Clustering

Machine Learning Roadmap

supervised / unsupervised	continuous (predicting a quantity)	discrete (predicting a category)
unsupervised	**Dimension Reduction**	**Clustering**
supervised	**Regression**	**Classification**

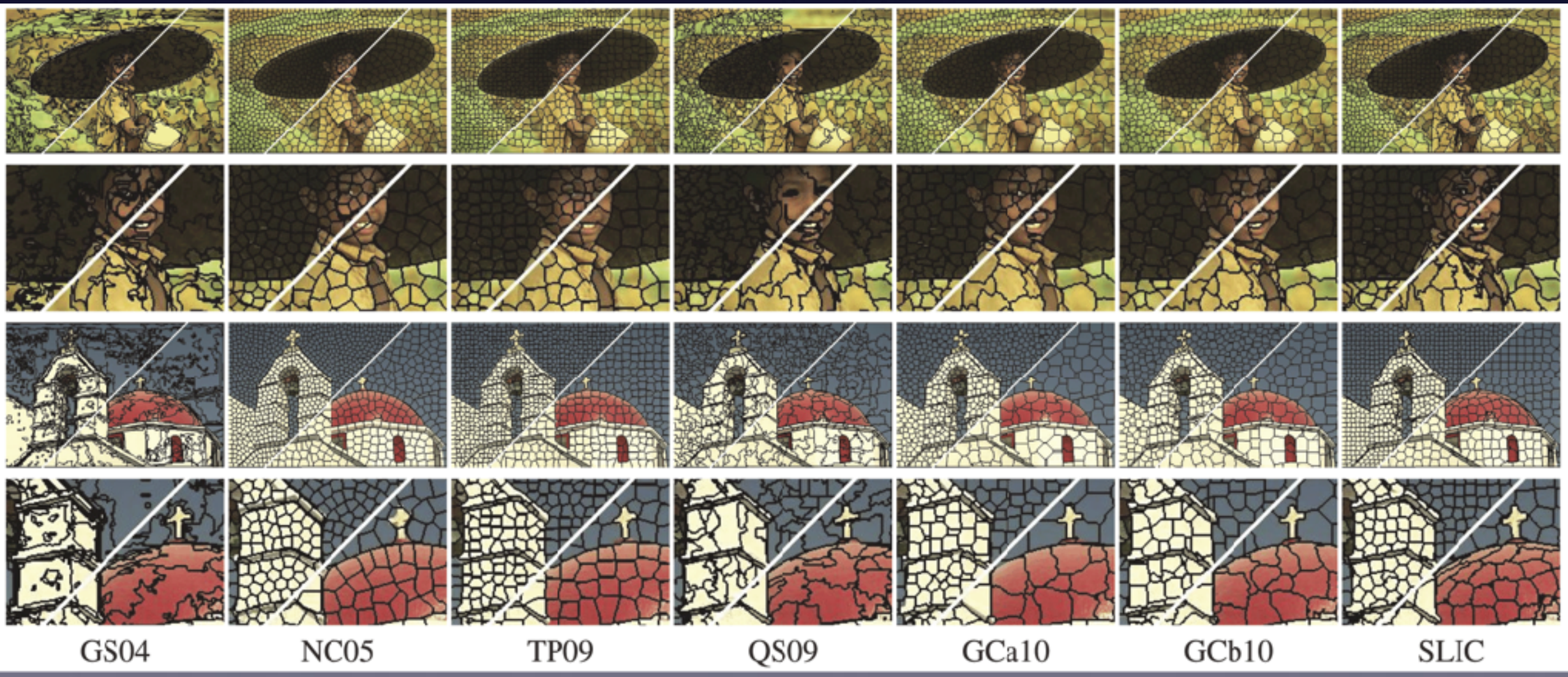
Clustering

- Group together similar points and represent them with a single token.

- Issues:
  - How do we define two points/images/patches being "similar"?
  - How do we compute an overall grouping from pairwise similarity?

Clustering Example

- Grouping pixels of similar appearance and spatial proximity together; there's so many ways to do it, yet none are perfect.



Clustering Example

Why do we cluster?

- **Summarizing Data**
  - Look at large amounts of data
  - Patch-based compression or denoising
  - Represent a large continuous vector with the cluster number
- **Counting**
  - Histograms of texture, color, SIFT vectors
- **Segmentation**
  - Separate the image into different regions
- **Prediction**
  - Images in the same cluster may have the same labels

How do we cluster?

- **K-means**
  - Iteratively re-assign points to the nearest cluster center
- **Gaussian Mixture Model (GMM) Clustering**
- **Mean-shift clustering**
  - Estimate modes of pdf
- **Hierarchical clustering**
  - Start with each point as its own cluster and iteratively merge the closest clusters
- **Spectral clustering**
  - Split the nodes in a graph based on assigned links with similarity weights

Clustering for Summarization

- Goal: cluster to minimize variance in data given clusters while preserving information.

$$\boldsymbol{c}^*, \boldsymbol{\delta}^* = \underset{\boldsymbol{c},\boldsymbol{\delta}}{\operatorname{argmin}} \frac{1}{N} \sum_{j=0}^{N} \sum_{i=0}^{K} \delta_{i,j} (c_i - x_j)^2$$

data

cluster center

Whether x_j is assigned to c_i

How do we measure similarity?

- Euclidean Distance:

$$\text{distance}(\text{x}, \text{y}) = \sqrt{(y_1 - x_1)^2 + (y_2 - x2)^2 + \cdots + (y_n - x_n)^2}$$

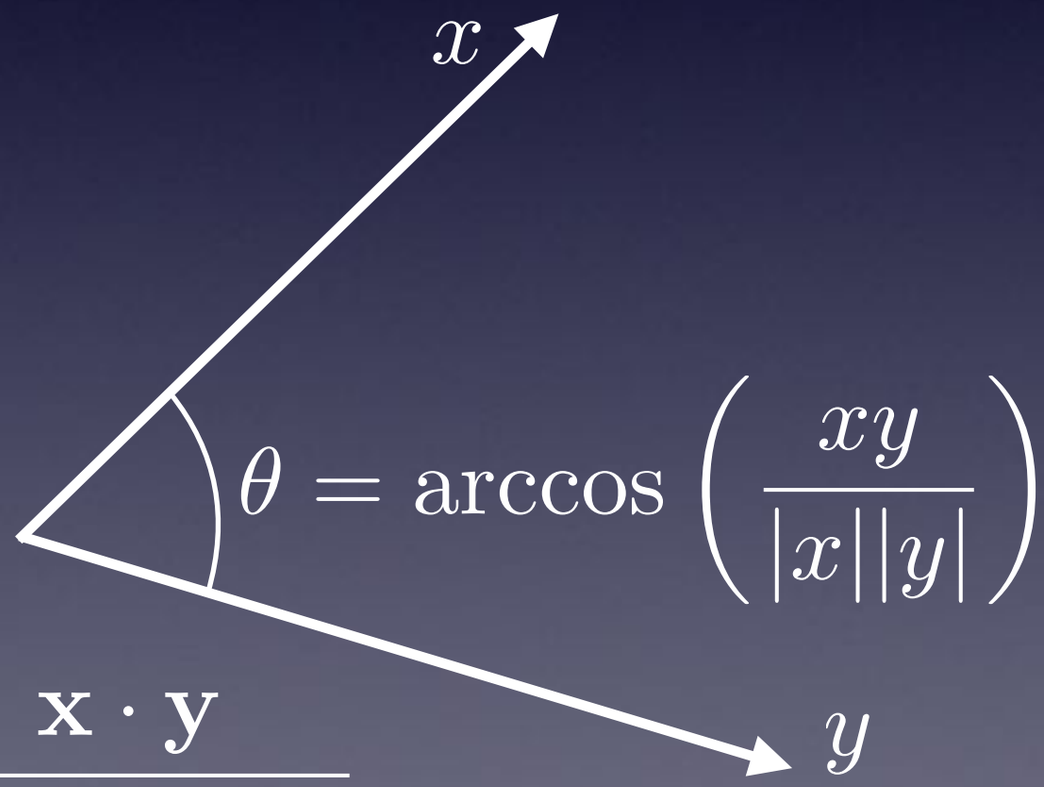
$$= \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

$$||\mathbf{y} - \mathbf{x}|| = \sqrt{(\mathbf{y} - \mathbf{x}) \cdot (\mathbf{y} - \mathbf{x})}$$

- Cosine similarity:

$$\mathbf{x} \cdot \mathbf{y} = ||\mathbf{x}||_2 \, ||\mathbf{y}||_2 \cos\theta$$

$$\text{similarity}(\mathbf{x}, \mathbf{y}) = \cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||_2 \, ||\mathbf{y}||_2}$$

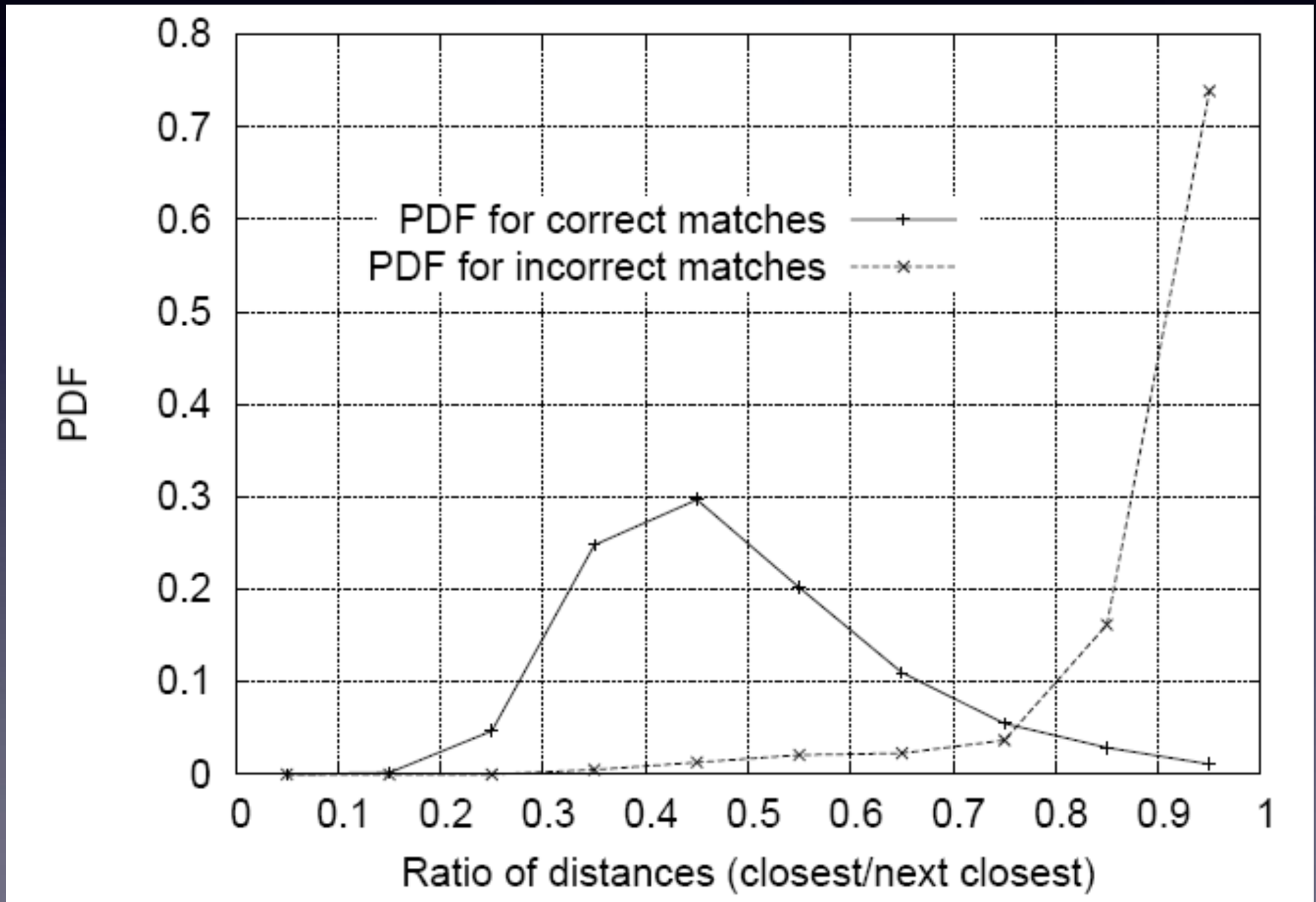


Nearest Neighbor Distance Ratio

- Compare distance of closest (NN1) and second closest (NN2) feature vector neighbor.
- If NN1≈NN2, ratio NN1/NN2 will be ≈1 → matches too close.
- As NN1 << NN2, ratio NN1/NN2 tends to 0.
- Sorting by this ratio puts matches in order of confidence.

Nearest Neighbor Distance Ratio

- How to threshold the nearest neighbor ratio?



Lowe IJCV 2004 on 40,000 points.

Threshold depends on data and specific applications

k-means clustering

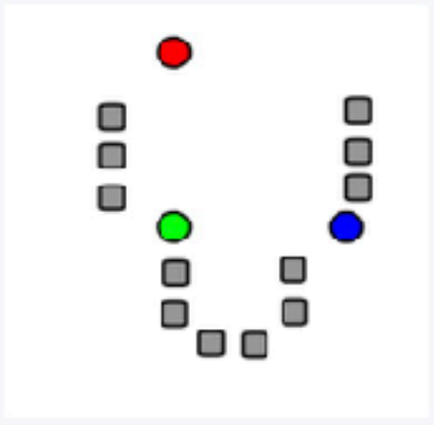
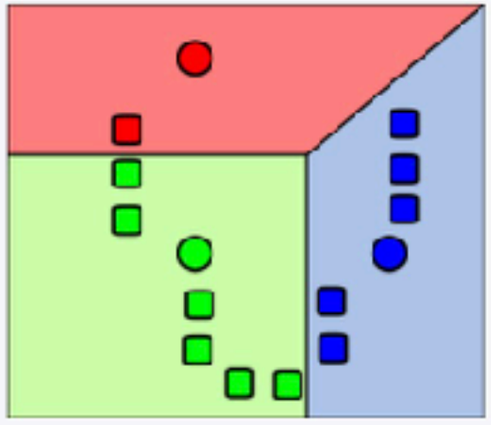
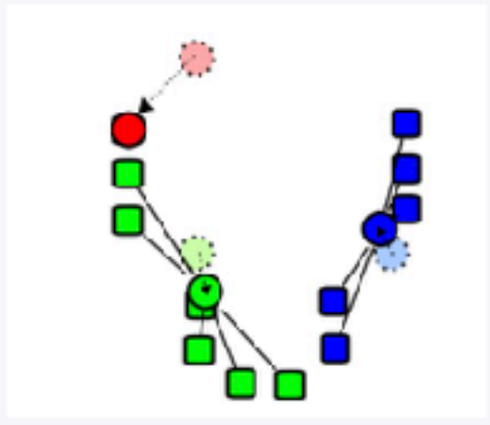
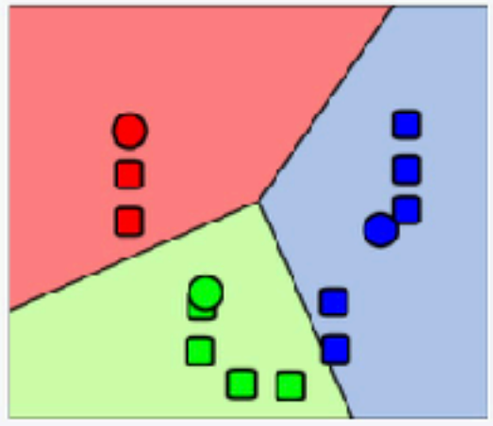
1. Randomly select k initial cluster centers
2. Assign each point to nearest center

$$\boldsymbol{\delta}^t = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{K} \delta_{i,j} \left(c_i^{t-1} - x_j \right)^2$$

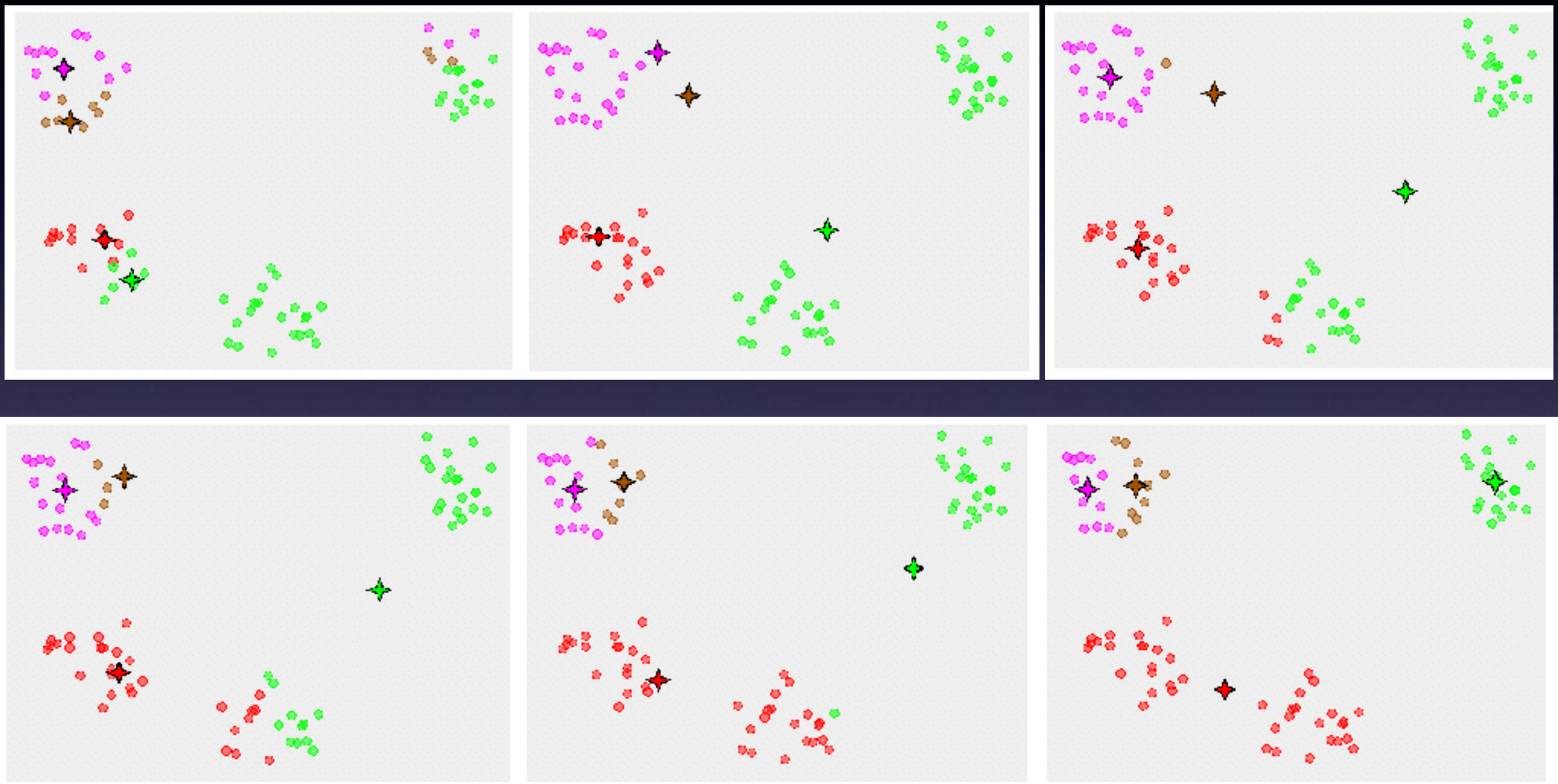
3. Update cluster centers as the mean of the points

$$\boldsymbol{c}^t = \underset{\boldsymbol{c}}{\operatorname{argmin}} \frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{K} \delta_{i,j}^t \left(c_i - x_j \right)^2$$

4. repeat 2-3 until no points are re-assigned.

k-means convergence example



k-means: design choices

- Initialization
  - Randomly select K points as initial cluster center
  - Greedily choose K points to minimize residual
- Distance measures
  - Euclidean or others?
- Optimization
  - Will converge to local minimum
  - May want to use the best out of multiple trials

How to choose k

- Cluster on one set, use another (reserved) set to test K.
- Minimum Description Length (MDL) principal for model comparison.
- Minimize Schwarz Criterion, a.k.a. Bayes Information Criteria (BIC)
- (When building dictionaries, more clusters typically work better.)

How to evaluate clusters?

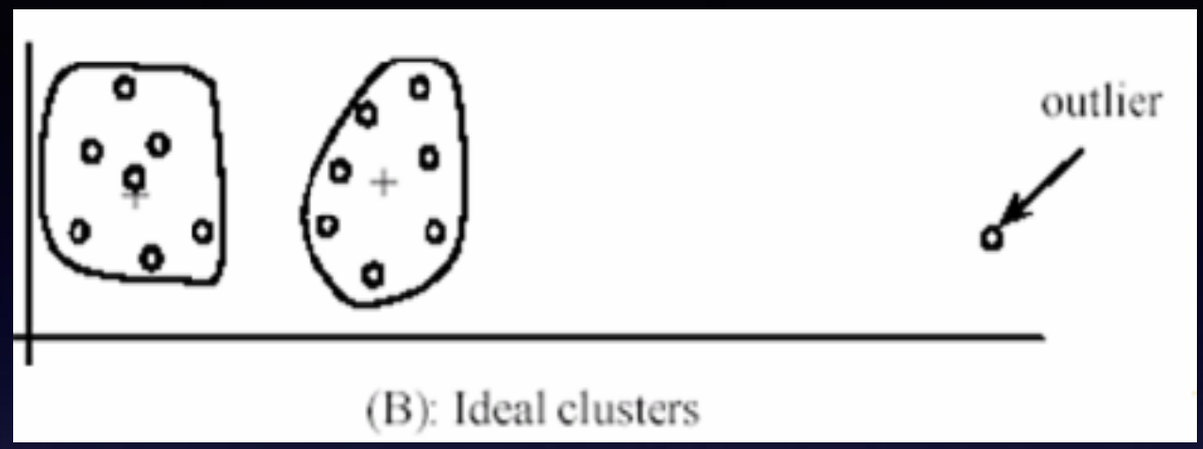
- Generative
  - How well are points reconstructed from the cluster?
- Discriminative
  - How well do the clusters correspond to labels (purity)

k-means pros & cons

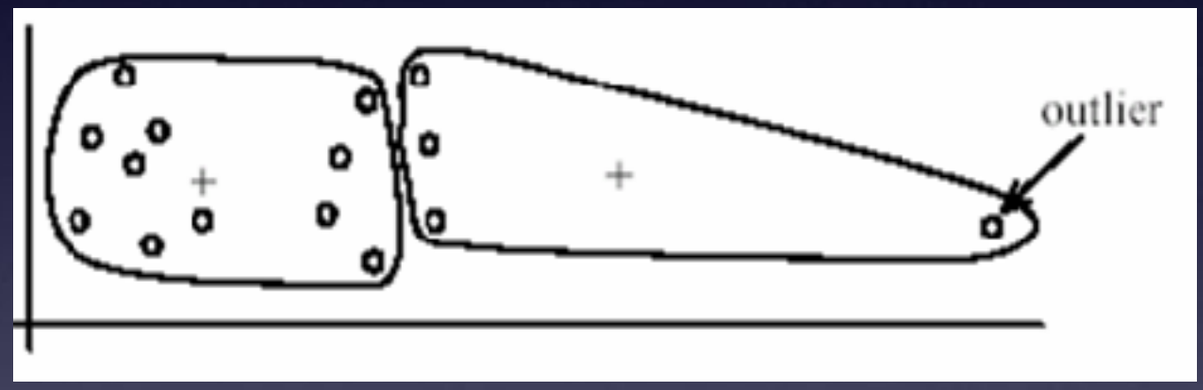
- Pros
    - Finds cluster center that minimize conditional variance (good representation of data)
    - simple and fast
    - easy to implement

k-means pros & cons

- Cons
  - Need to choose K
  - Sensitive to outliers
  - Prone to local minima
  - All clusters have the same parameters
  - Can be slow. Each iteration is O(KNd) for N d-dimensional points



