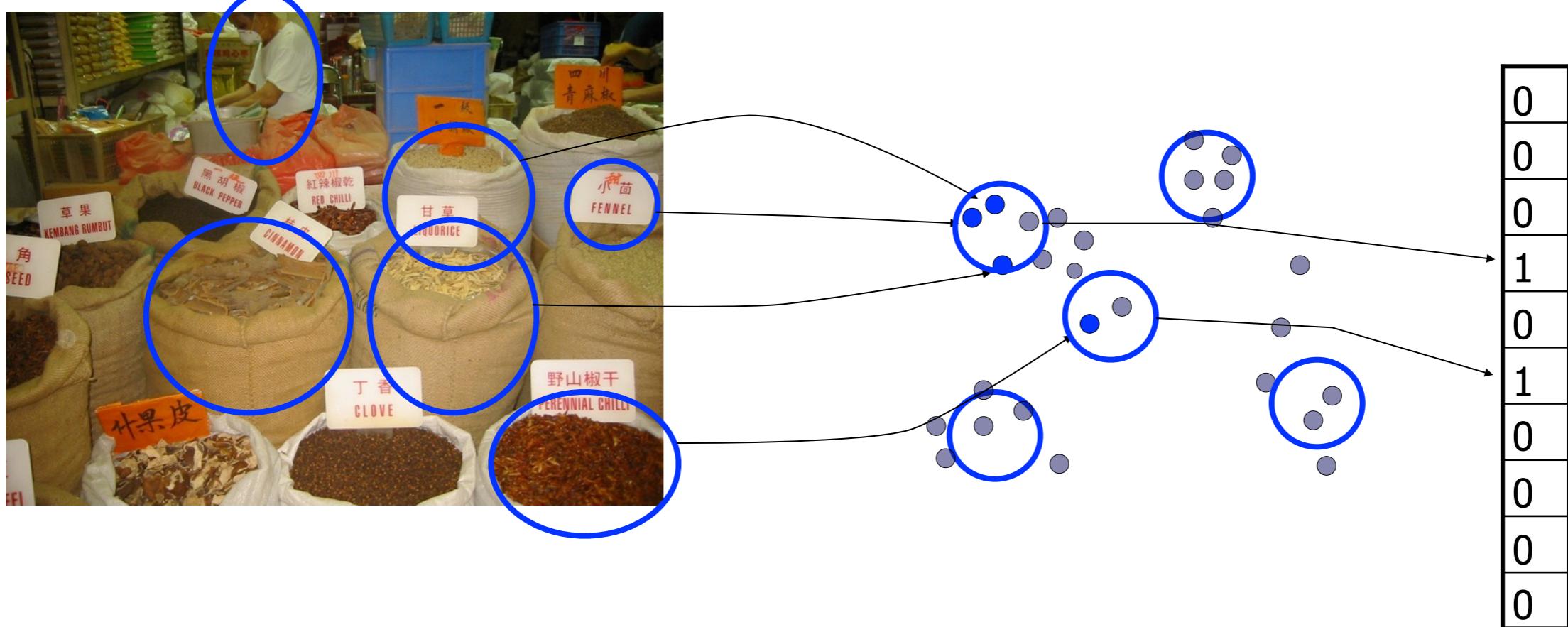
- For evaluating a new bag, you need to compute the distances to the original prototype