(B): Ideal clusters

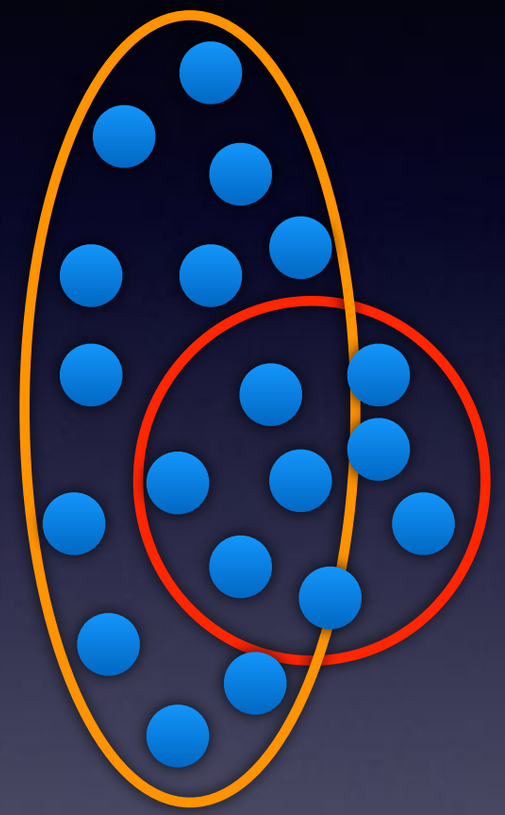


k-means works if

- Clusters are spherical
- Clusters are well separated
- Clusters are of similar volumes
- Clusters have similar number of points

GMM Clustering

- Hard assignments, or probabilistic assignments?
  - Case against hard assignments:
    - Clusters may overlap
    - Clusters may be wider than others

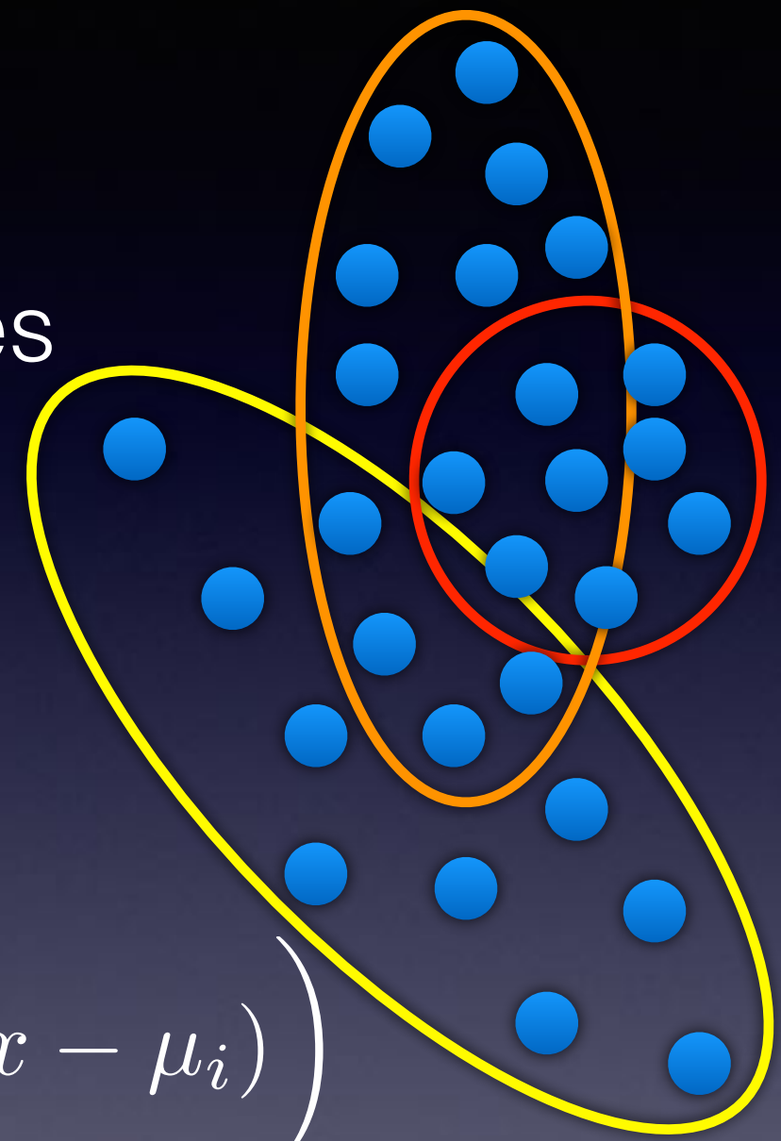


- Can use a probabilistic model,
  - Challenge: need to estimate model parameters without labeled Ys.

$$P(X|Y)P(Y)$$

Gaussian Mixture Models

- Assume m-dimensional data points
- $P(Y)$ still multinomial, with k classes
- $P(\mathbf{X}|Y=i),\ i=1,\cdots,k$ are k multivariate Gaussians



$$P(X=x|Y=i)$$

$$= \frac{1}{\sqrt{(2\pi)^m|\Sigma_i|}} \exp\left(-\frac{1}{2}(x-\mu_i)^{\mathrm{T}}\Sigma^{-1}(x-\mu_i)\right)$$

determinant of matrix

variance (m*m matrix)

mean (m-dim vector)

Maximum Likelihood Estimation (MLE)

- Estimating parameters (when given data label *Y*)
  - Solve optimization problem:

$$P(X = x|Y = i) = \frac{1}{\sqrt{(2\pi)^m |\Sigma_i|}} \exp\left(-\frac{1}{2}(x - \mu_i)^{\mathrm{T}} \Sigma^{-1}(x - \mu_i)\right)$$

  - MLE has closed form solution:

$$\mu_{ML} = \frac{1}{n}\sum_{i=1}^{n} x^i \quad \Sigma_{ML} = \frac{1}{n}\sum_{i=1}^{n}(x^i - \mu_{ML})(x^i - \mu_{ML})^{\mathrm{T}}$$

  - i.e., solve $\mathrm{argmax}_\theta \prod_j P(y^j, x^j; \theta)$
- Estimating parameters (without y^j), solve:

$$\mathrm{argmax}_\theta \prod_j P(x^j, \theta) = \mathrm{argmax} \prod_j \sum_{i=1}^{k} P(y^j = i, x^j; \theta)$$

θ: all model param including mean, variance, etc.

Solving MLE for GMM Clustering

- Maximize marginal likelihood

$$\text{argmax}_\theta \prod_j P(x^j, \theta) = \text{argmax} \prod_j \sum_{i=1}^{k} P(y^j = i, x^j; \theta)$$

  - Almost always a hard problem
  - Usually no closed form solution
  - Even when $P(X, Y; \theta)$ is convex, $P(X; \theta)$ generally isn't
  - For all but the simplest $P(X; \theta)$, *we will have to do gradient ascent, in a big messy space with lots of local optimum.*

Solving MLE for GMM Clustering

- Simple example: GMM with 1D data, k=2 Gaussians, variance=1, distribution over classes is uniform, only need to estimate $\mu_1,\ \mu_2$.

$$\prod_{j=1}^{n}\sum_{i=1}^{k} P(X = x^j, Y = i) \propto \prod_{j=1}^{n}\sum_{i=1}^{k} \exp\left(-\frac{1}{2\sigma^2}(x^j - \mu_i)^2\right)$$

- Skipping the derivations.... still need to differentiate and solve for μ_i, Σ_i and $P(Y=1)$ for i=1...k. There are still no closed form solution, gradient is complex with lots of local optimum.

Solving MLE for GMM Clustering

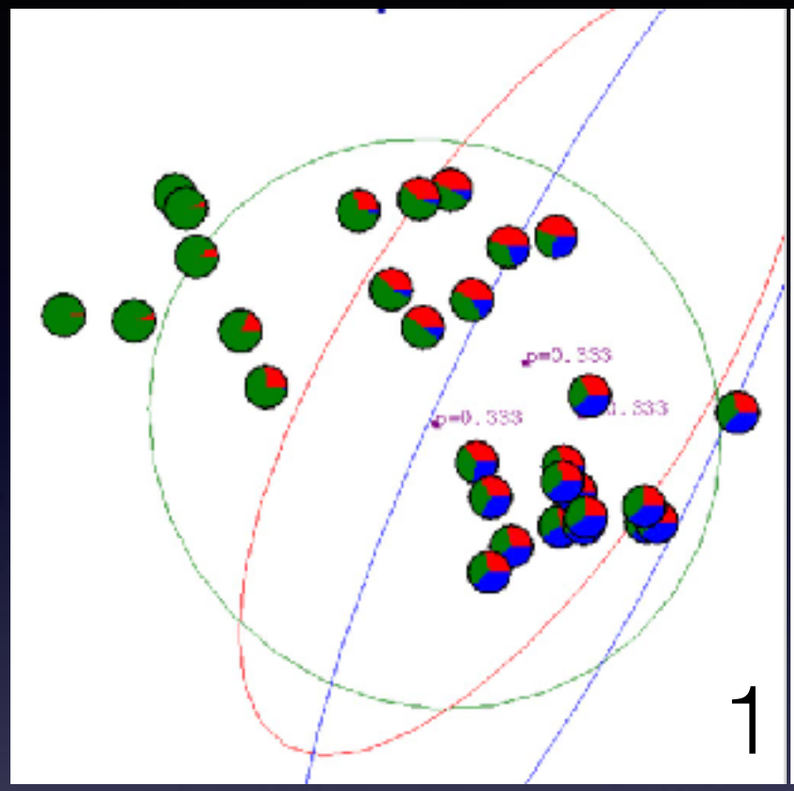
- **Expectation Maximization**
  - Objective:

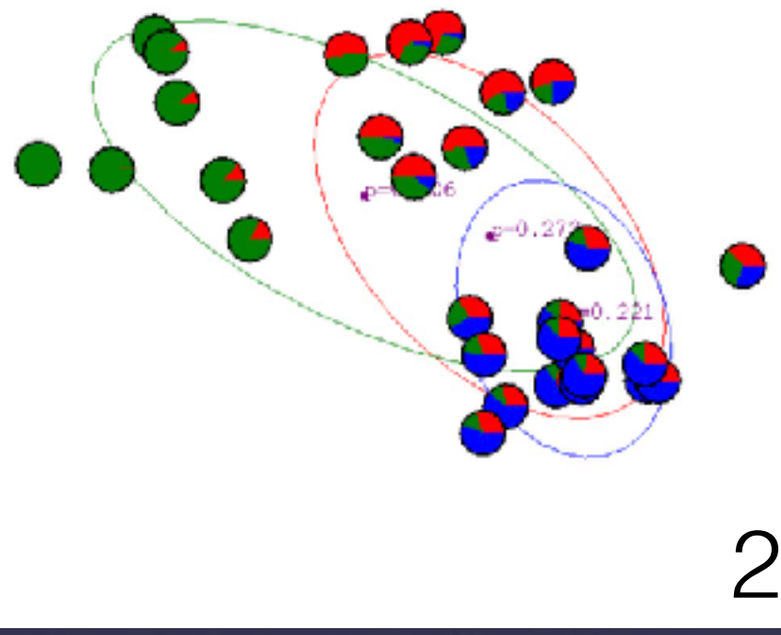
$$\underset{\theta}{\operatorname{argmax}} \prod_j \sum_{i=1}^{k} P(y^j = i, x^j|\theta) = \sum_j \log \sum_{i=1}^{k} P(y^j = i, x^j|\theta)$$

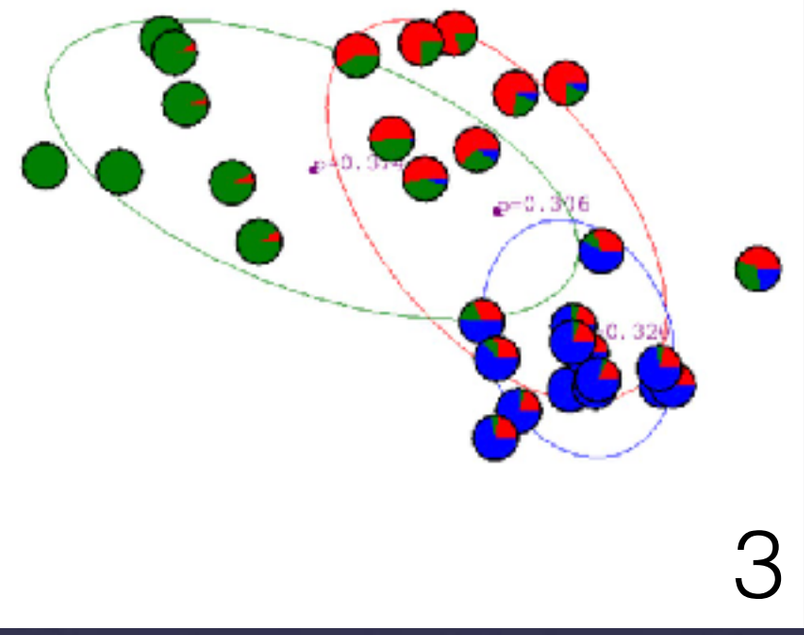
  - Data: $\{x^j|j = 1 \ldots n\}$
  - **E-step**: For all examples j and values i for y, compute: $P(y^j = i|x^j, \theta)$
  - **M-step**: re-estimate the parameters with weighted MLE estimates, set:

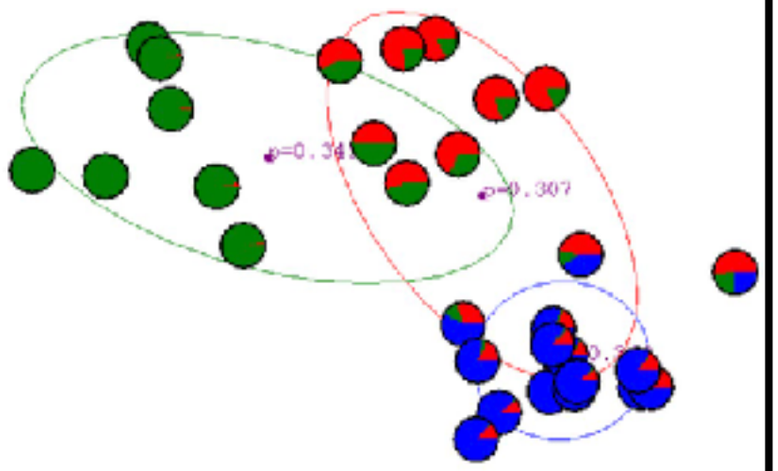
$$\theta = \operatorname{argmax}_\theta \sum_j \sum_{i=1}^{k} P(y^j = i|x^j, \theta) \log P(y^j = i, x^j|\theta)$$

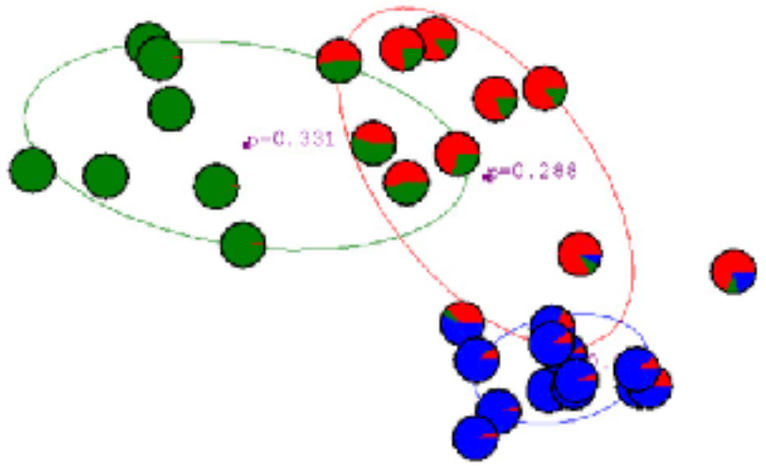
EM for GMM MLE example

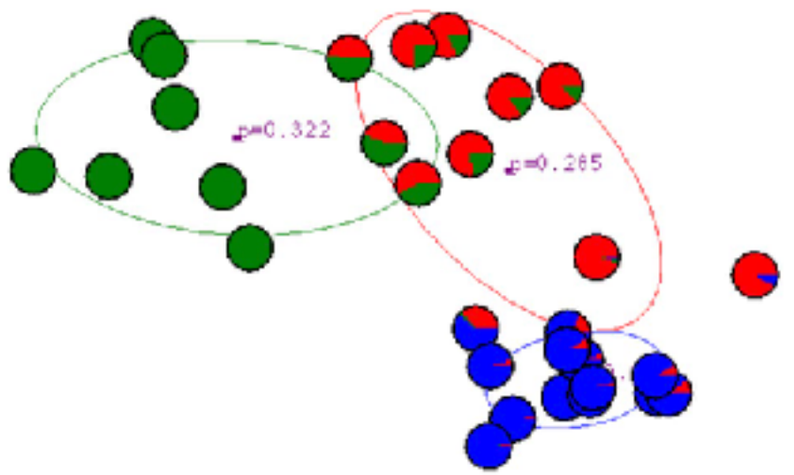






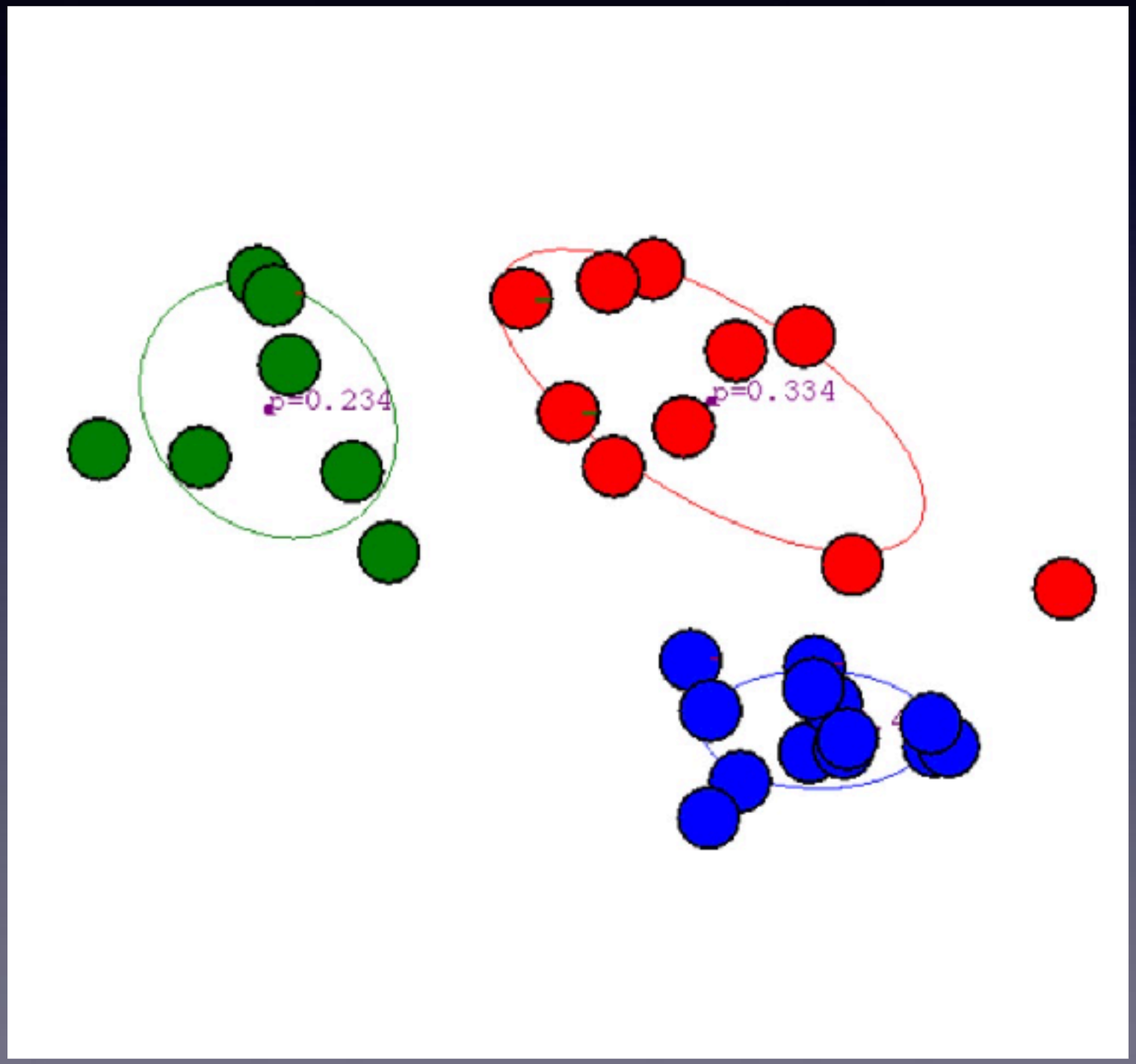






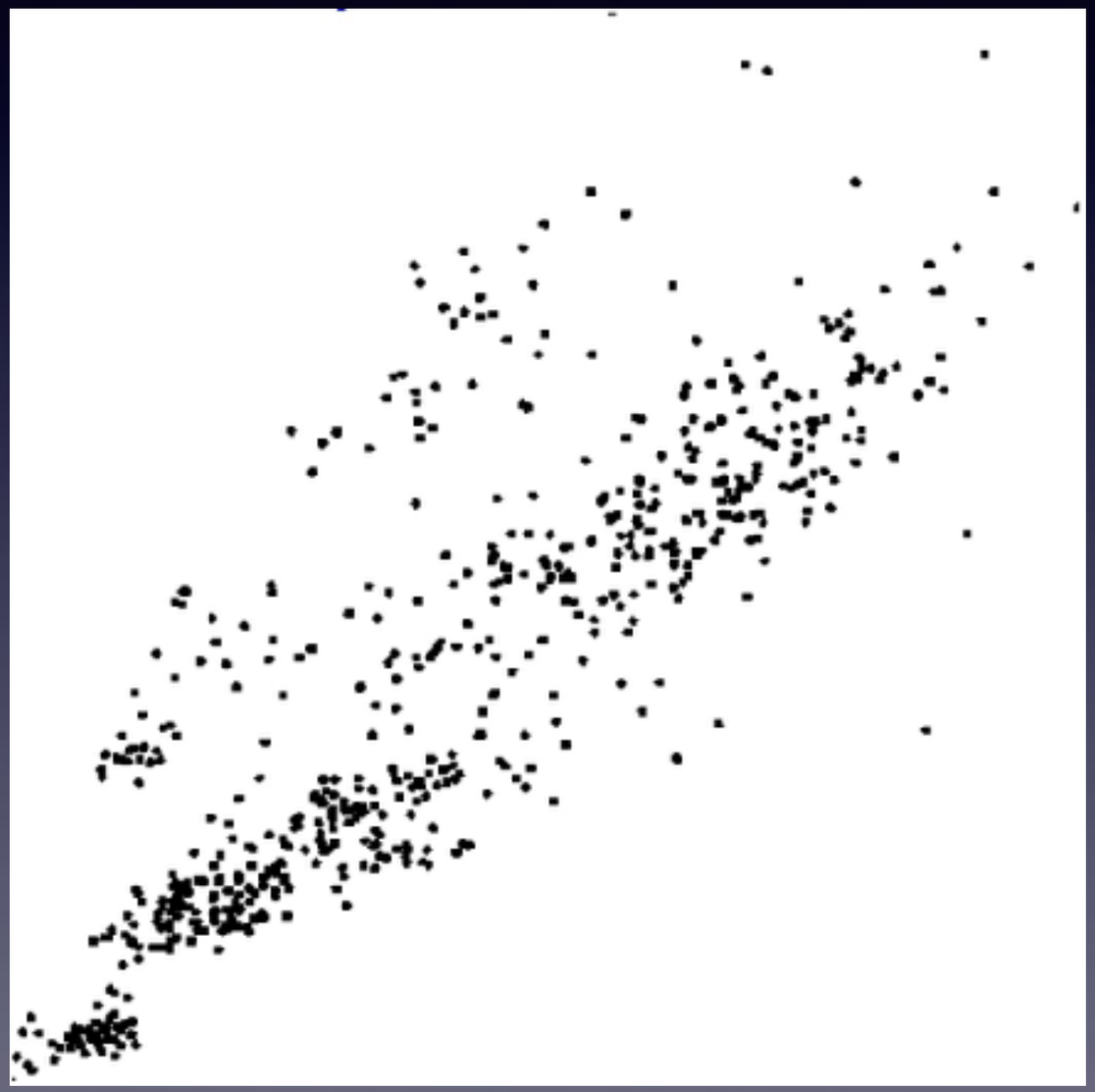
EM for GMM MLE example

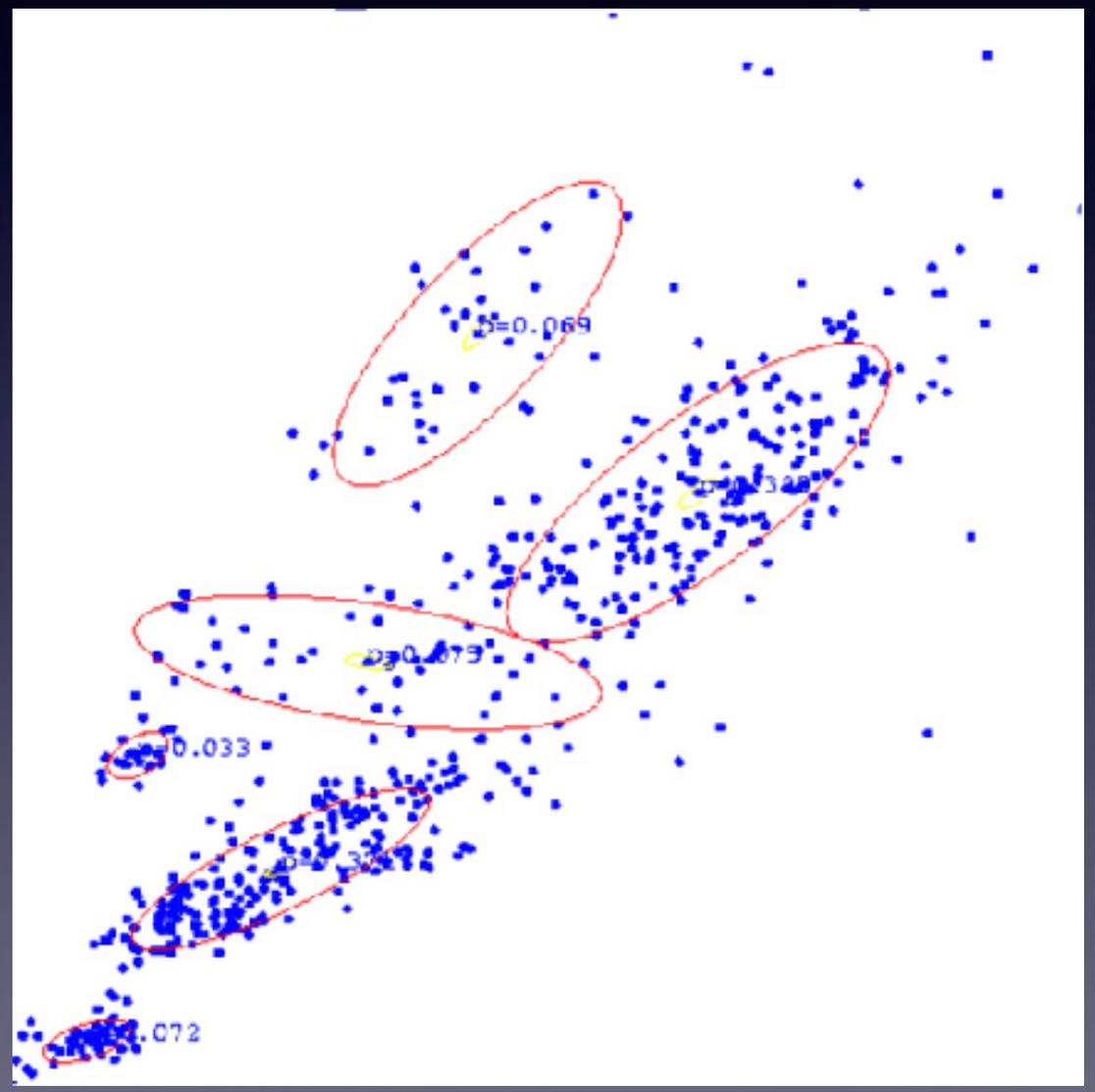
- EM after 20 iterations



EM for GMM MLE example

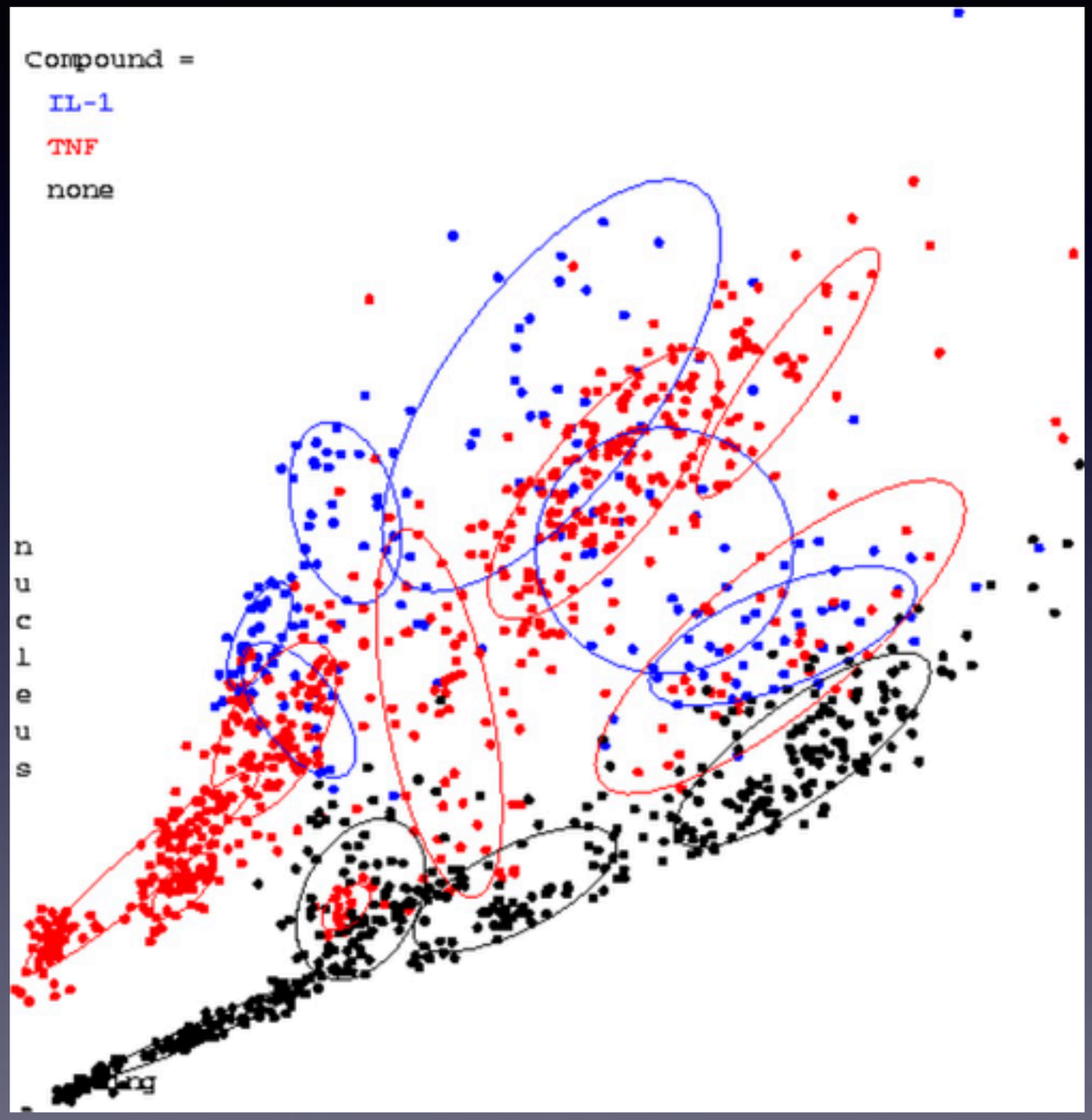
- GMM for some bio assay data





EM for GMM MLE example

- GMM for some bio assay data, fitted separately for three diffrent compounds.



EM for GMM Clustering, notes

- GMM with hard assignments and unit variance, EM is equivalent to k-means clustering algorithm!!!

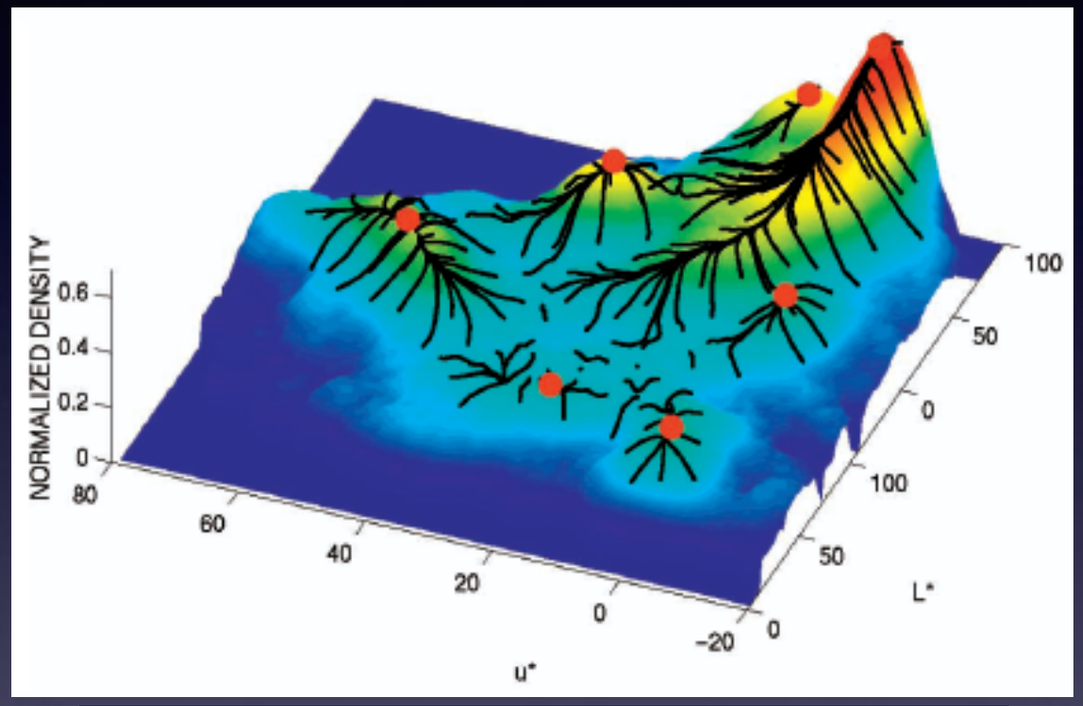
- EM, like k-NN, uses coordinate ascent, and can get stuck in local optimum.

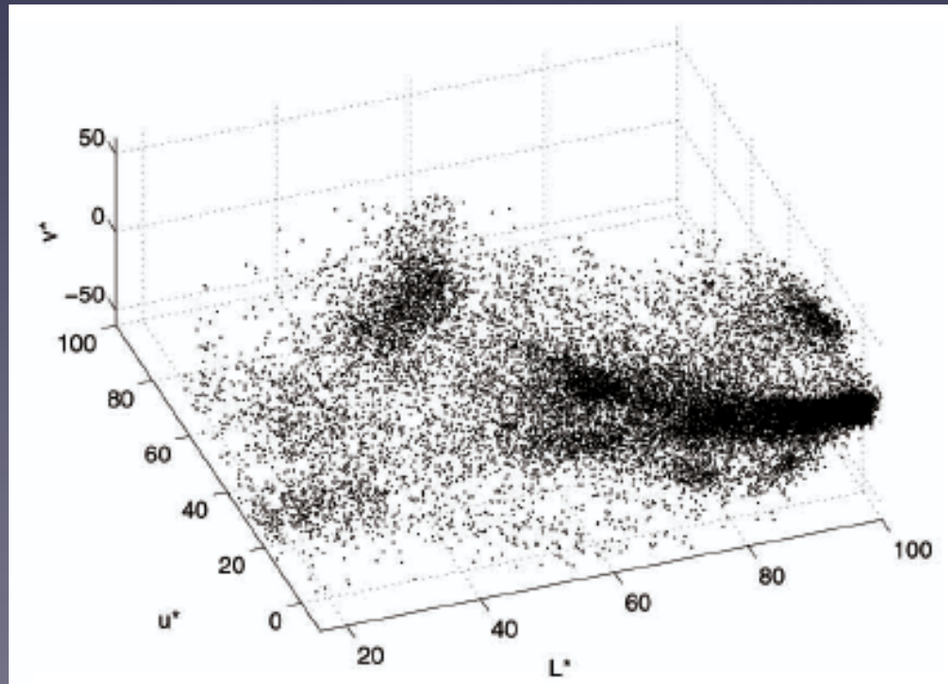
Mean-Shift Clustering

- mean-shift seeks *modes* of a given set of points
    1. Choose kernel and bandwidth
    2. For each point:
        1. center a window on that point
        2. compute the mean of the data in the search window
        3. center the search window at the new mean location, repeat 2,3 until converge.
    3. Assign points that lead to nearby modes to the same cluster.

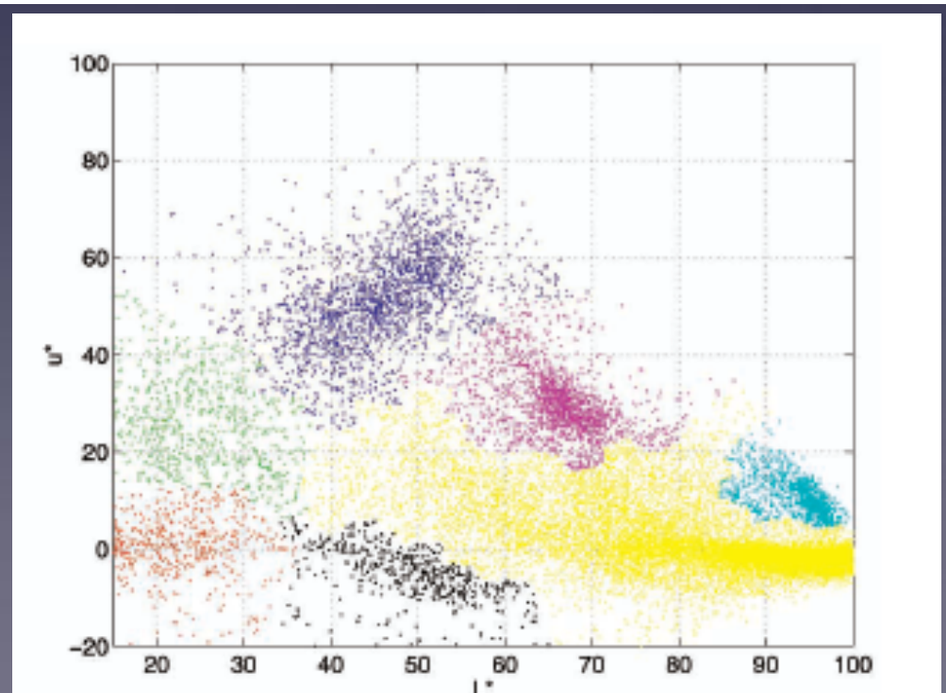
Mean-shift algorithm

- Try to find *modes* of a non-parametric density





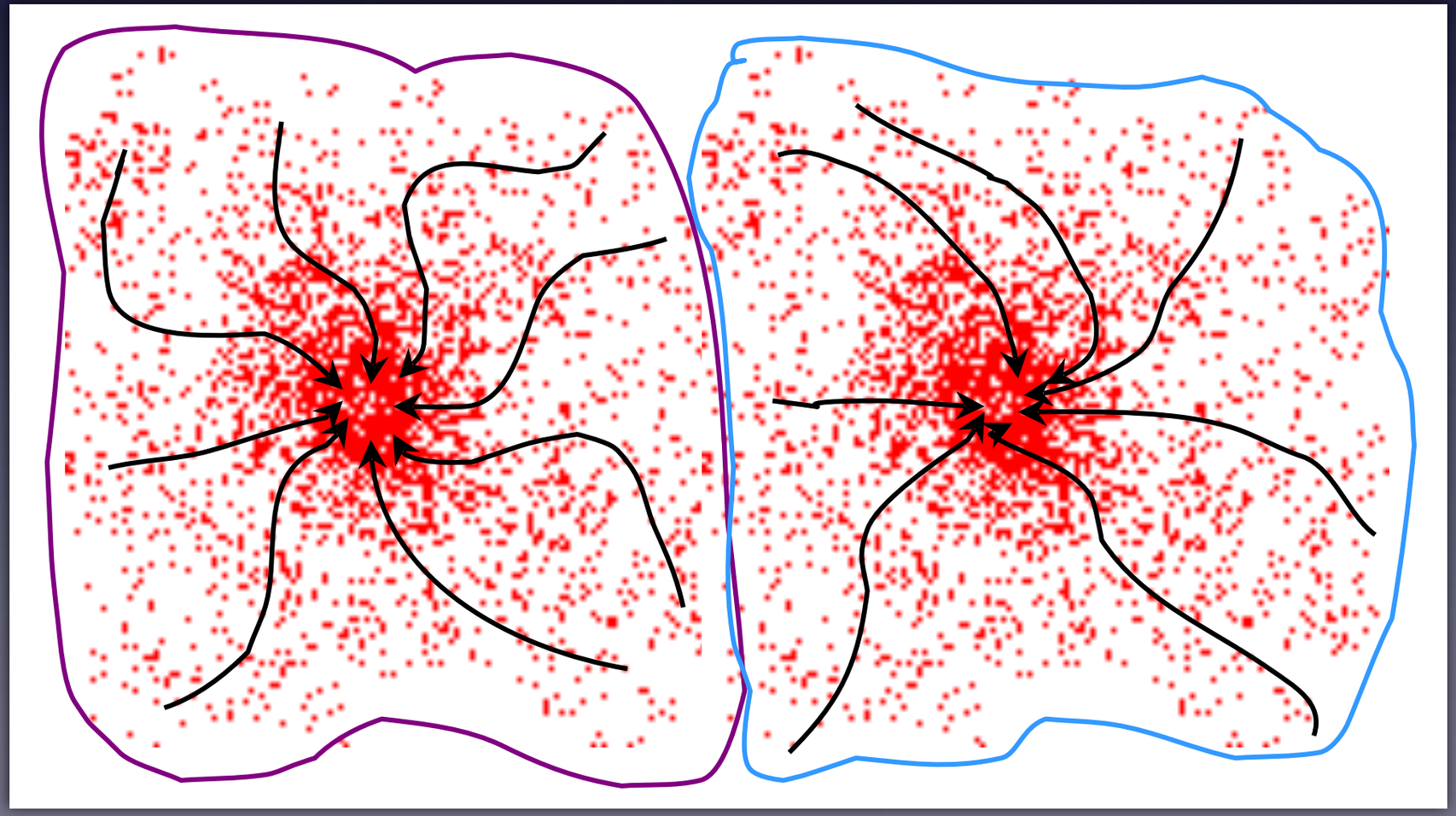
Color space



Color space clusters

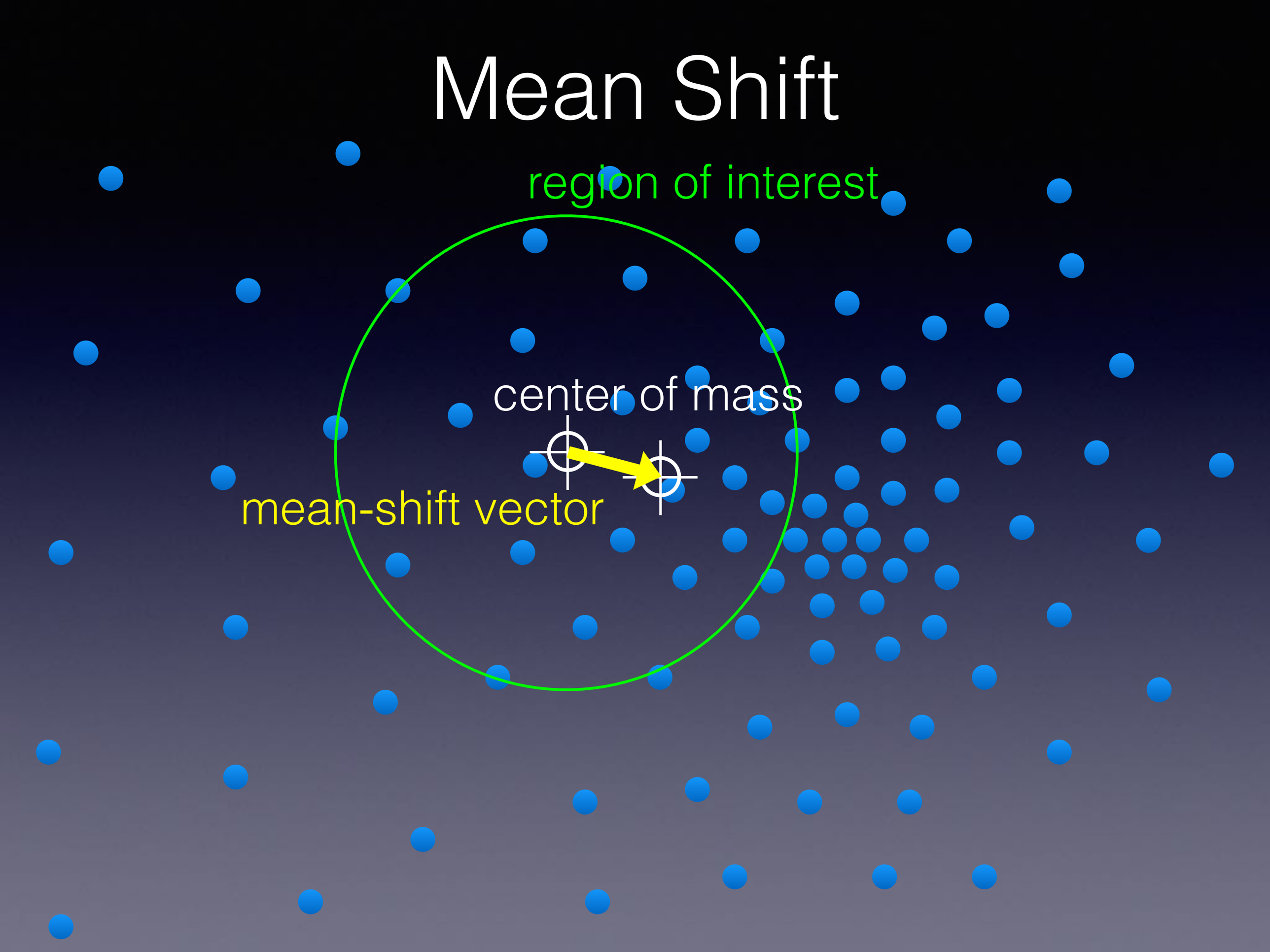
Attraction Basin

- Attraction basin: the region for which all trajectories lead to the same mode.
- Cluster: all data points in the attraction basin of a mode.