On artificial data



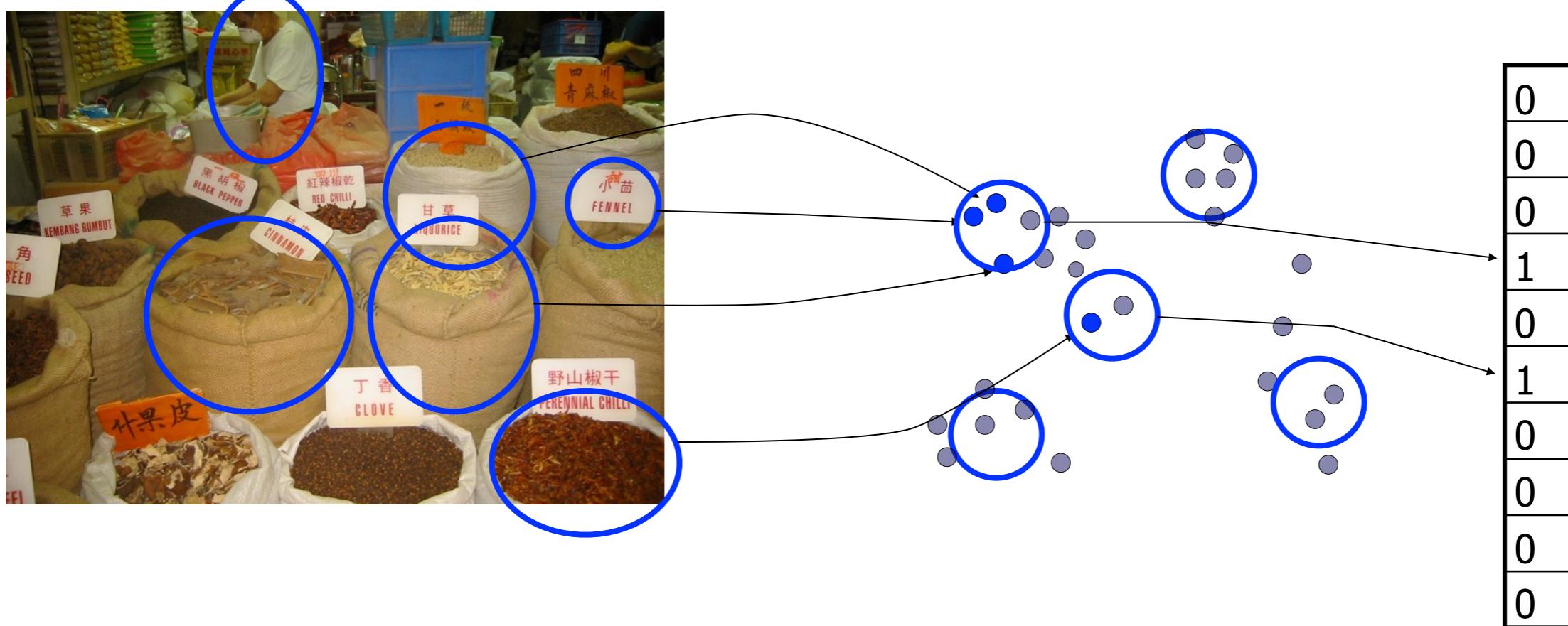
Bag-of-... representation



- A dictionary is constructed (often by clustering of all instances)
- The presence of a (visual) word is counted
- A bag is represented by a histogram

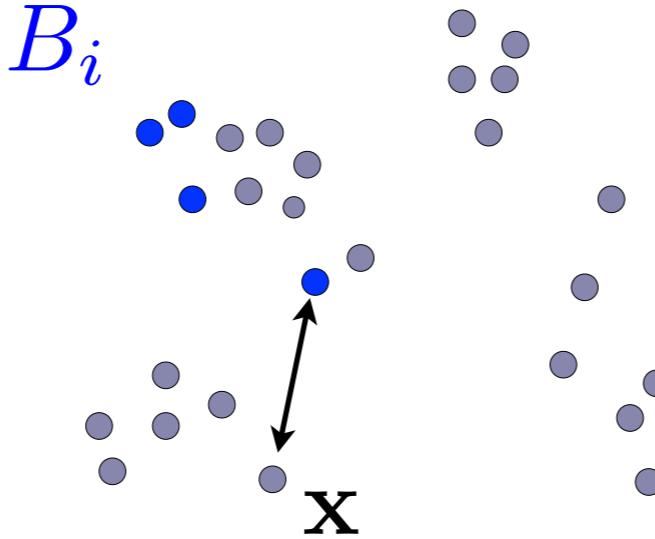


Bag-of-... representation



- The histogram often has many bins: high-dimensional feature vector!
- Typically, a support vector classifier is trained
- The clustering typically does not involve labels