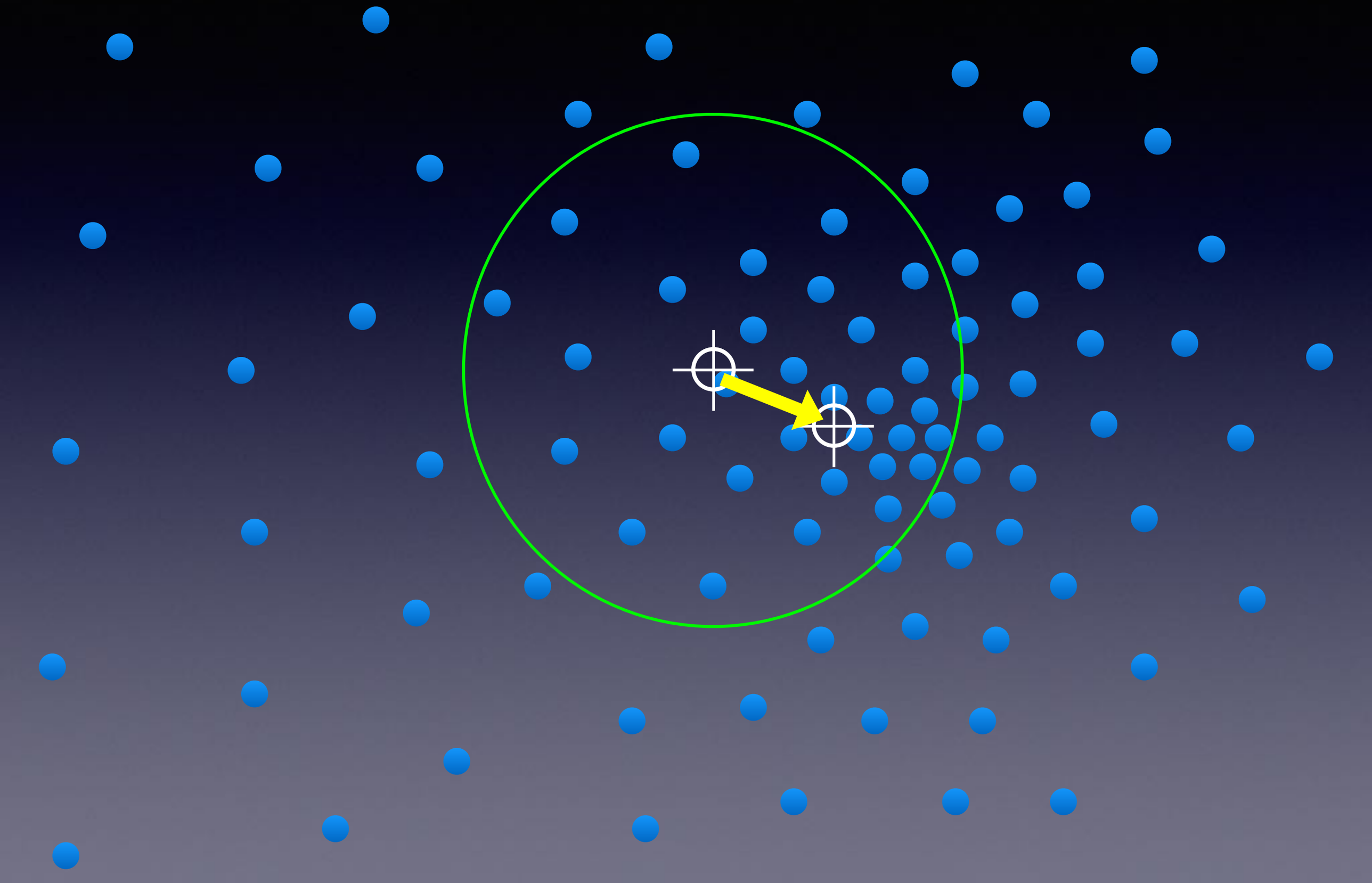


Slides by Y. Ukrainitz & B. Sarel

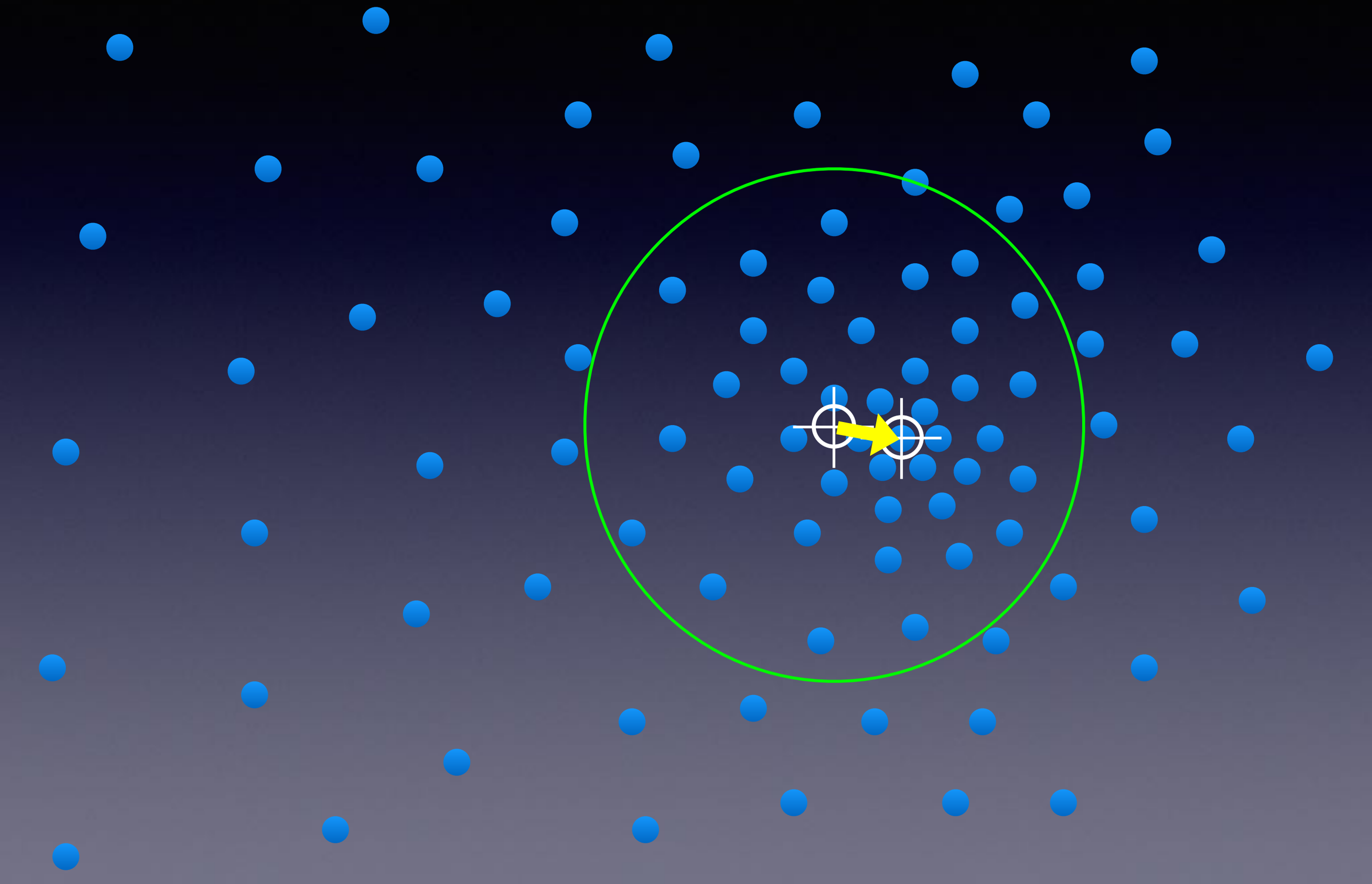
Mean Shift



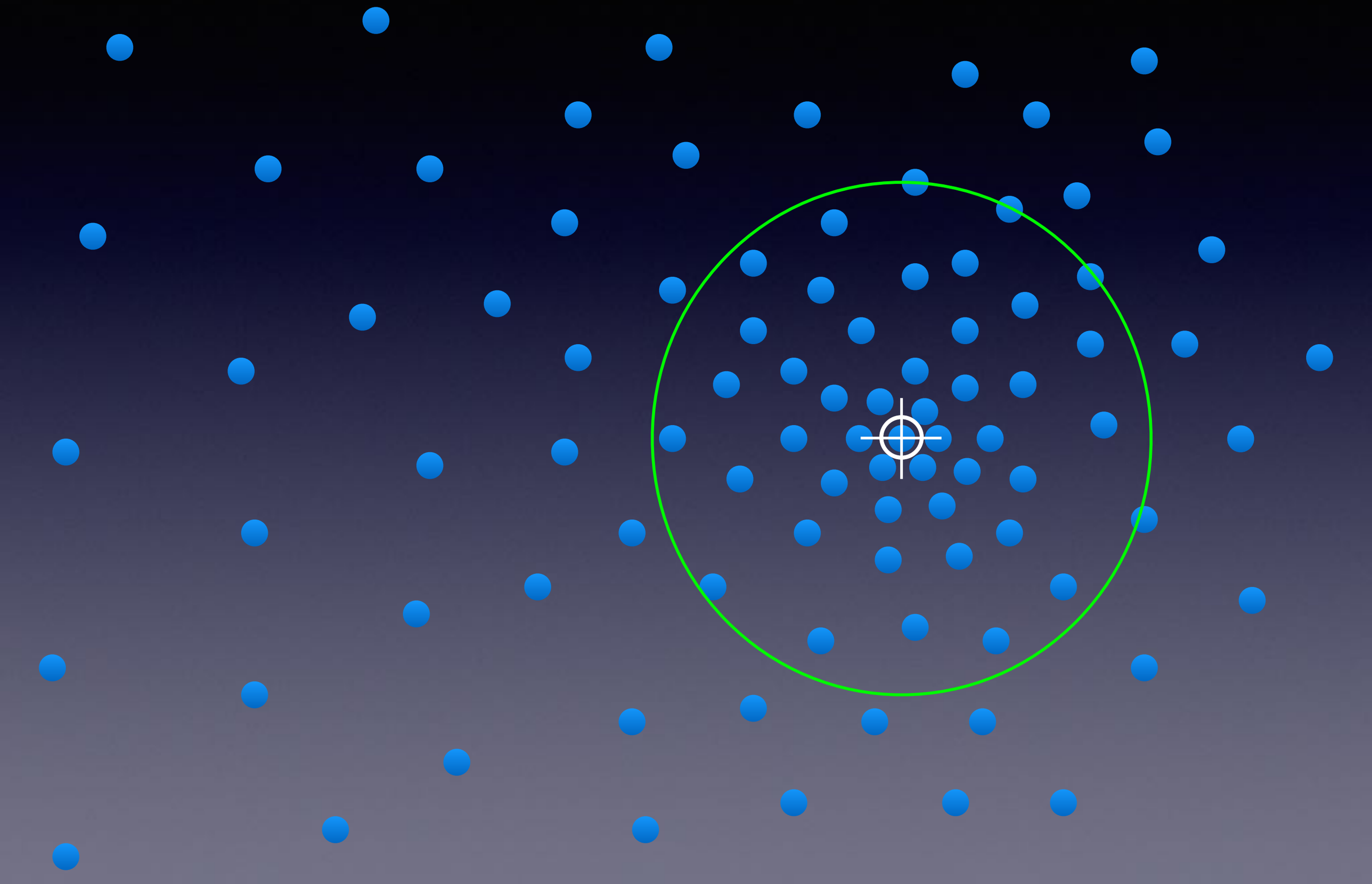
Mean Shift



Mean Shift



Mean Shift



Kernel Density Estimation

- Kernel density estimation function

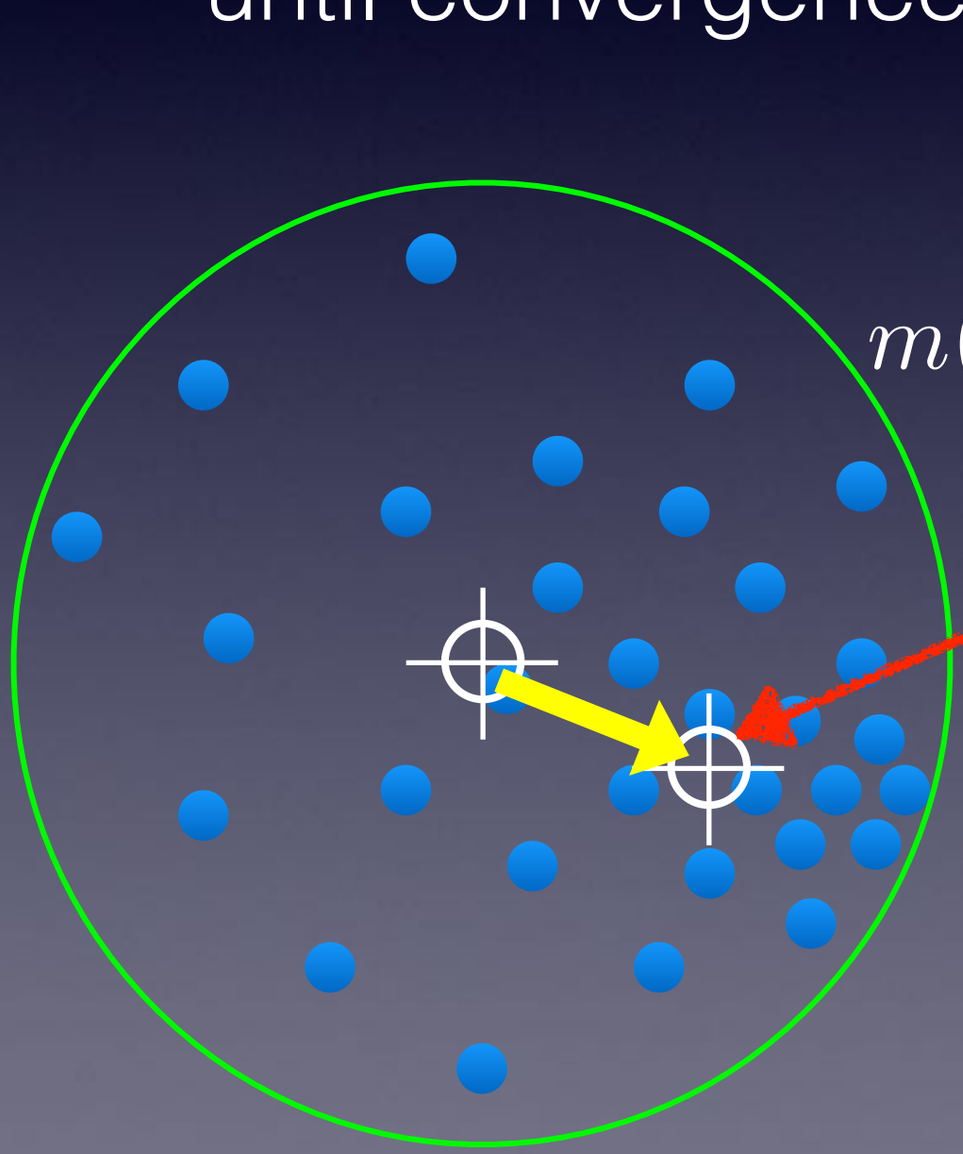
$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

- Gaussian kernel

$$K\left(\frac{x - x_i}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-xi)^2}{2h^2}}$$

Computing the Mean Shift

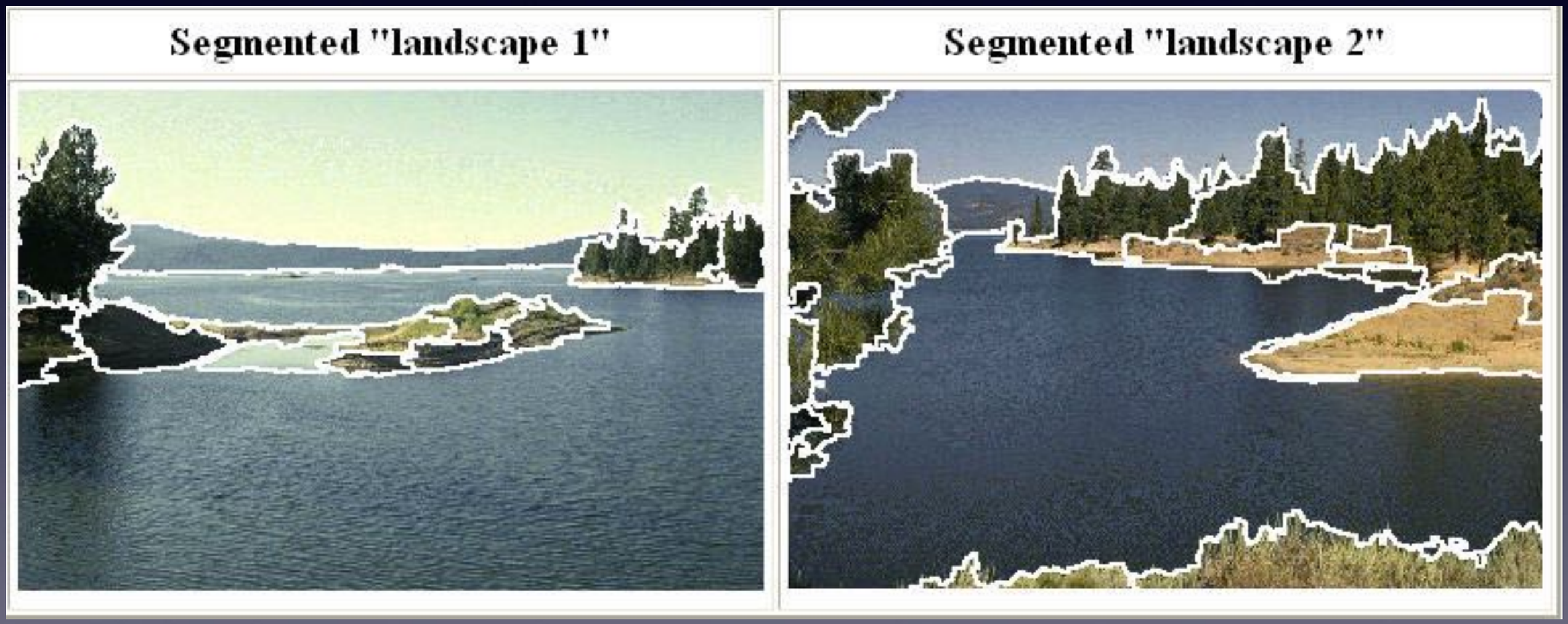
- Compute mean shift vector m(**x**)
- Iteratively translate the kernel window y m(**x**) until convergence



$$m(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\frac{||\mathbf{x}-\mathbf{x}_i||^2}{h}\right)}{\sum_{i=1}^{n} g\left(\frac{||\mathbf{x}-\mathbf{x}_i||^2}{h}\right)} - \mathbf{x} \right]$$

Mean-Shift Segmentation

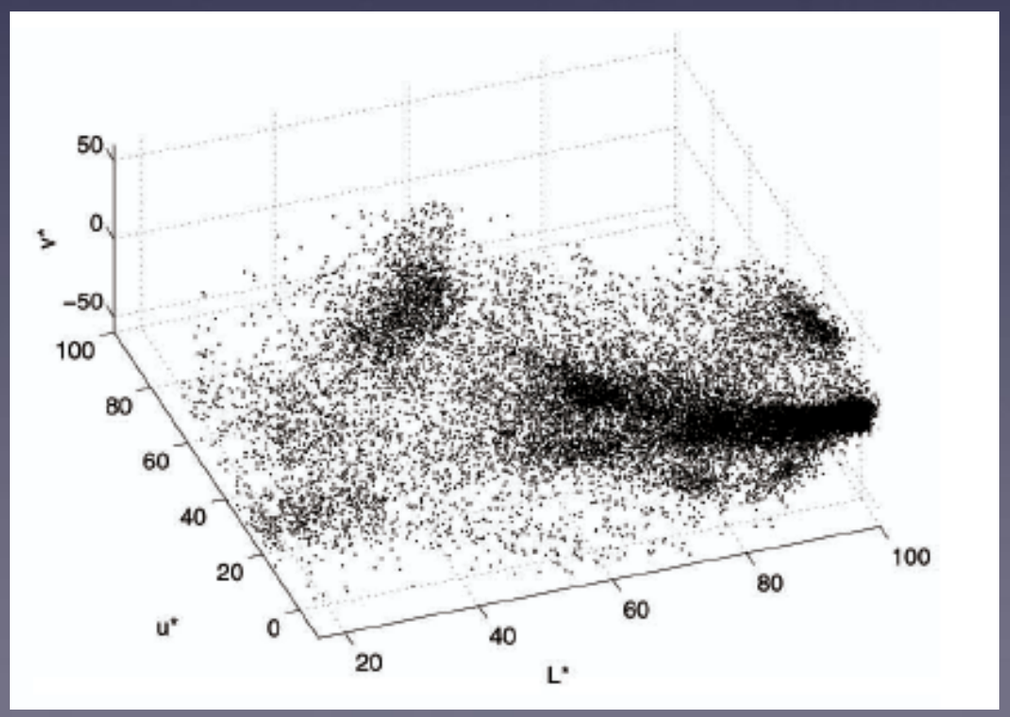
- Mean-shift can also be used as clustering-based image segmentation.



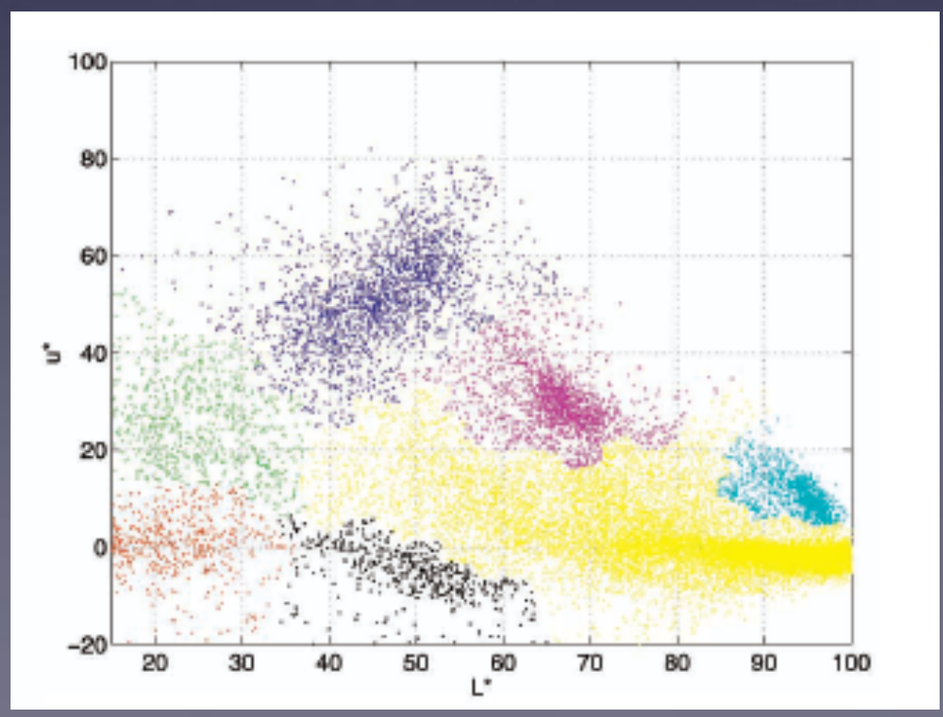
D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

Mean-Shift Segmentation

- Compute features for each pixel (color, gradients, texture, etc.).
- Set kernel size for features K_f and position K_s.
- Initialize windows at individual pixel locations.
- Run mean shift for each window until convergence.
- Merge windows that are within width of K_f and K_s.



Color space



Color space clusters

Mean-Shift

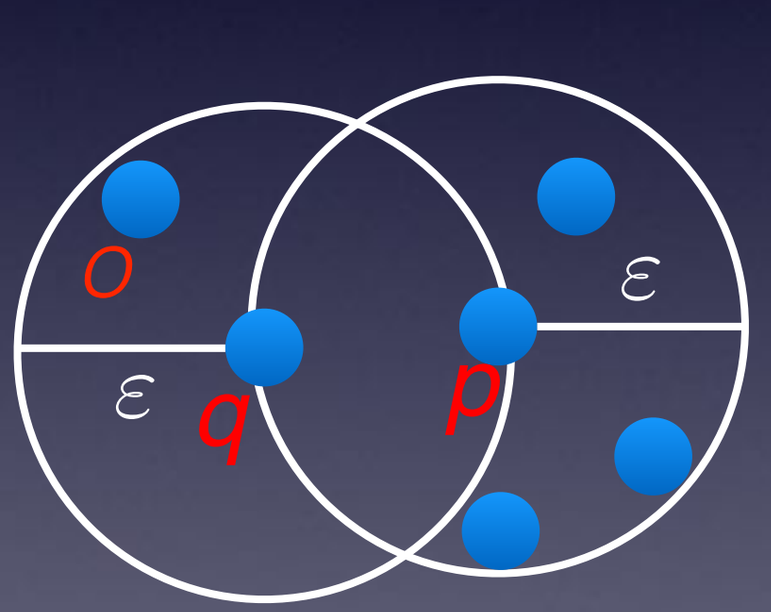
- Speedups:
  - binned estimation
  - fast neighbor search
    - update each window in each iteration
- Other tricks
  - Use kNN to determine window sizes adaptively

Mean-Shift pros & cons

- Pros
  - Good general-practice segmentation
  - Flexible in number and shape of regions
    - robust to outliers
- Cons
  - Have to choose kernel size in advance
  - Not suitable for high-dimensional features

DBSCAN

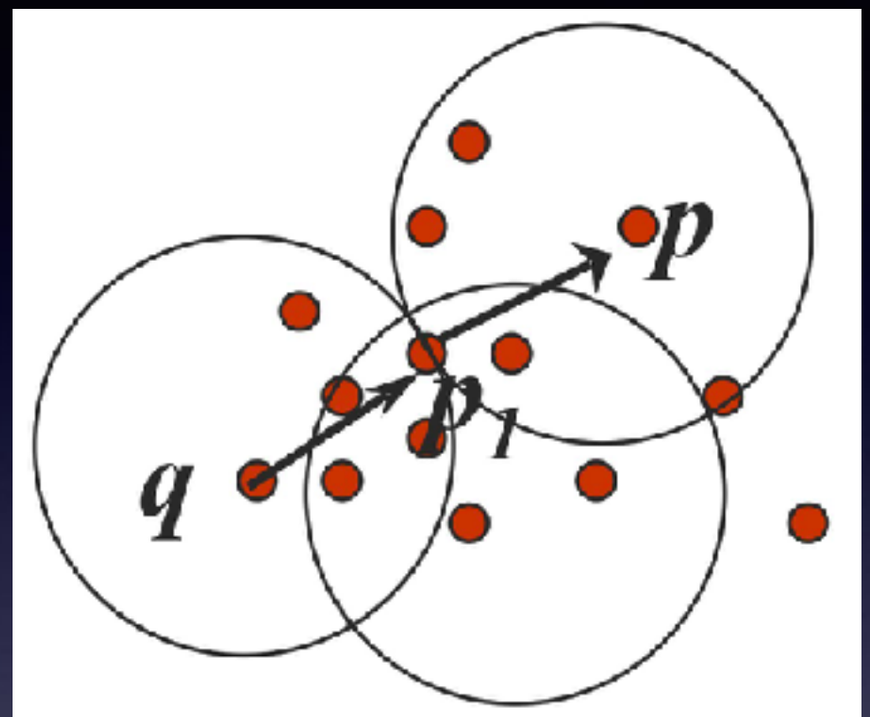
- **DBSCAN**: Density-based spatial clustering of applications with noise.
  - *Density*: number of points within a specified radius (*ε*-Neighborhood)
  - *Core point*: a point with more than a specified number of points (*MinPts*) within *ε*.
  - *Border point*: has fewer than *MinPts* within *ε*, but is in the neighborhood of a core point.
  - *Noise point*: any point that is not a core point or border point.