- Instead of distance to cluster centers
 - use ALL instances as clusters
 - use a similarity to the instance

$$s(B_i, \mathbf{x}) = \max_j \exp\left(-\frac{\|\mathbf{x}_{ij} - \mathbf{x}\|^2}{\sigma^2}\right)$$

$$\mathbf{m}(B_i) = [s(B_i, \mathbf{x}_1), \dots, s(B_i, \mathbf{x}_N)]$$

Classifiers on the bag representation

- Note that bag representations typically are large (high dimensional)
- Need for robust, sparse, classifiers
- Typically used is Liknon, or L₁-support vector classifier:

$$\min |\mathbf{w}|_1 + C \sum_{i=1}^N \xi_i$$

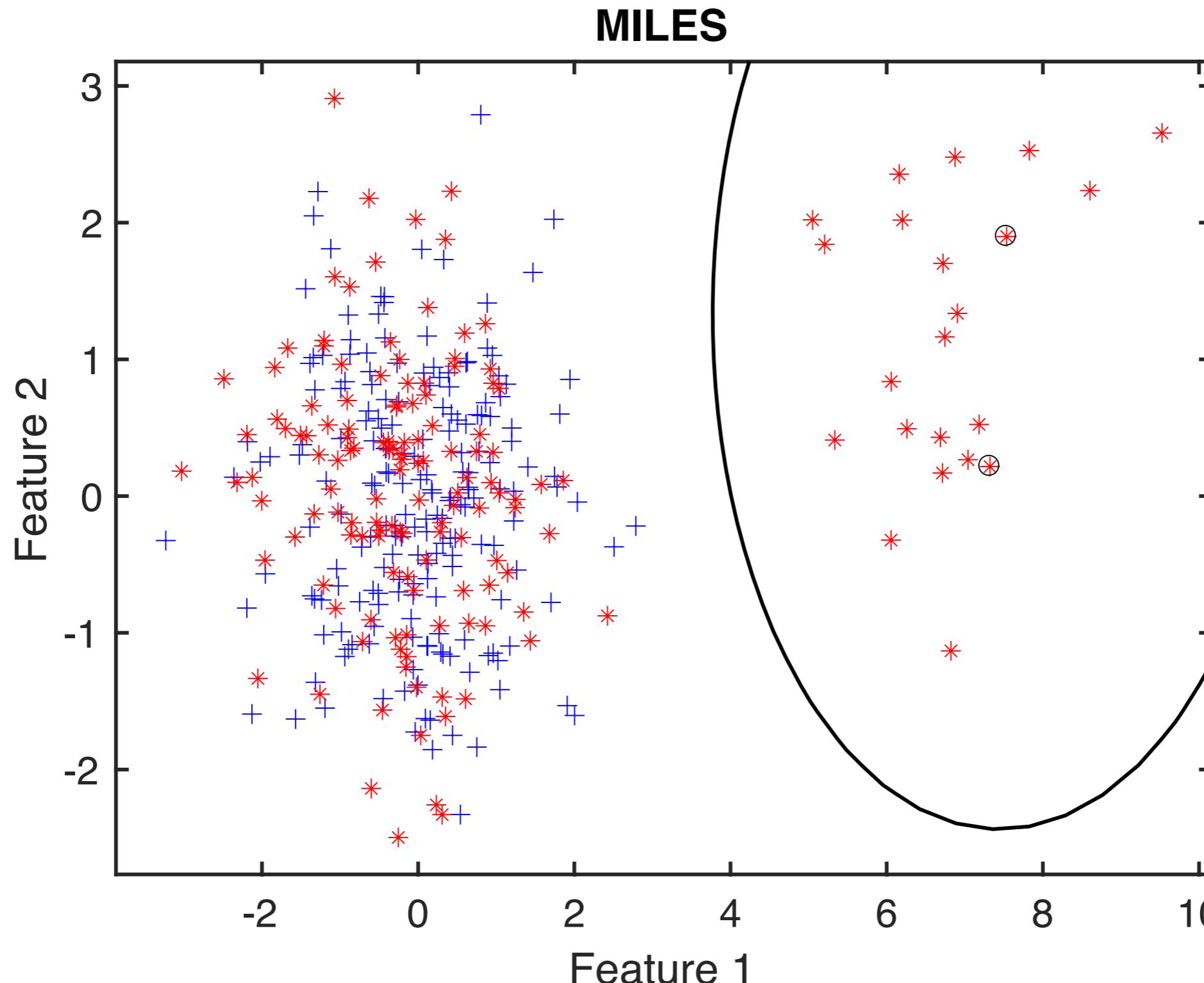
such that

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i$$

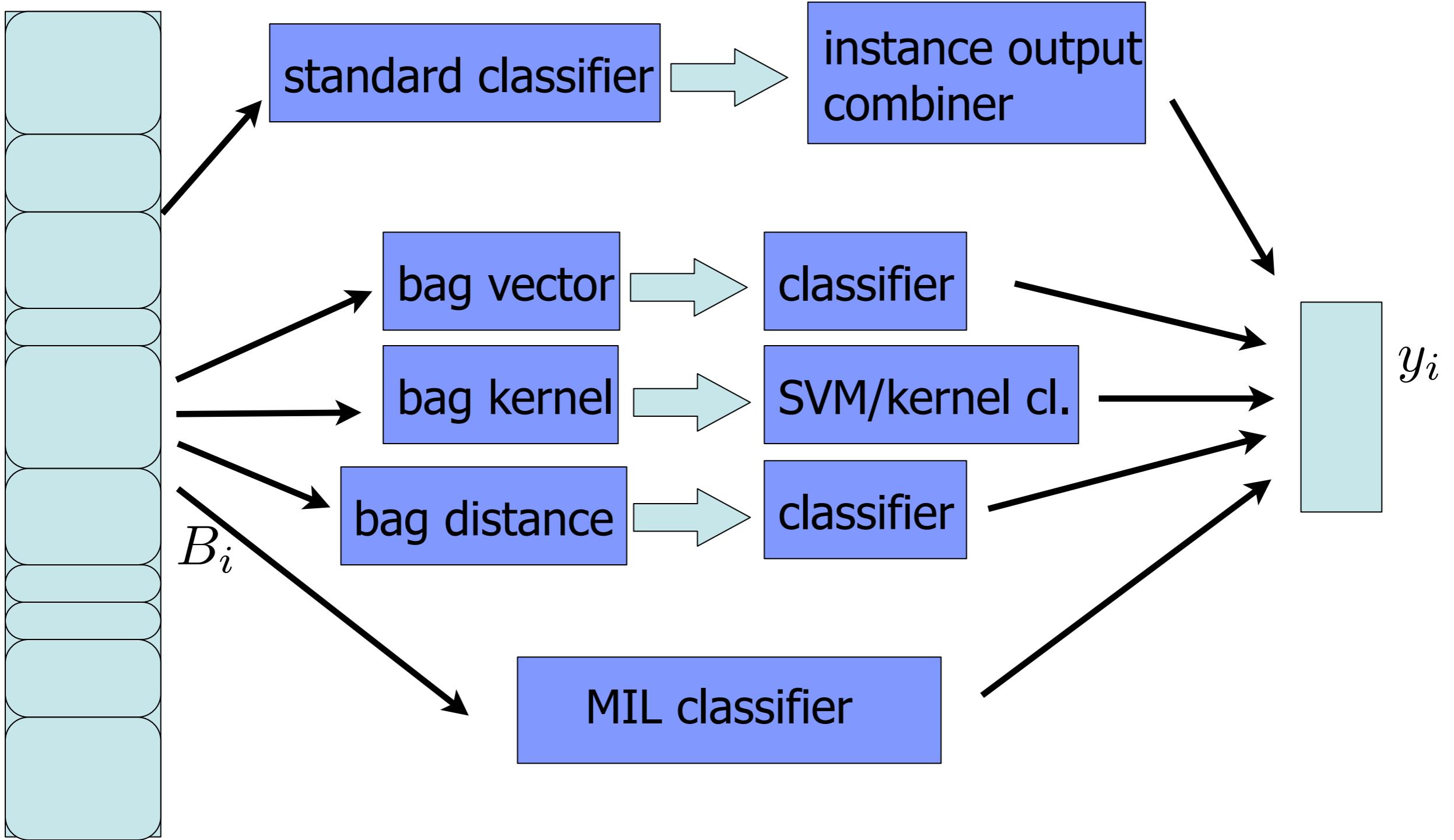


On artificial data

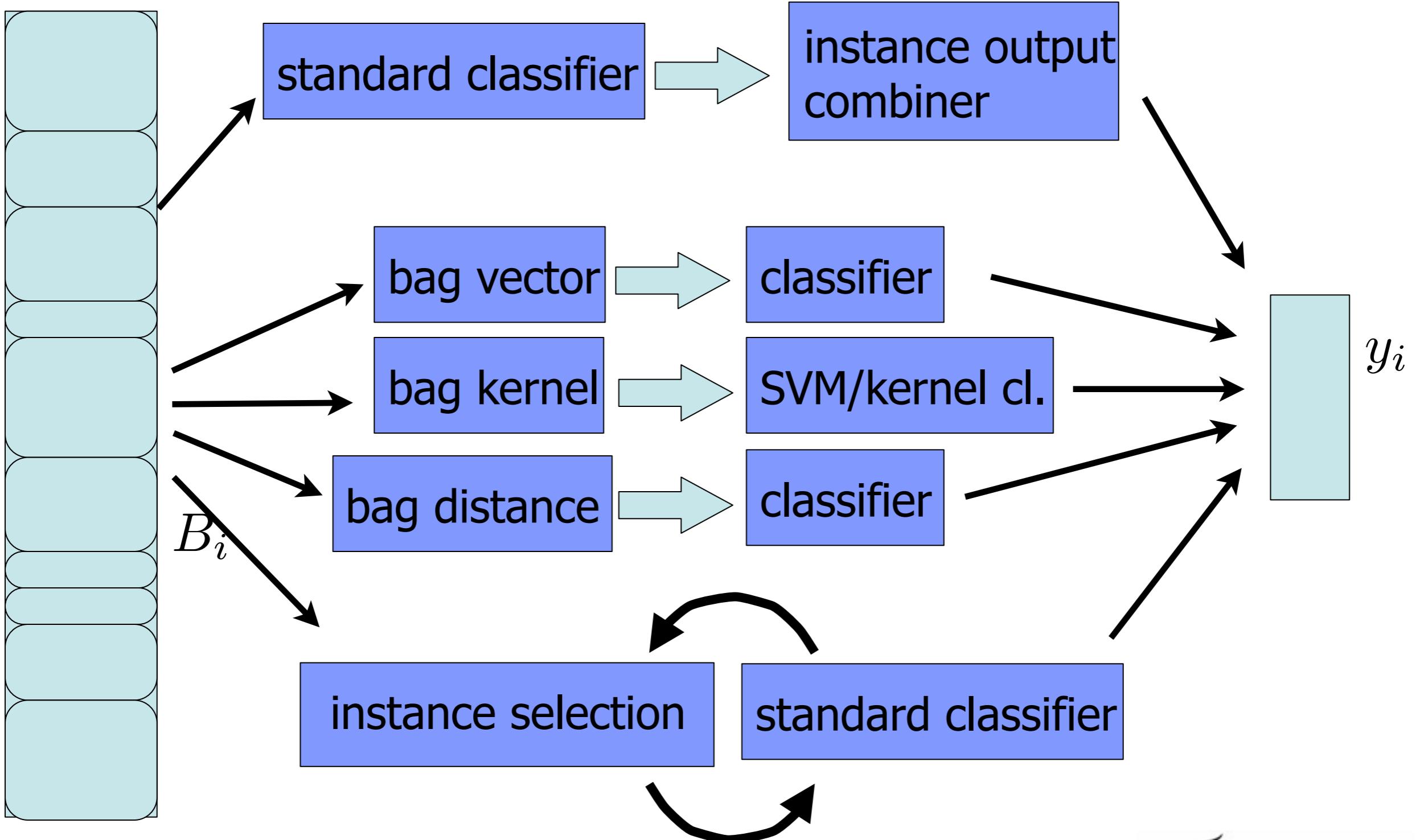
- Only two prototype instances selected:



Overview approaches



Overview approaches



Many other approaches

- Rule-based classifiers (RELIC, FOIL, ...)
- Expectation-Maximization Diverse Density (extension of the Diverse Density)
- Citation-kNN, a variant of the k-nearest neighbor classifier
- (probably someone already proposed Deep MIL classifiers, or convolutional MIL neural networks...)
- ...