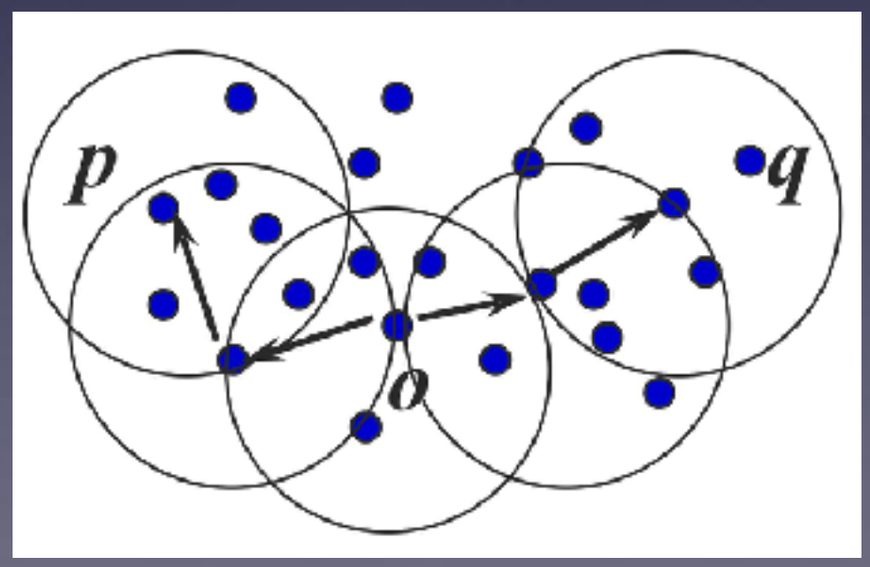
MinPts=4
p is core point
q is border point
o is noise point

DBSCAN

- *Density-reachable*: *p* is density-reachable from *q* w.r.t. *ε* and *MinPts* if there is a chain of objects p_1, ..., p_n with $p_1=q$ *and* $p_n=p$, s.t. p_{i+1} is directly density-reachable from p_i w.r.t. *ε* and *MinPts* for all $1 \leq i \leq n$



- *Density-connectivity*: *p* is density-connected to *q* w.r.t. *ε* and *MinPts* if there is an object *o, s.t.* both *p* and *q* are density-reachable from *o* w.r.t. *ε* and *MinPts*.



DBSCAN clustering

- *Cluster*: a cluster ***C*** in a set of objects ***D*** w.r.t. ε and *MinPts* is a non-empty subset of D satisfying
  - *Maximality*: for all *p*,*q*, if $p \in$ C and if *q* is density reachable from *p* w.r.t. *ε*.
  - *Connectivity*: for all $p,q \in \boldsymbol{C}$, *p* is density-connected to *q* w.r.t. *ε* and *MinPts* in ***D***.
  - Note: cluster contains core & border points.
- *Noise*: objects which are not directly density-reachable from at least one core object.

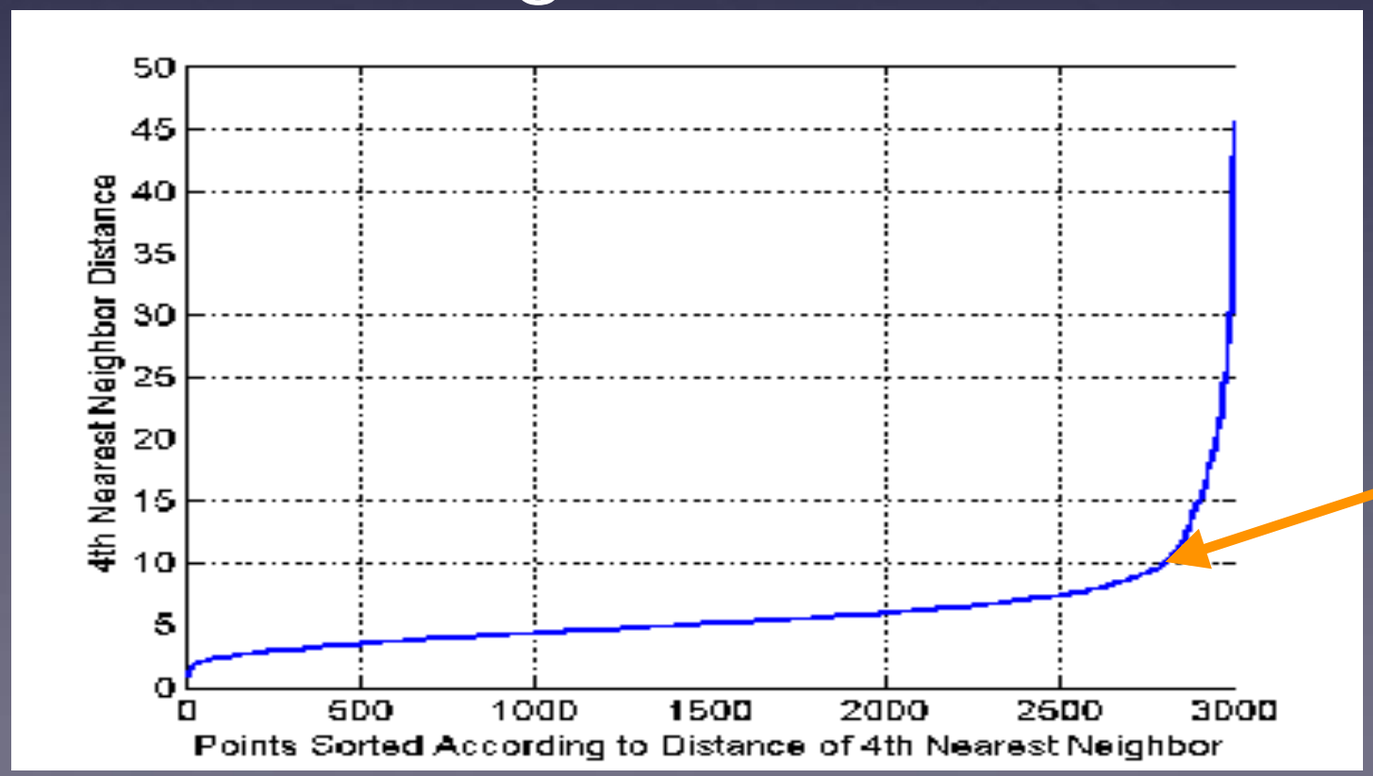
DBSCAN clustering algorithm

1. Select a point *p*
2. Retrieve all points density-reachable from *p* w.r.t. *ε* and MinPts.
    1. if *p* is a core point, a cluster is formed
    2. if *p* is a border point, no points are density reachable from *p* and DBSCAN visits the next point of the database
3. continue 1,2, until all points are processed.

(result independent of process ordering)

DBSCAN parameters

- Heuristic: for points in a cluster, their k^{th} nearest neighbors are at roughly the same distance.
- Noise points have the k^{th} nearest neighbor at farthest distance.
- So, plot sorted distance of every point to its k^{th} nearest neighbor.



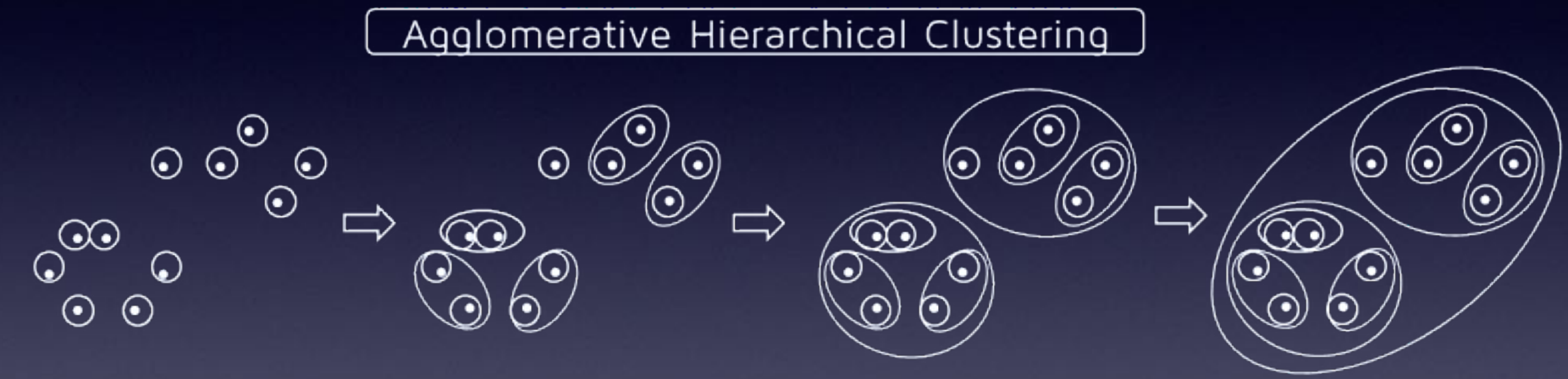
sharp change; good candidate for ε and MinPts.

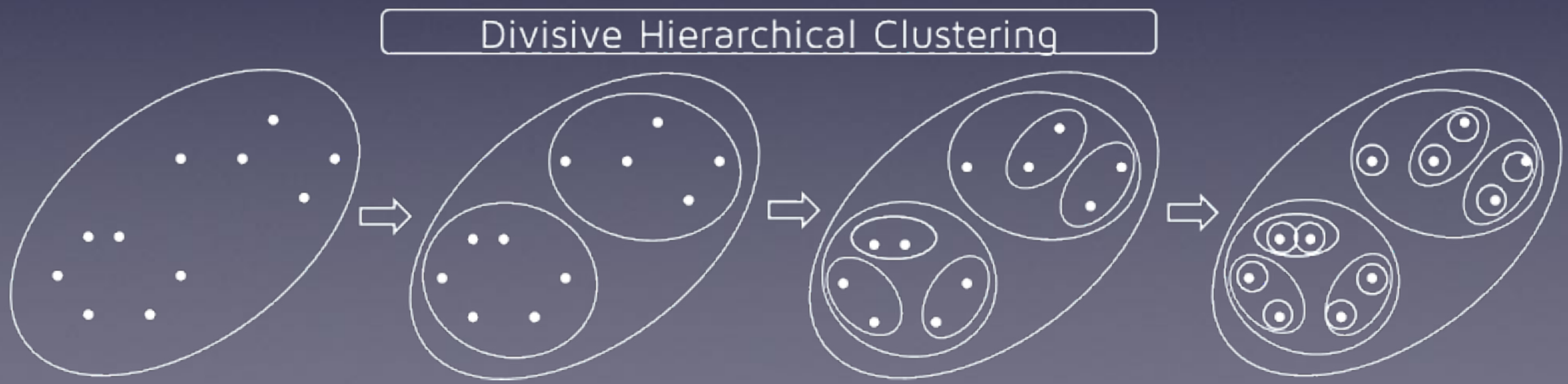
DBSCAN pros & cons

- Pros
  - No need to decide K beforehand,
  - Robust to noise, since it doesn't require every point being assigned nor partition the data.
  - Scales well to large datasets with .
  - Stable across runs and different data ordering.
- Cons
  - Trouble when clusters have different densities.
  - ε may be hard to choose.

Hierarchical Clustering

- Agglomerative clustering v.s. Divisive clustering





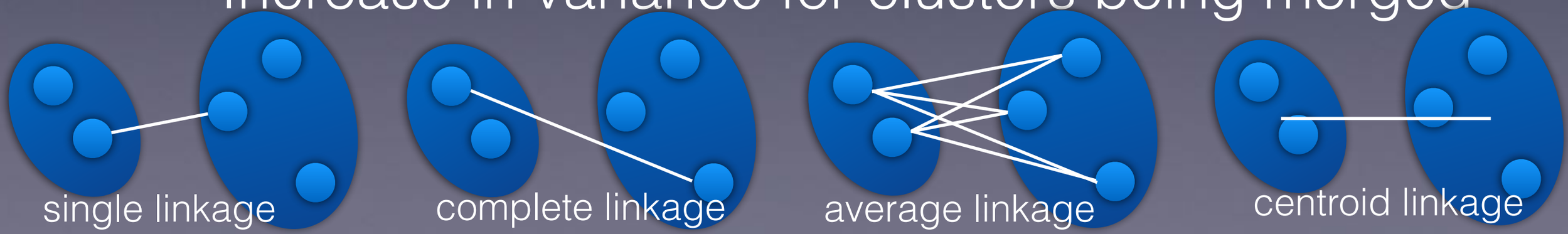
Agglomerative Clustering

- Method:
  1. Every point is its own cluster
  2. Find ***closest*** pair of clusters, merge into one
  3. repeat

- The definition of ***closest*** is what differentiates various flavors of agglomerative clustering algorithms.

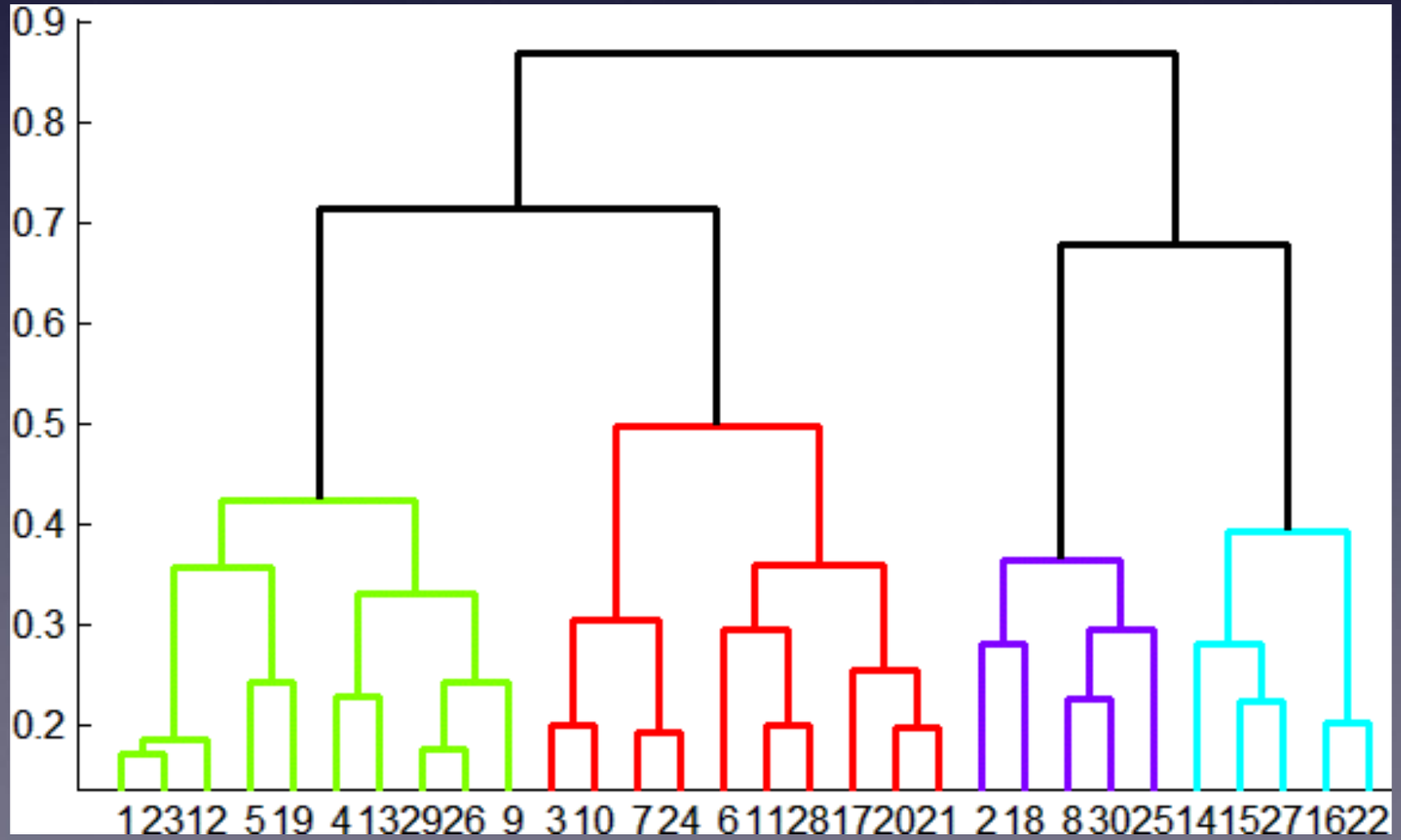
Agglomerative Clustering

- How to define the linkage/cluster similarity?
  - Maximum or complete-linkage clustering (a.k.a., farthest neighbor clustering)
  - Minimum or single linkage clustering (UPGMA) (a.k.a., nearest neighbor clustering)
  - Centroid linkage clustering (UPGMC)
  - Minimum Energy Clustering
  - Sum of all intra-cluster variance
  - Increase in variance for clusters being merged



Agglomerative Clustering

- How many clusters?
  - Clustering creates a dendrogram (a tree)
  - Threshold based on max number of clusters or based on distance between merges.

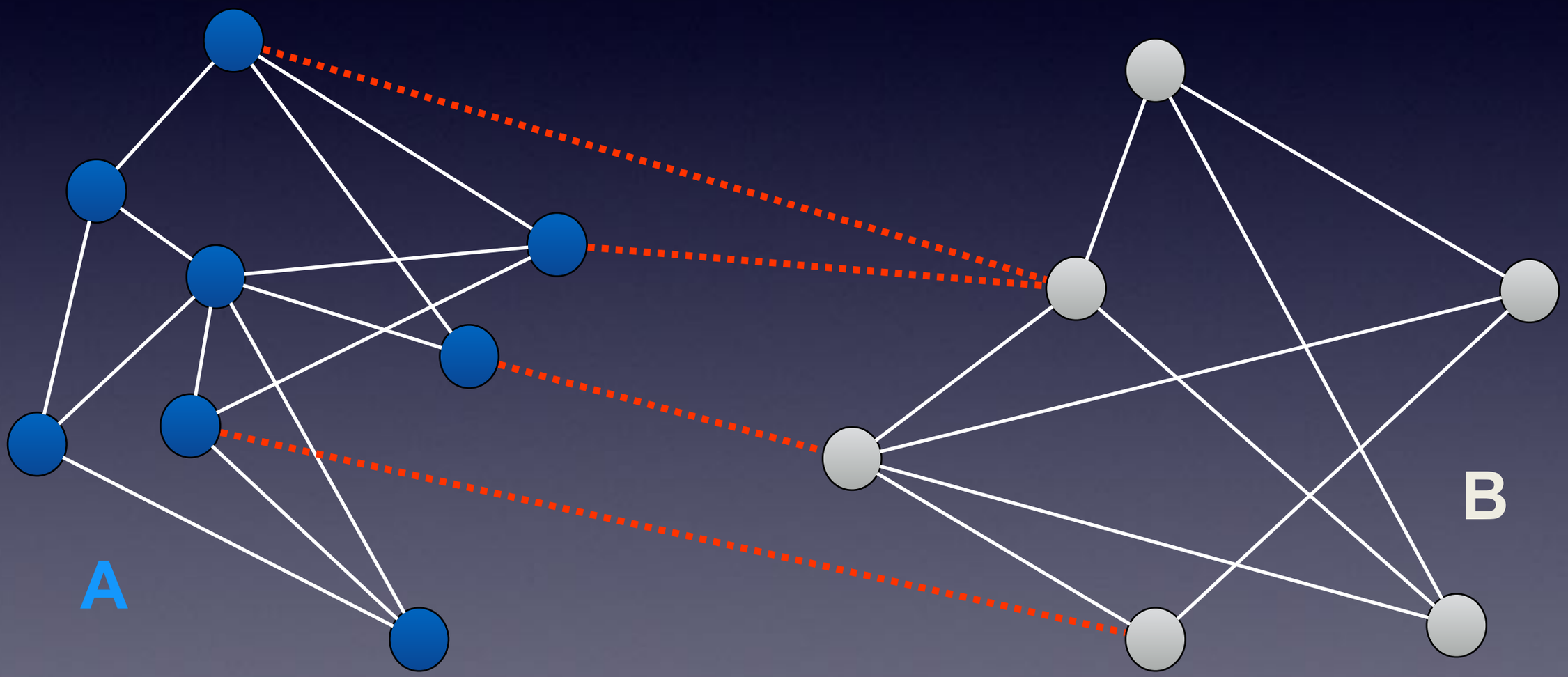


Agglomerative Clustering

- Pros
  - Simple to implement, widespread application
  - Clusters have adaptive shapes
  - Provides a hierarchy of clusters
- Cons
  - May have imbalanced clusters
  - Still have to choose the number of clusters or thresholds
  - Need to use an ultrametric to get a meaningful hierarchy

Spectral Clustering

- Group points based on links in a graph



Spectral Clustering

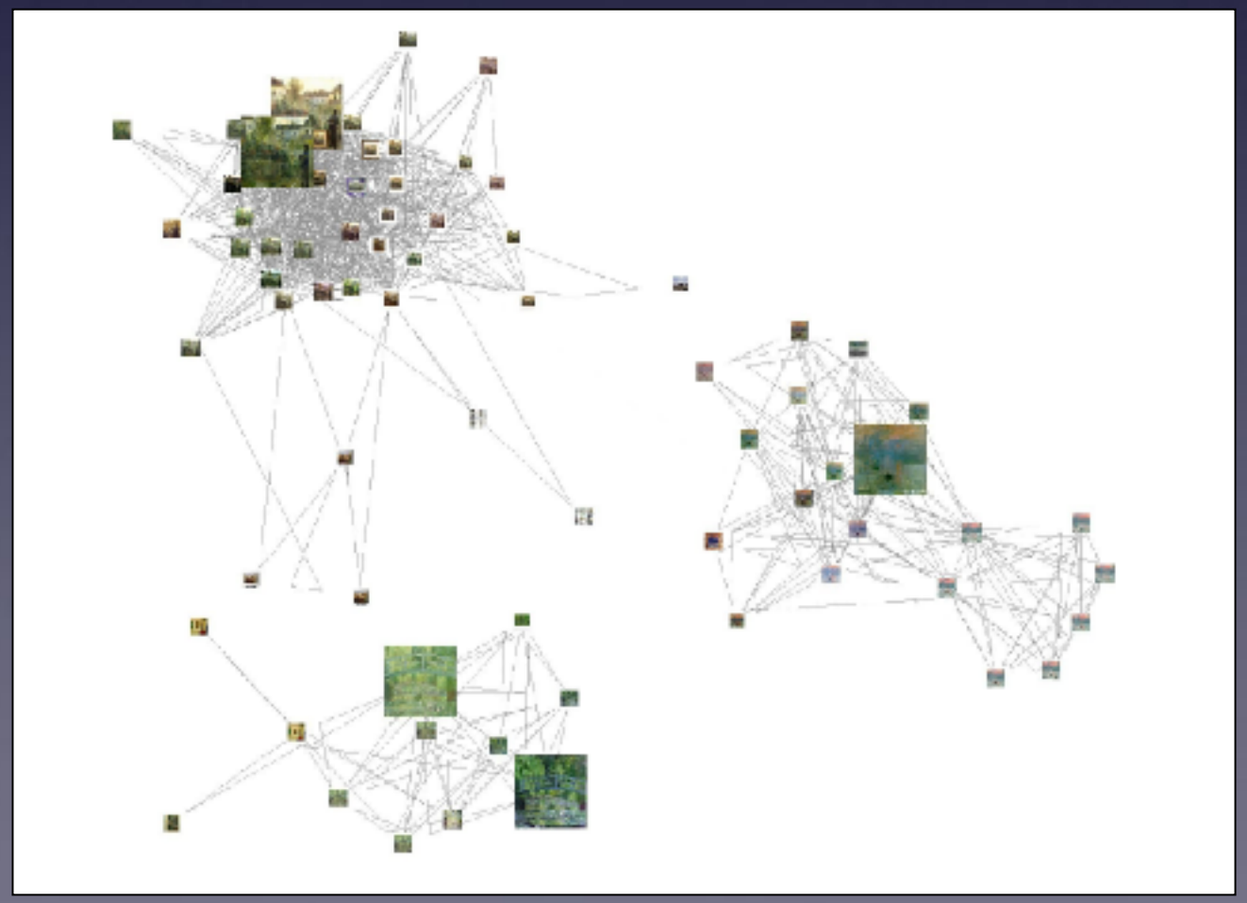
- Normalized Cut
  - A cut in a graph that penalizes large segments
  - Fix by normalizing for size of segments

$$\text{Normalized Cut}(A, B) = \frac{\text{cut}(A, B)}{\text{volume}(A)} + \frac{\text{cut}(A, B)}{\text{volume}(B)}$$

volume(A) = sum of costs of all edges that touch A

Visual Page Rank

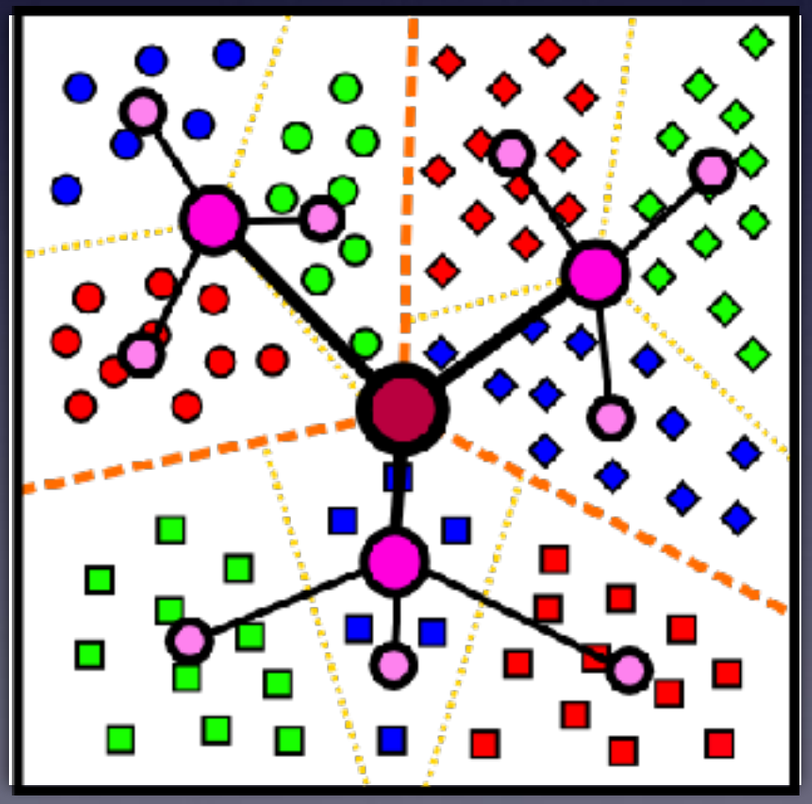
- Determining importance by random walk
- What's the probability of visiting a given node?
  - Create adjacency matrix based on visual similarity
  - Edge weights determine probability of transition



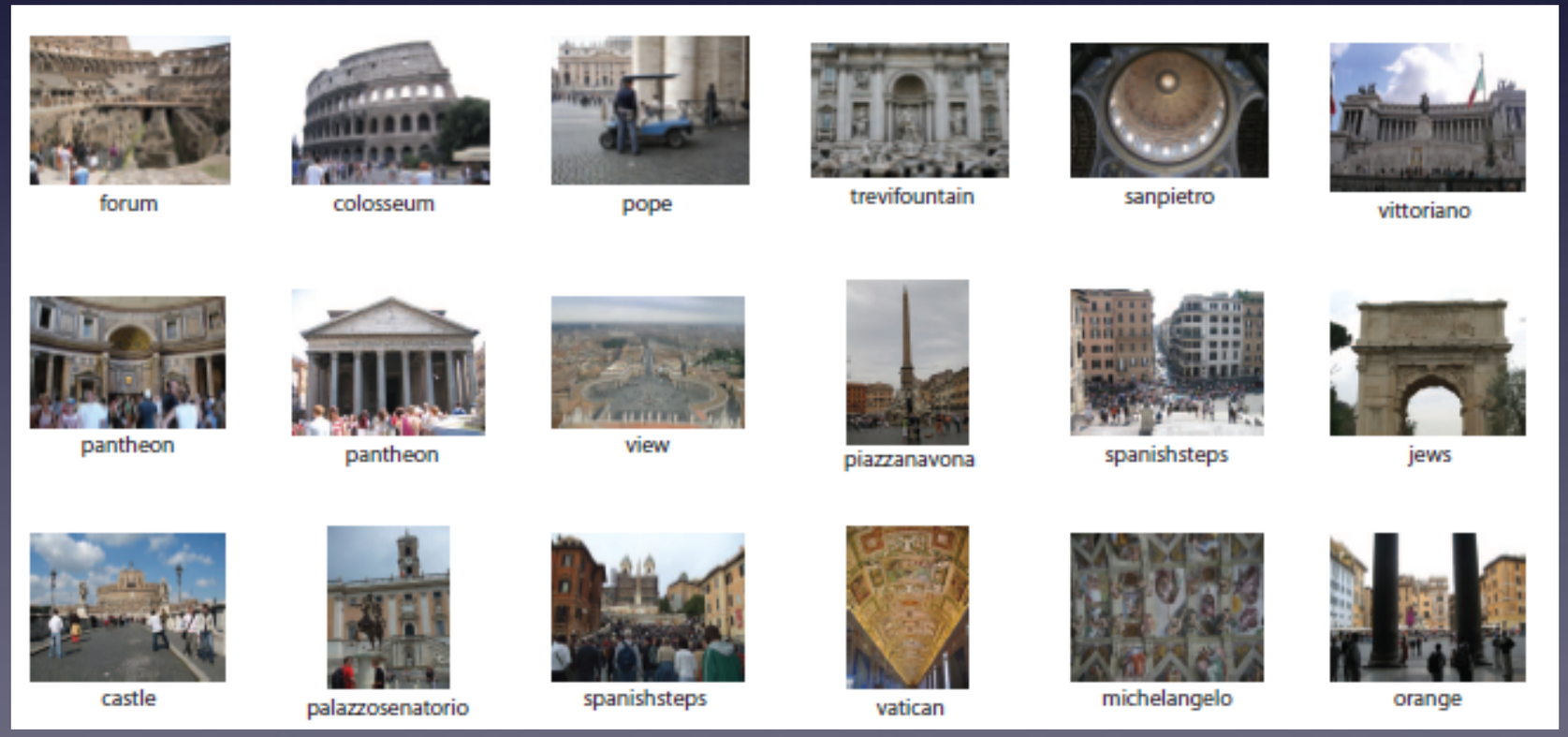
Jing Baluja 2008

Which Clustering Algorithm to use?

- Quantization/Summarization: K-means
  - aims to preserve variance of original data
  - can easily assign new point to a cluster



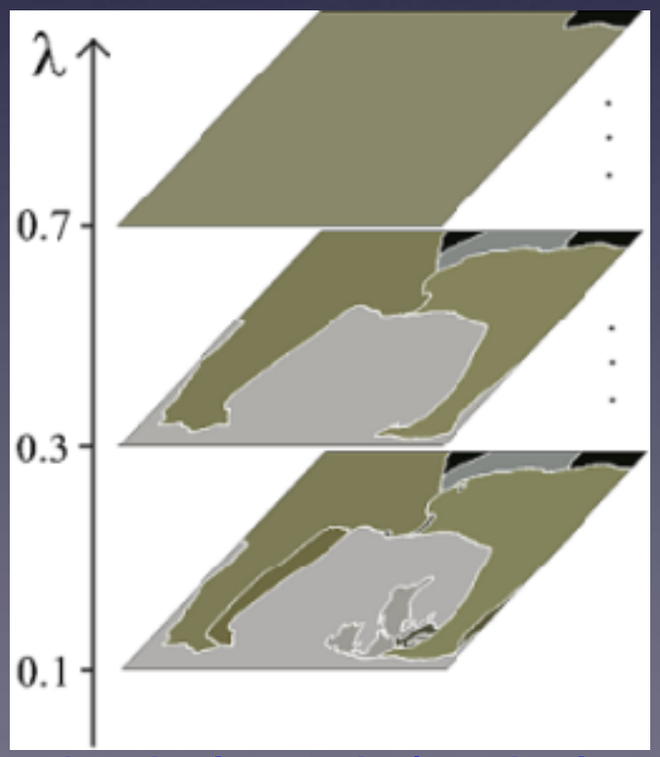
Quantization for computing histograms

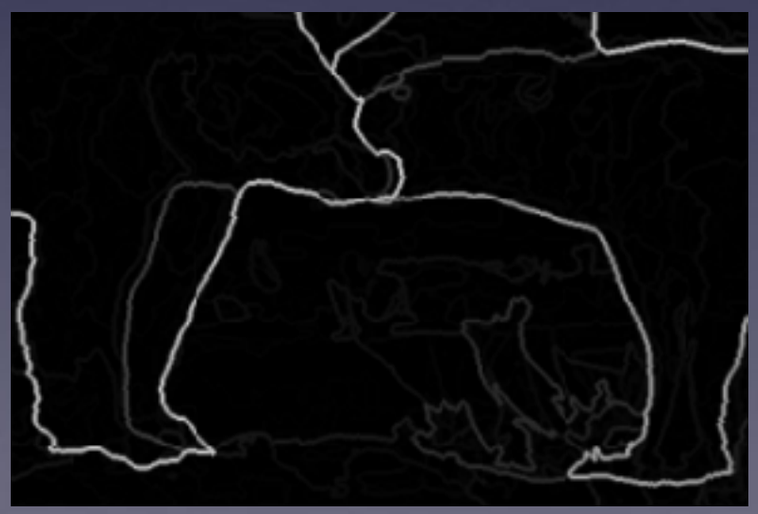


Summary of 20,000 photos of Rome using “greedy k-means”
http://grail.cs.washington.edu/projects/canonview/

Which Clustering Algorithm to use?

- Image segmentation: agglomerative clustering
  - More flexible with distance measures (e.g., can be based on boundry prediction)
  - adapts better to specific data
- hierarchy can be useful





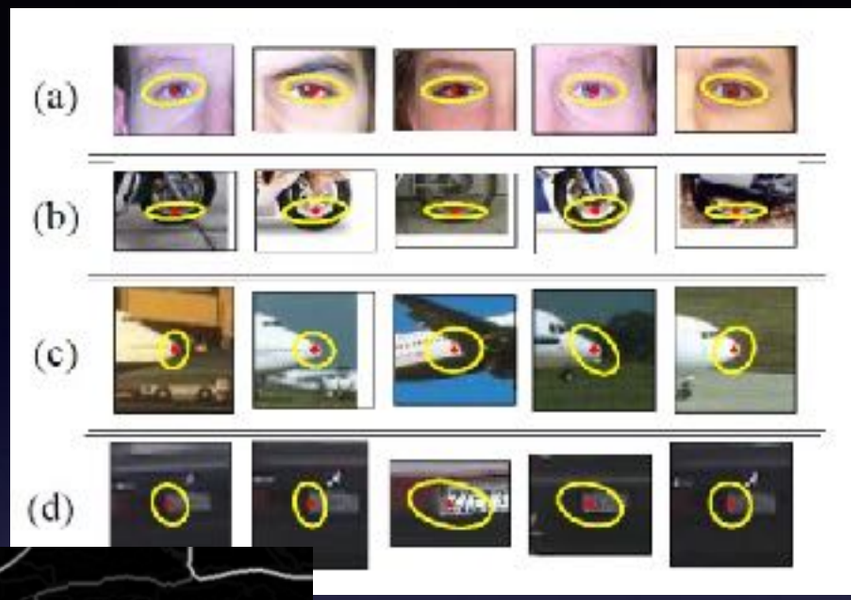
http://www.cs.berkeley.edu/~arbelaez/UCM.html

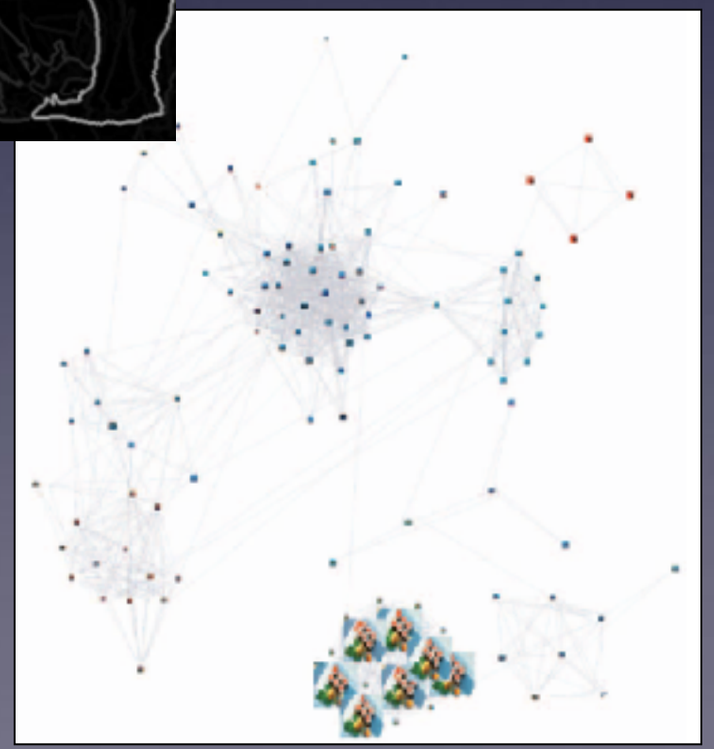
Which Clustering Algorithm to use?

- K-means useful for summarization, building dictionaries of patches, general clustering.

- Agglomerative clustering useful for segmentation, general clustering.

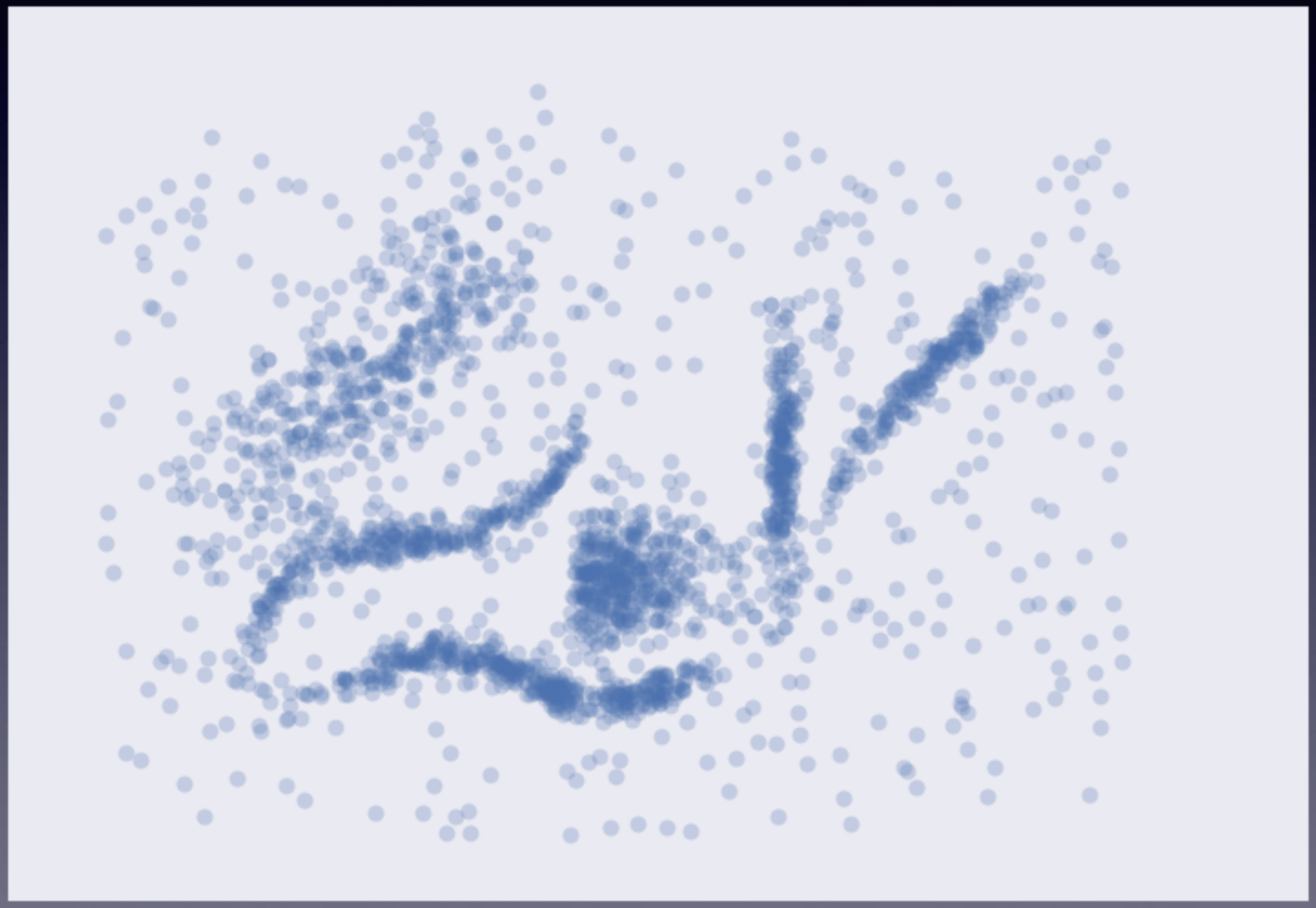
- Spectral clustering useful for determining relevance, summarization, segmentation.