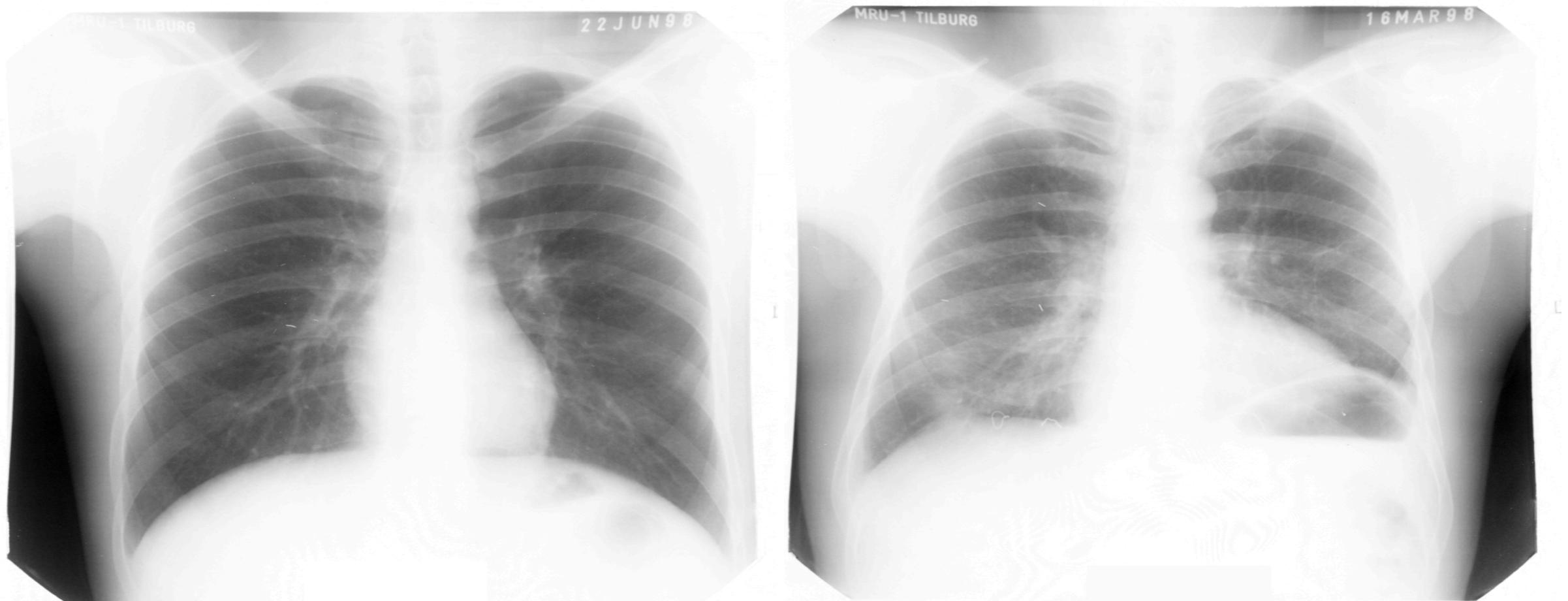
Some results on MUSK

- See <http://homepage.tudelft.nl/n9d04/milweb/> :-)



ILD/Tuberculosis detection

- Try to predict if a patient has a lung disease



healthy	ill	instances	features
241	147	$\{1.7, 1.0\} \times 10^6$	158



Clean-speech detection



example overlapping speech



new speech: overlap?

- Find the 'clean' speech (where a single person is speaking)
- The detection of multiple speakers should not depend on the speakers themselves (it should generalize over different movies)



Presence of Action Units



From: Timing is everything, A spatio-temporal approach to the analysis of facial actions, M.F. Valstar

- 244 expression sequences (with around 85 frames per sequence)
- Can we detect the presence of the activity of a certain Action Unit?
- Can we detect the phase of the activity of the Action Unit?



Open (fundamental) questions...

- What is the sample size of a MIL problem?
(what is the influence of the #instances in a bag, the fraction of 'truly' positive instances?)
- How to characterise MI problem (bags/dim)?
- What is the complexity of a MI problem? How to vary the complexity of a MI classifier?
- How to efficiently train a MI classifier? Is the idea of a 'concept' fruitful?
- How to include more structure in a MI classifier?
(use more than one concept, use temporal structure, spatial structure, ...)
- How to incorporate (partial) instance labels?
- How to optimise the features used in MIL? Deep learning!?
- Multi-class MIL?



Conclusions

- For many real-world classification problems complex/compound objects have to be represented
- One possibility: represent the object by a **set** of feature vectors
- Many procedures have been proposed, not very clear when one will work well:
 - Diverse-Density is very good, but very slow
 - Straightforward methods sometimes surprisingly good
 - Represent bags with feature vectors is good
 - My personal recommendation:
MILES



Questions?

- ...?