Clustering algo. compared

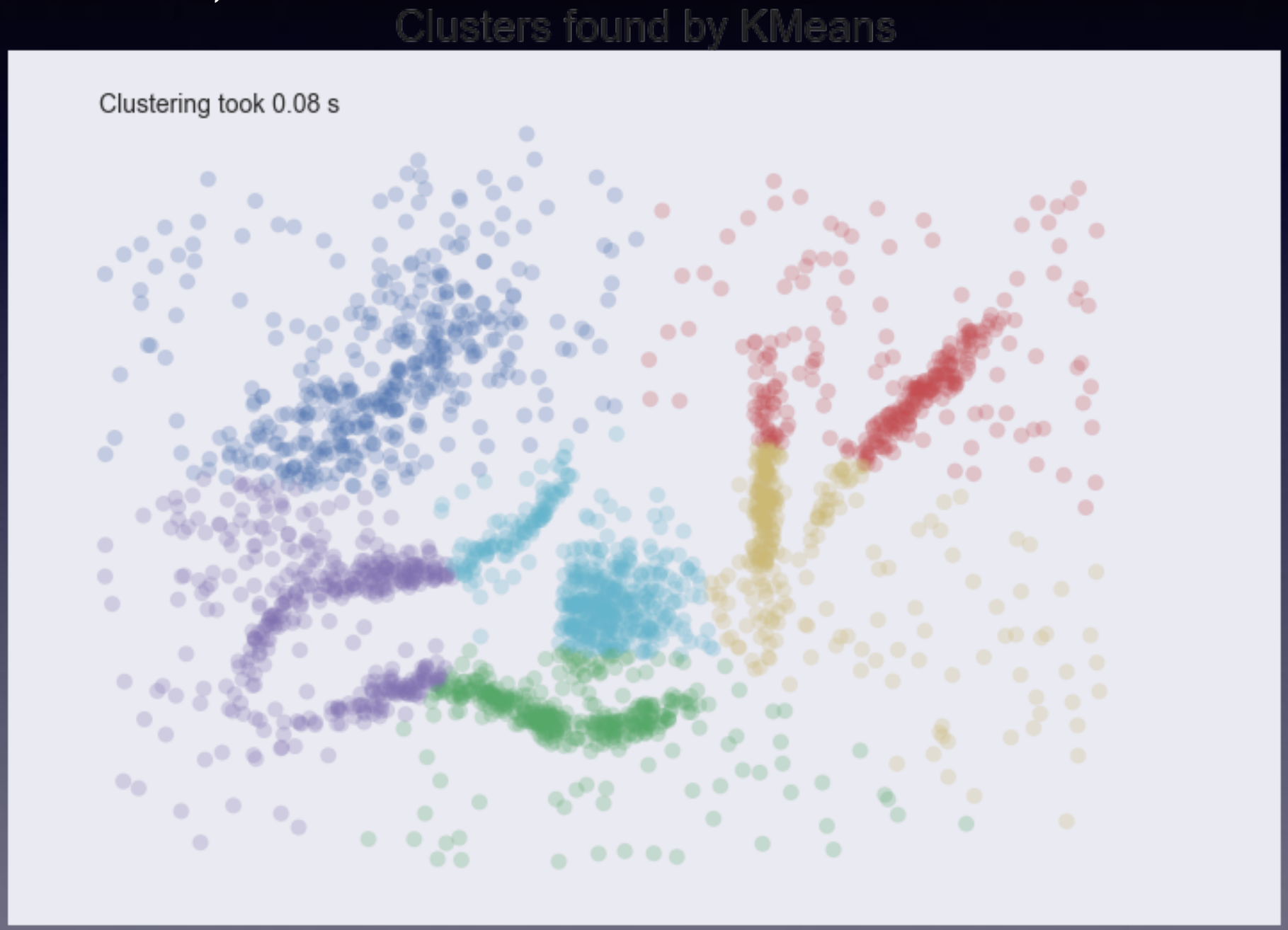
- Synthetic dataset



http://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html

Clustering algo. compared

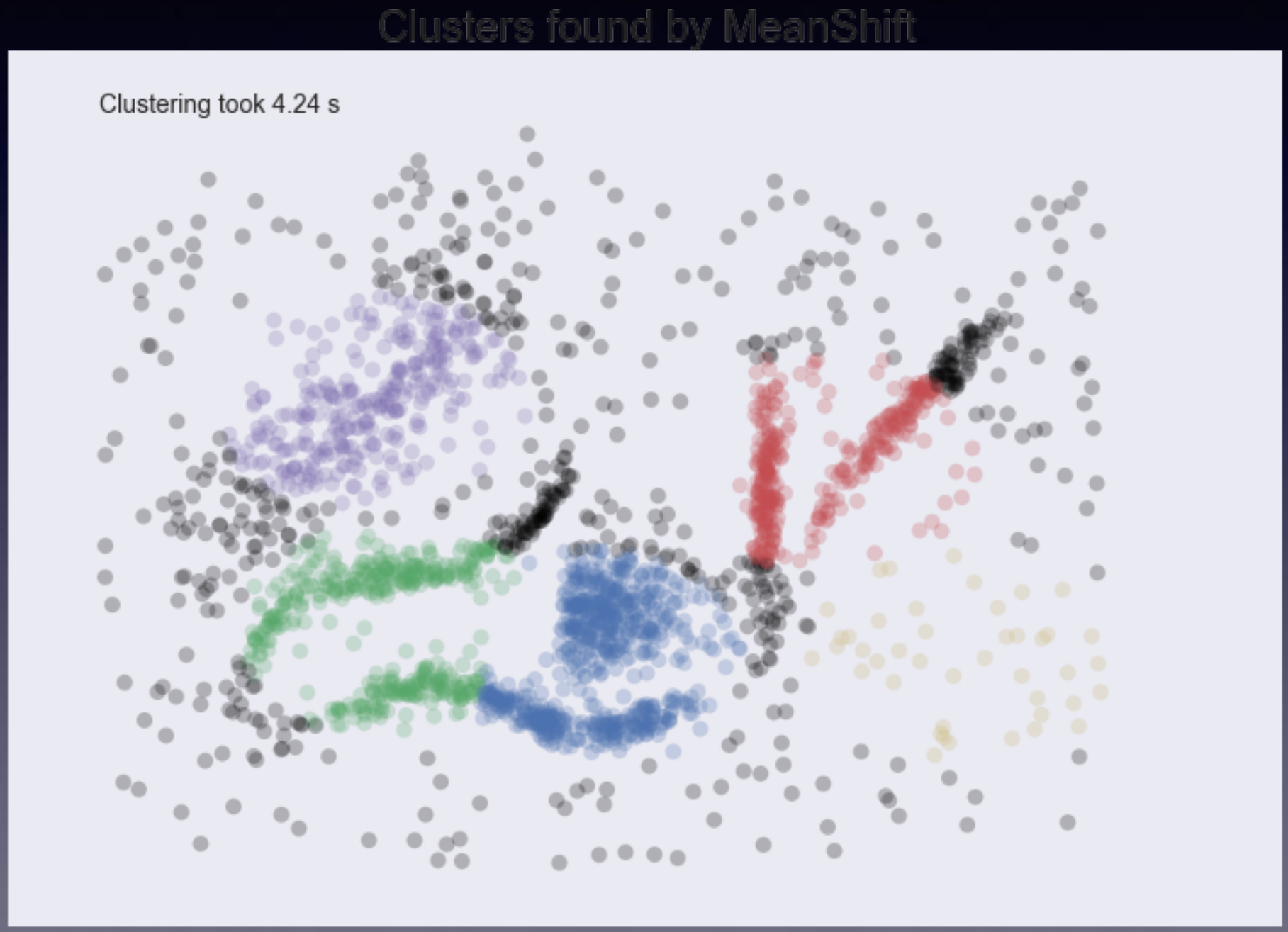
- K-means, k=6



http://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html

Clustering algo. compared

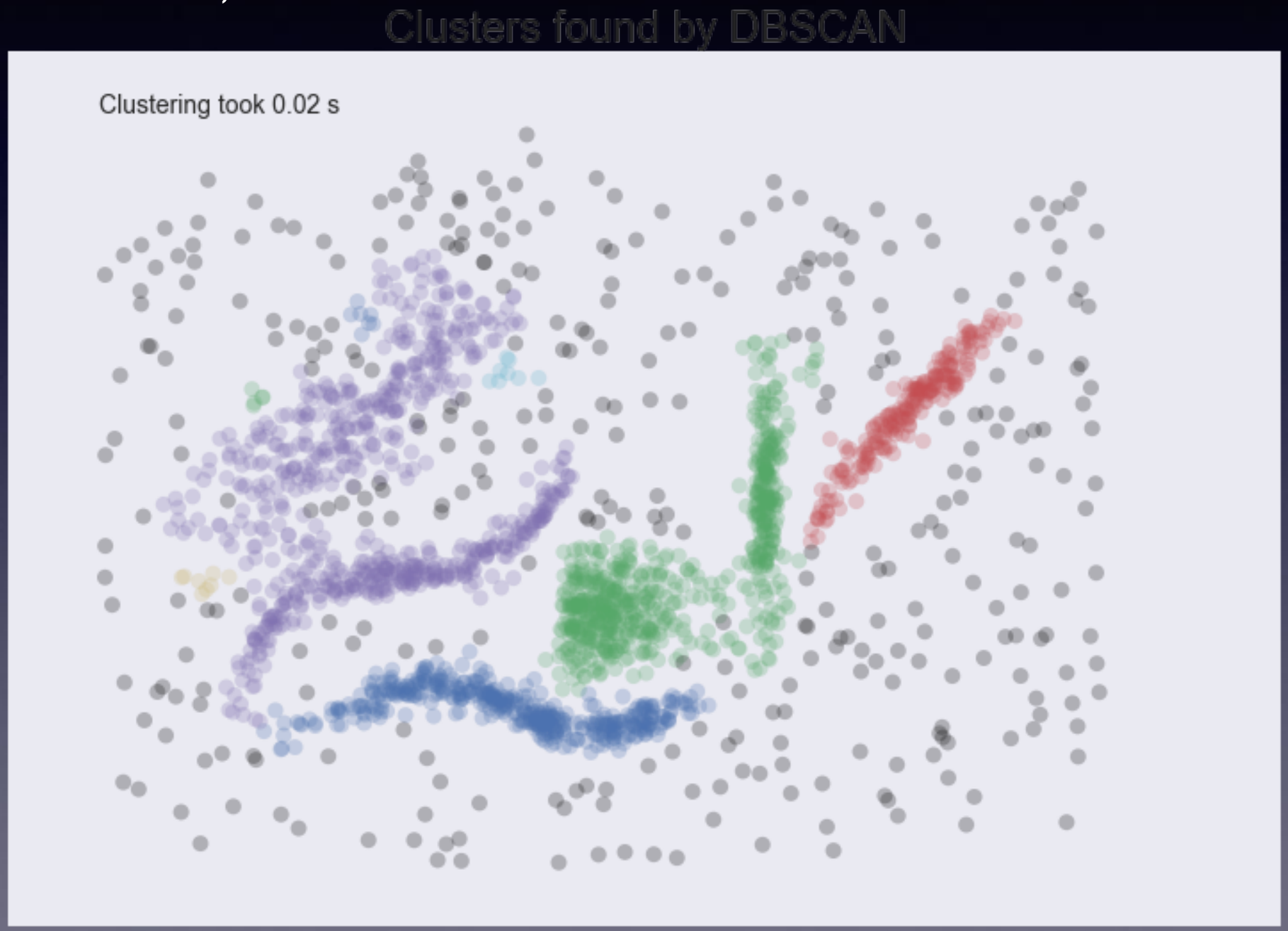
- Meanshift



http://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html

Clustering algo. compared

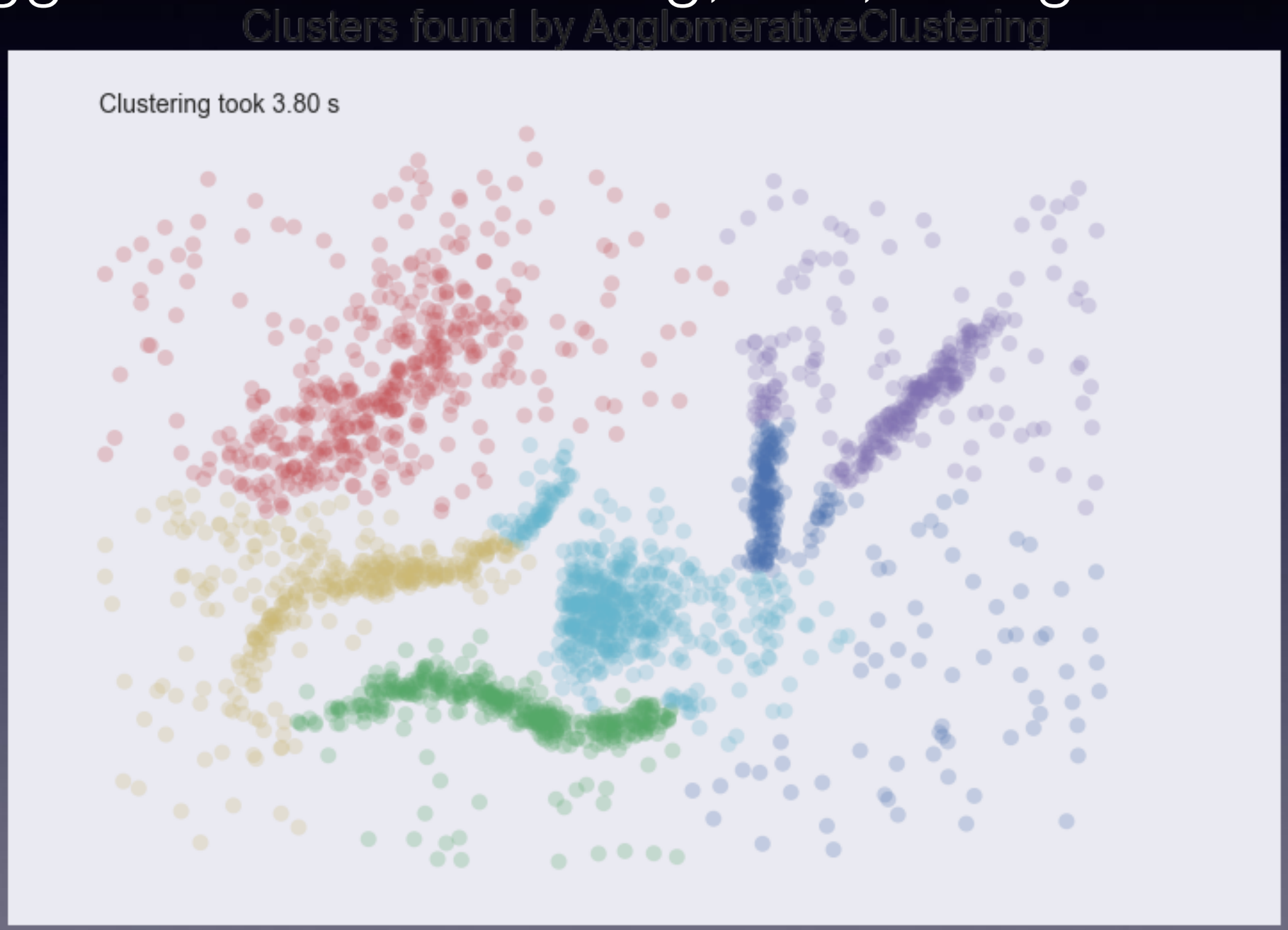
- DBSCAN, ε=0.025



http://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html

Clustering algo. compared

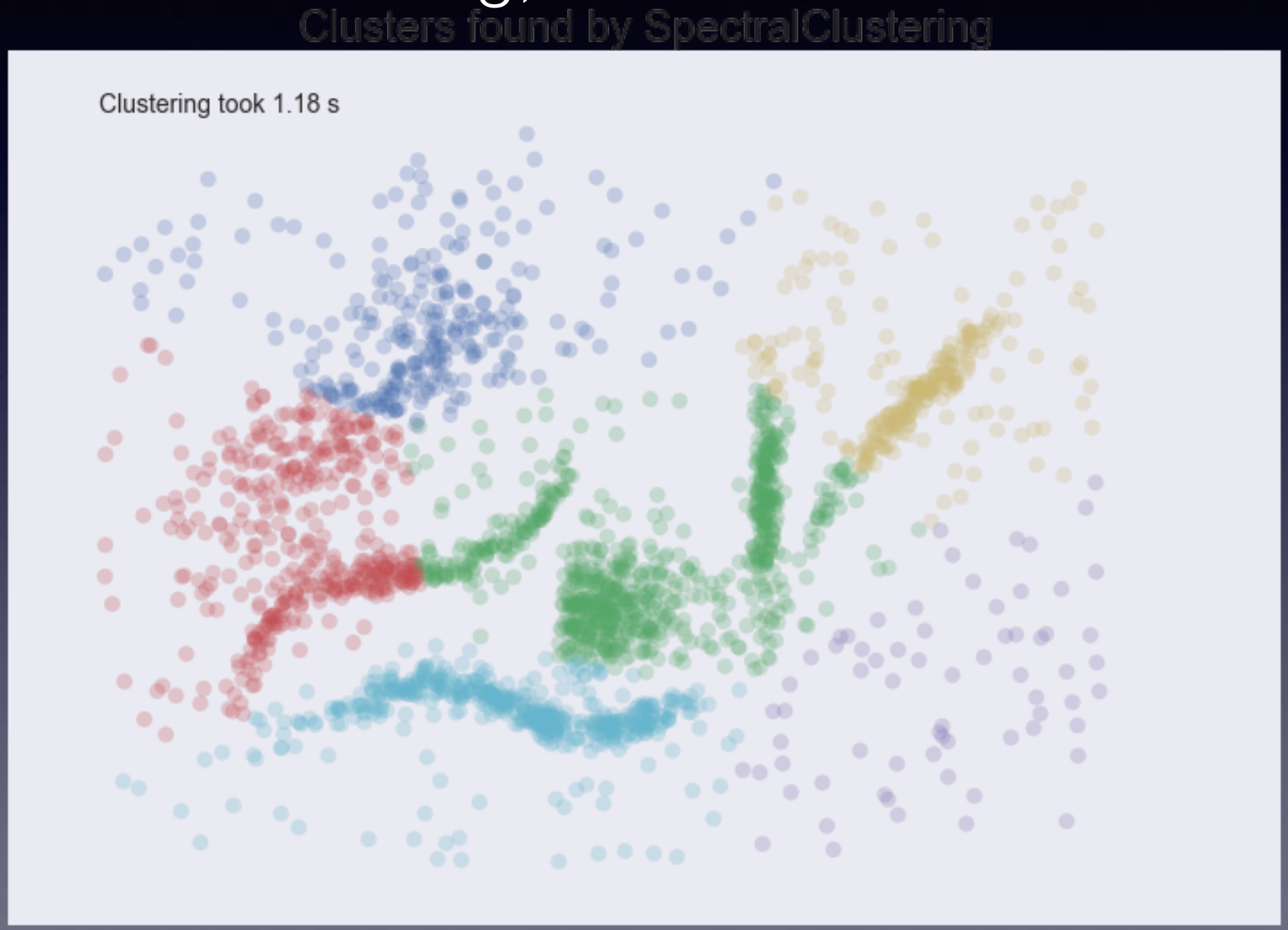
- Agglomerative Clustering, k=6, linkage=ward



http://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html

Clustering algo. compared

- Spectral Clustering, k=6



http://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html

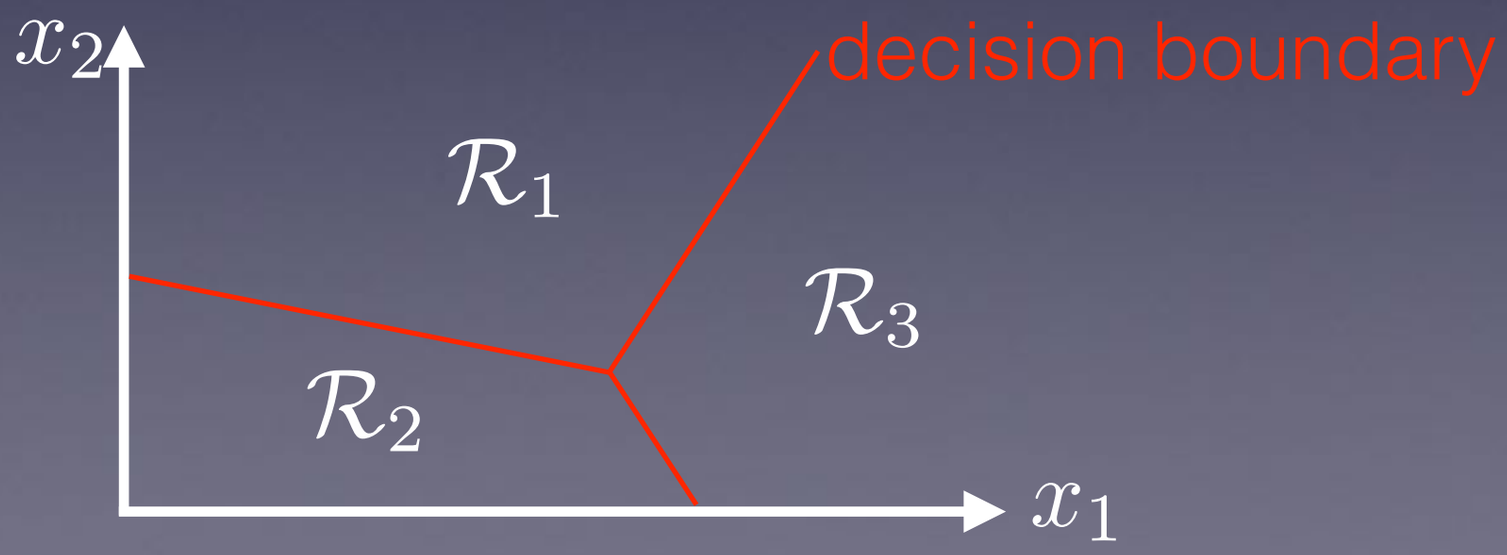
Classification

Machine Learning Roadmap

supervised / unsupervised	continuous (predicting a quantity)	discrete (predicting a category)
unsupervised	**Dimension Reduction**	**Clustering**
supervised	**Regression**	**Classification**

Supervised Learning

- Given a set of samples $\mathbf{x}_i \in \mathcal{X}$ and their ground truth annotation y_i , learn a function $y = f(\mathbf{x})$ that minimizes the prediction error $E(y_j, f(\mathbf{x}_j))$ for new $\mathbf{x}_j \notin \mathcal{X}$.
- The function $y = f(\mathbf{x})$ is a classifier. Classifiers divides input space into *decision regions* separated by *decision boundaries*.

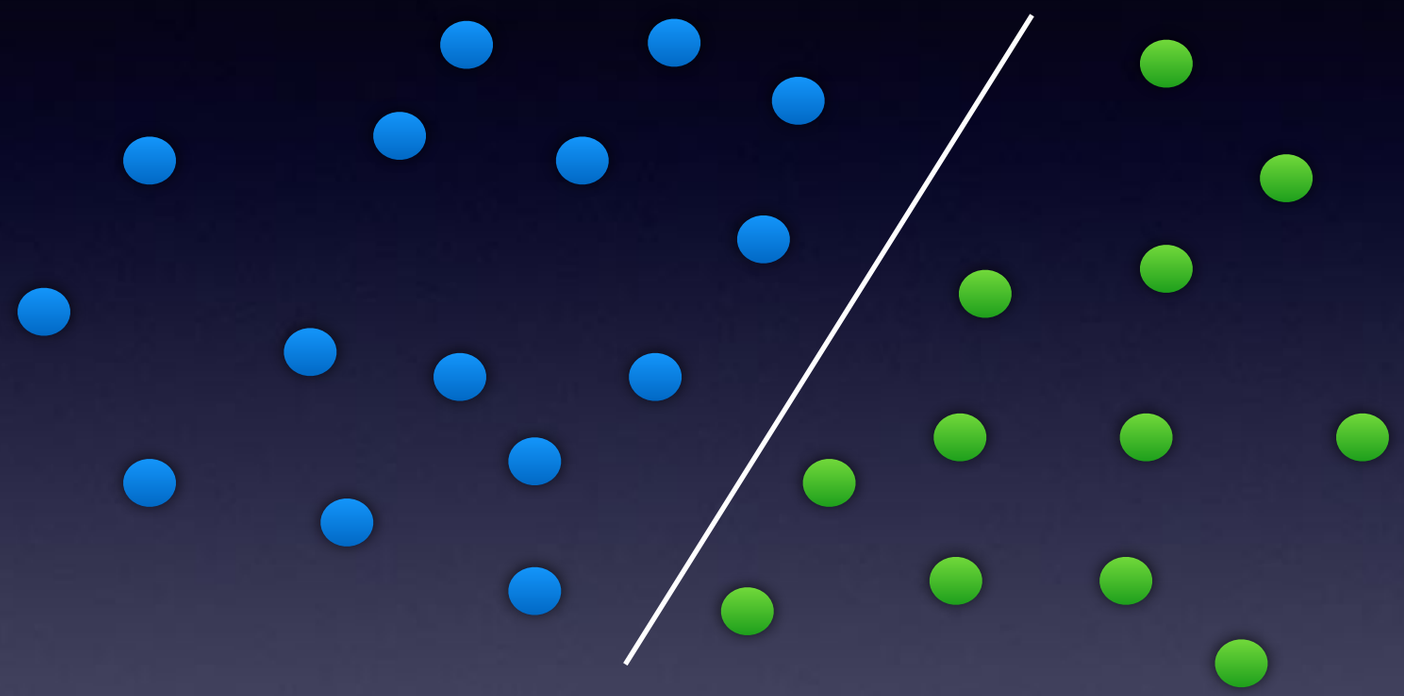


Supervised Learning Examples

- Spam detection:
  - X = { characters and words in the email }
  - Y = { spam, not spam}
- Digit recognition: 
  - X = cut out, normalized images of digits
  - Y = {0,1,2,3,4,5,6,7,8,9}
- Medical diagnosis
  - X = set of all symptoms
  - Y = set of all diseases

Linear Classifiers

- Find a linear function to separate the classes



- Logistic Regression
- Naïve Bayes
- Linear SVM

Naïve Bayes Classifier

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}, \text{ where } p(X) = \sum_{Y} p(X|Y)p(Y)$$

- Using a probabilistic approach to model data, the distribution of P(**X**, *Y*): given data **X**, find the Y that maximizes the posterior probability p(*Y*|**X**).
- Problem: we need to model all p(**X**|*Y*) and p(*Y*). If | **X** | = n, there are 2^n possible values for **X**.
- The Naïve Bayes' assumption assumes that xi's are conditionally independent.

$$p(X_1 \dots X_n|Y) = \prod_i p(X_i|Y)$$

Naïve Bayes Classifier

- Given:
  - Prior $p(Y)$
  - n conditionally independent features, represented by the vector **X**, given the class Y
    - For each X_i, we have likelihood $p(X_i \mid Y)$
- Decision rule:

$$Y^* = \operatorname*{argmax}_Y p(Y)p(X_1, \ldots, X_n|Y)$$

$$= \operatorname*{argmax}_Y p(Y) \prod_i p(X_i|Y)$$

Maximum Likelihood for Naïve Bayes

- For discrete Naïve Bayes, simply count:
- Prior:

$$p(Y = y') = \frac{\text{Count}(Y = y')}{\sum_y \text{Count}(Y = y)}$$

- Likelihood:

$$p(X_i = x'|Y = y') = \frac{\text{Count}(X_i = x', Y = y')}{\sum_x \text{Count}(X_i = x, Y = y)}$$

- Naïve Bayes Model:

$$p(Y|\mathbf{X}) \propto p(Y) \prod_{i,j} p(\mathbf{X}|Y)$$

Naïve Bayes Classifier

- Conditional probability model over:

$$p(C_k|x_1,\ldots,x_n) = \frac{1}{Z}p(C_k)\prod_{i=1}^{n} p(x_i|C_k)$$

- Classifier:

$$\tilde{y} = \underset{k\in\{1,\ldots,K\}}{\operatorname{argmax}}\ p(C_k)\prod_{i=1}^{n} p(x_i|C_k)$$

Naïve Bayes for Text Classification

- Features **X** are entire document. X_i for i^{th} word in article. X is huge! NB assumption helps a lot!

Article from rec.sport.hockey

```
Path: cantaloupe.srv.cs.cmu.edu!das-news.harvard.e
From: xxx@yyy.zzz.edu (John Doe)
Subject: Re: This year's biggest and worst (opinio
Date: 5 Apr 93 09:53:39 GMT

I can only comment on the Kings, but the most
obvious candidate for pleasant surprise is Alex
Zhitnik. He came highly touted as a defensive
defenseman, but he's clearly much more than that.
Great skater and hard shot (though wish he were
more accurate). In fact, he pretty much allowed
```

Naïve Bayes for Text Classification

- Typical additional assumption: X_i's position in document doesn't matter: *bag of words*.

aardvark	0
about	2
all	2
Africa	1
apple	0
...	
gas	1
...	
oil	1
...	
Zaire	0

Naïve Bayes for Text Classification

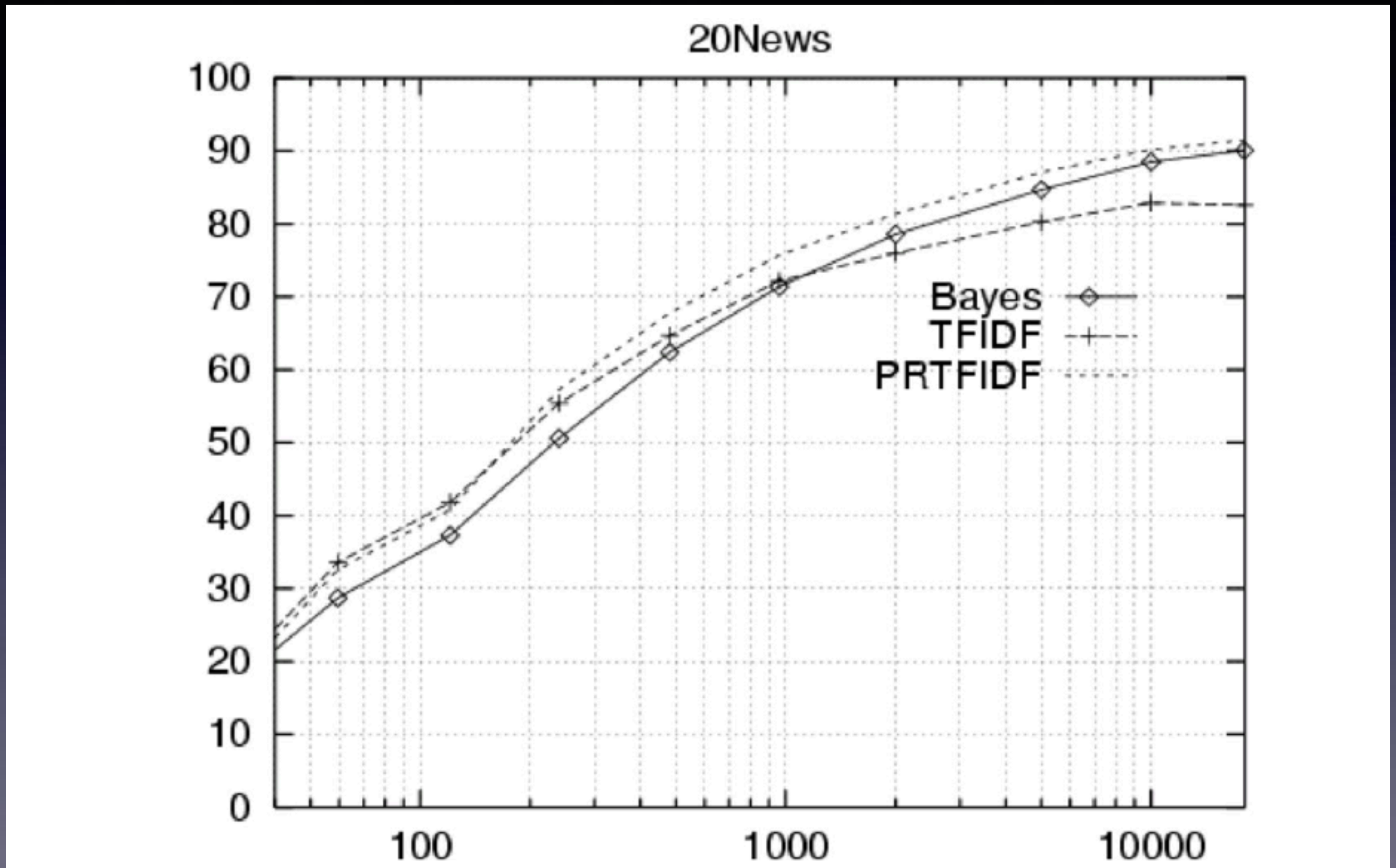
- Learning Phase:
  - Prior: p(Y), count how many documents in each topic (prior).
  - Likelihood: p(X_i|Y), for each topic, count how many times a word appears in documents of this topic.
- Testing Phase: for each document, use Naïve Bayes' decision rule:

$$\underset{y}{\operatorname{argmax}}\ p(y) \prod_{i=1}^{\text{words}} p(x_i|y)$$

Naïve Bayes for Text Classification

- Given 1000 training documents from each group, learn to classify new documents according to which newsgroup it came from.
  - comp.graphics,
  - comp.os.ms-windows.misc
  - ...
  - soc.religion.christian
  - talk.religion.misc
  - ...
  - misc.forsale
  - ...

Naïve Bayes for Text Classification



Accuracy vs. Training set size (1/3 withheld for test)

Naïve Bayes Classifier Issues

- Usually, features are not conditionally independent:

$$p(X1, \ldots, X_n|Y) \neq \prod_i p(X_i|Y)$$

- Actual probabilities p(*Y*|**X**) often bias towards 0 or 1
- Nonetheless, Naïve Bayes is the single most used classifier.
  - Naïve Bayes performs well, even when assumptions are violated.
- Know its assumptions and when to use it.

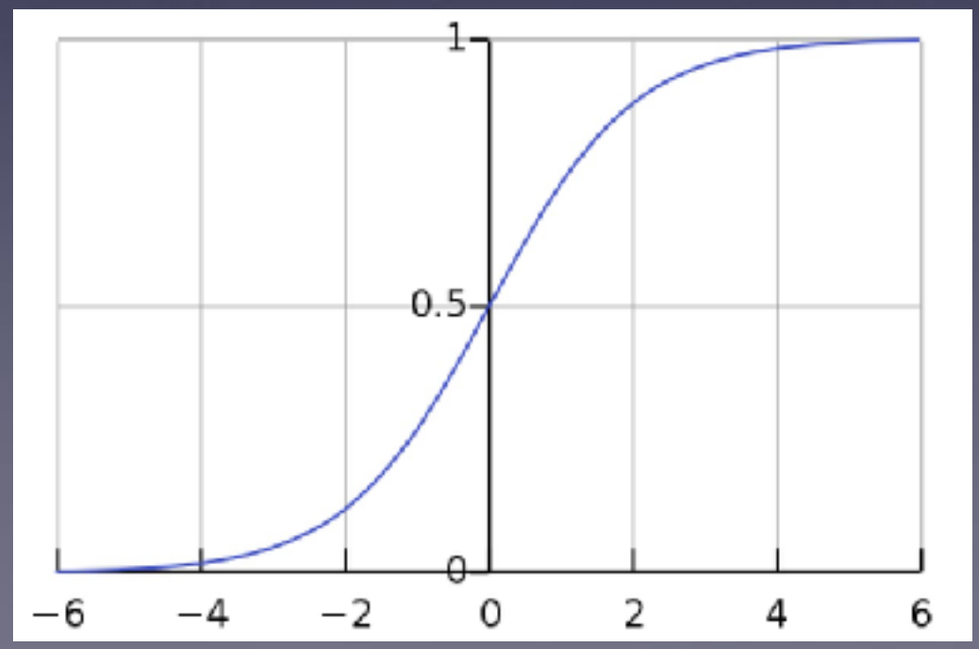
Logistic Regression

- Regression model for which the dependent variable is categorical.
  - Binomial/Binary Logistic Regression
  - Multinomial Logistic Regression
  - Ordinal Logistic Regression (categorical, but ordered)
- Substituting Logistic Function

$$f(\tilde{x}) = \frac{1}{1 + e^{-\tilde{x}}}, \tilde{x} = w_0 + w_1 x$$

we get:

$$y(x, \mathbf{w}) = \frac{1}{1 + e^{-(w_0 + w_1 x)}}$$

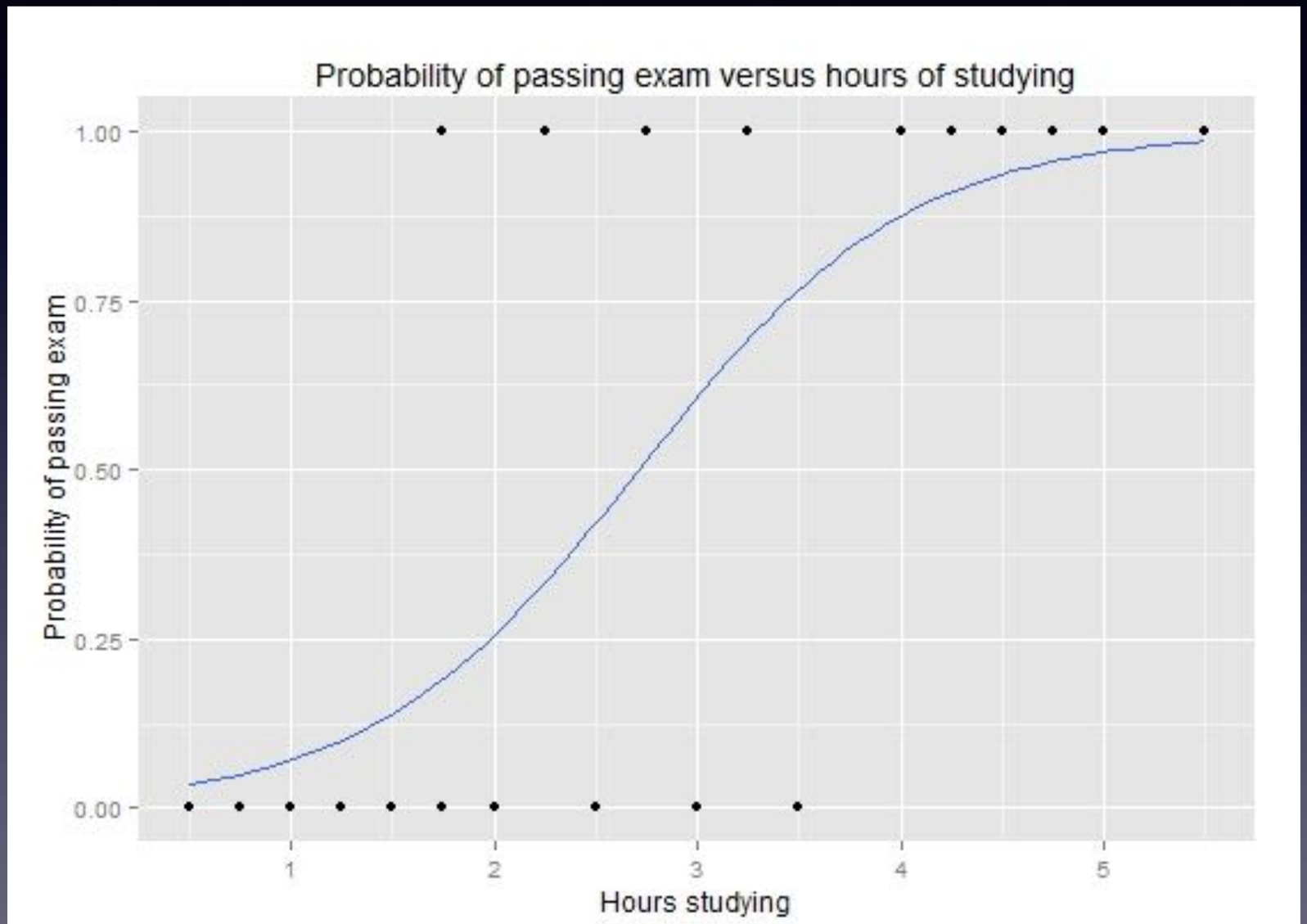


When to use logistic regression?

- E.g., for predicting:
  - mortality of injured patients,
  - risk of developing a certain disease based on observations of the patient,
  - whether an American voter would vote Democratic or Republican,
  - probability of failure of a given process, system or product,
  - customer's propensity to purchase a product or halt a subscription,
  - likelihood of homeowner defaulting on mortgage.

Logistic Regression Example

- Hours studied vs passing the exam



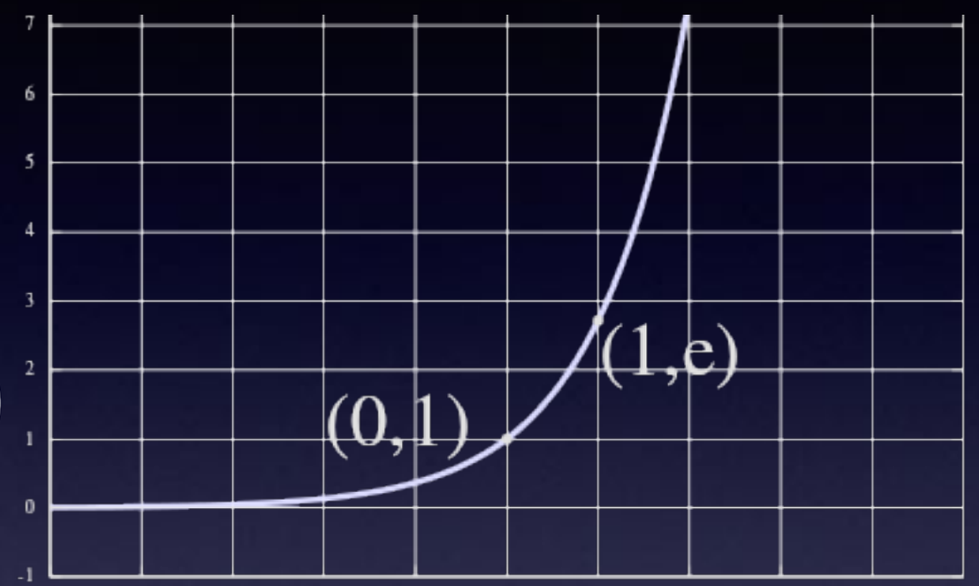
$$P_{\text{pass}}(h) = \frac{1}{1 + e^{-(-4.0777 + 1.5046 \cdot h)}}$$

Logistic Regression Classifier

- Learn p(Y|**X**) directly. Reuse ideas from regression, but let y-intercept define the probability.

$$p(Y = 1|\mathbf{X}, \mathbf{w}) \propto \exp(w_0 + \sum_i w_i X_i)$$

Exponential function

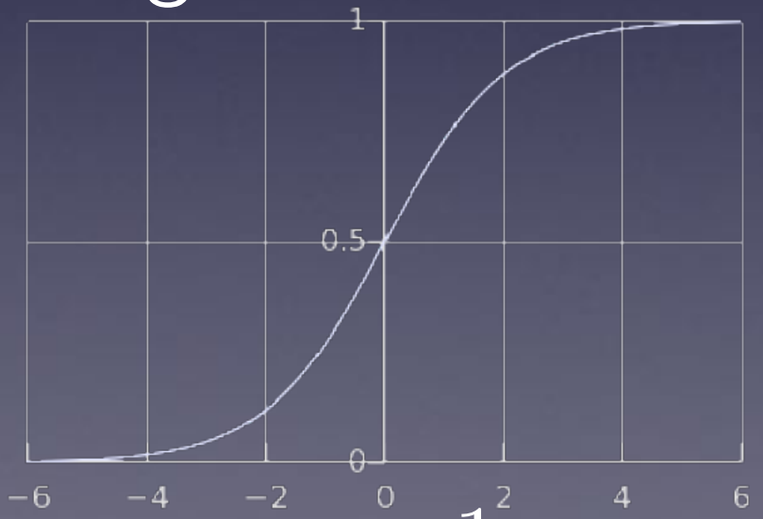


- With normalization

$$p(Y = 0|\mathbf{X}, \mathbf{w}) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

$$p(Y = 1|\mathbf{X}, \mathbf{w}) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

Logistic function



$$y = \frac{1}{1 + \exp(-x)}$$

Logistic Regression: decision boundary

$$p(Y=0|\mathbf{X},\mathbf{w}) = \frac{1}{1+\exp(w_0+\sum_i w_i X_i)}$$

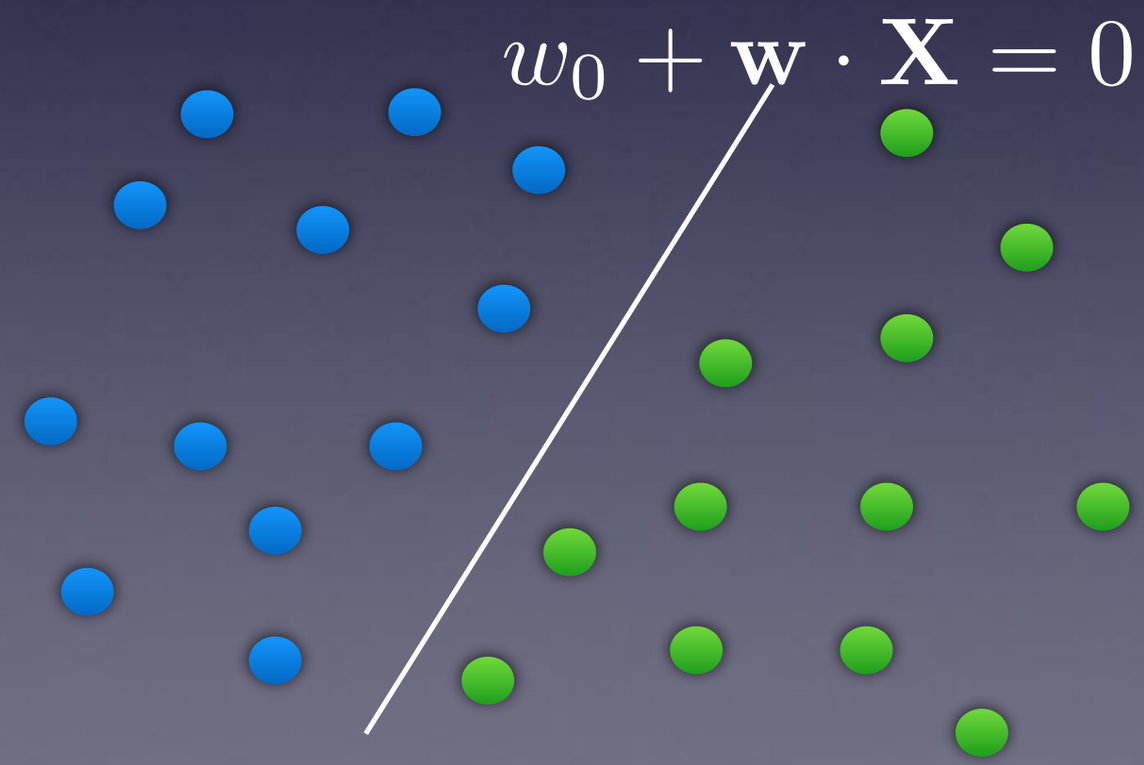
$$p(Y=1|\mathbf{X},\mathbf{w}) = \frac{\exp(w_0+\sum_i w_i X_i)}{1+\exp(w_0+\sum_i w_i X_i)}$$

- Prediction: output the *Y* with highest p(*Y*|**X**). For binary Y, output Y if

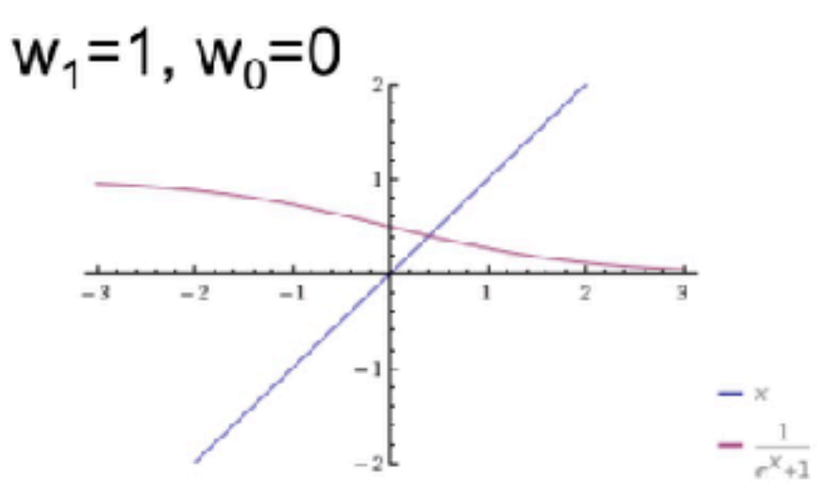
$$1 < \frac{P(Y=1|X)}{P(Y=0|X)}$$

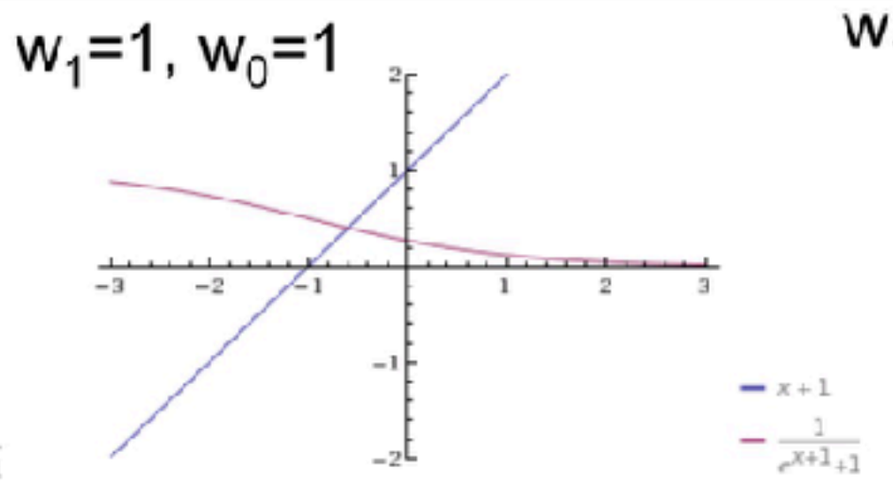
$$1 < \exp(w_0 + \sum_{i=1}^{n} w_i X_i)$$

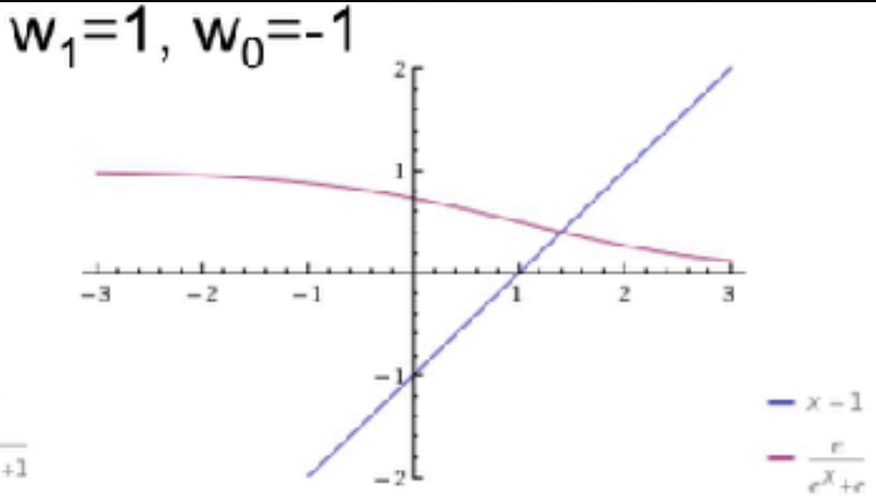
$$0 < w_0 + \sum_{i=1}^{n} w_i X_i$$

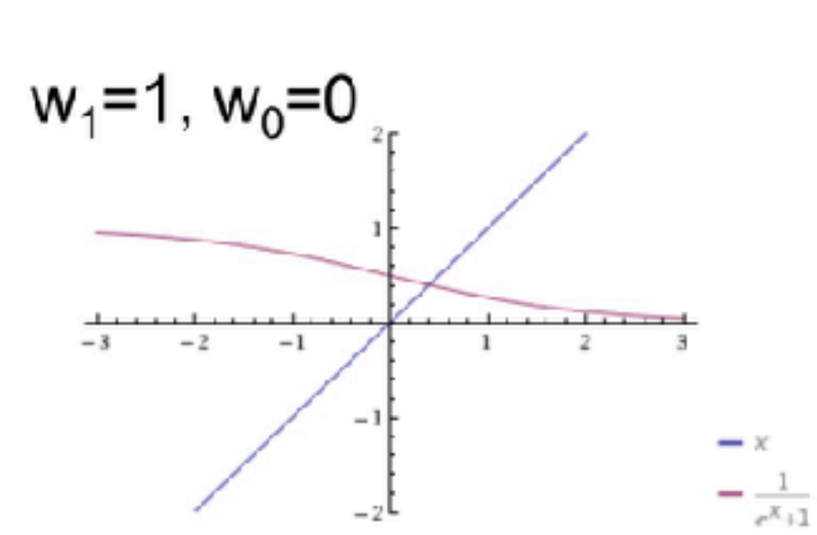


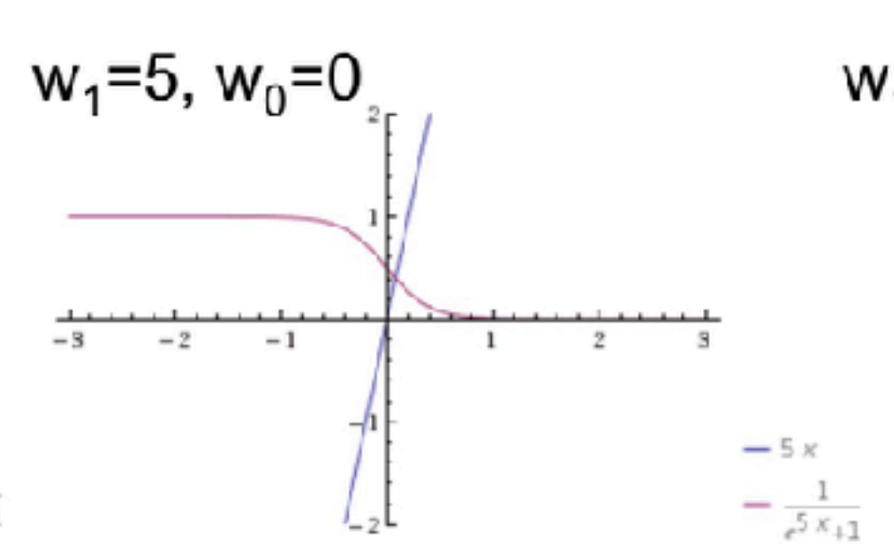
Visualizing $p(Y = 0|\mathbf{X}, \mathbf{w}) = \frac{1}{1 + \exp(w_0 + w_1 x_1)}$

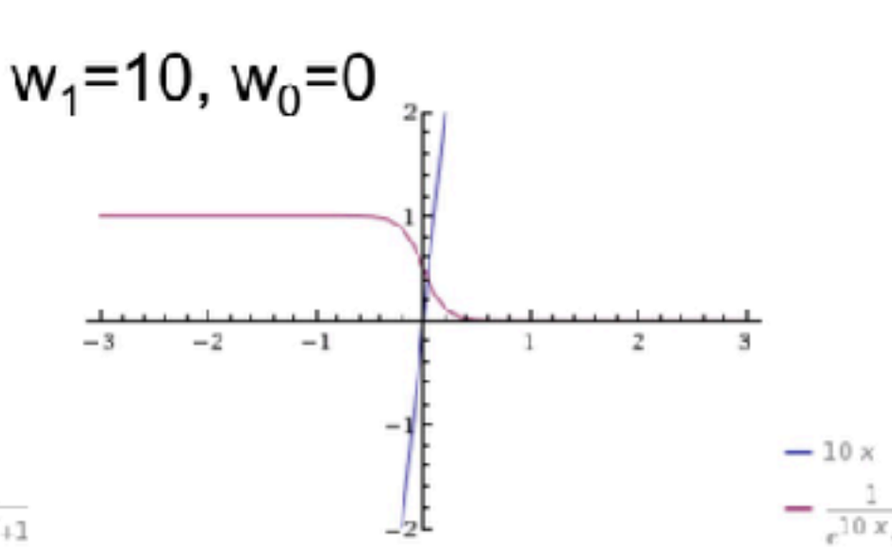








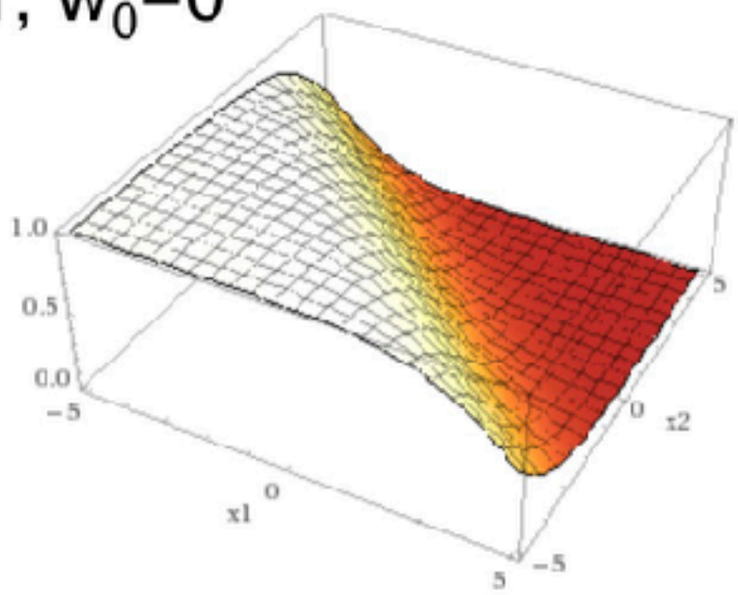




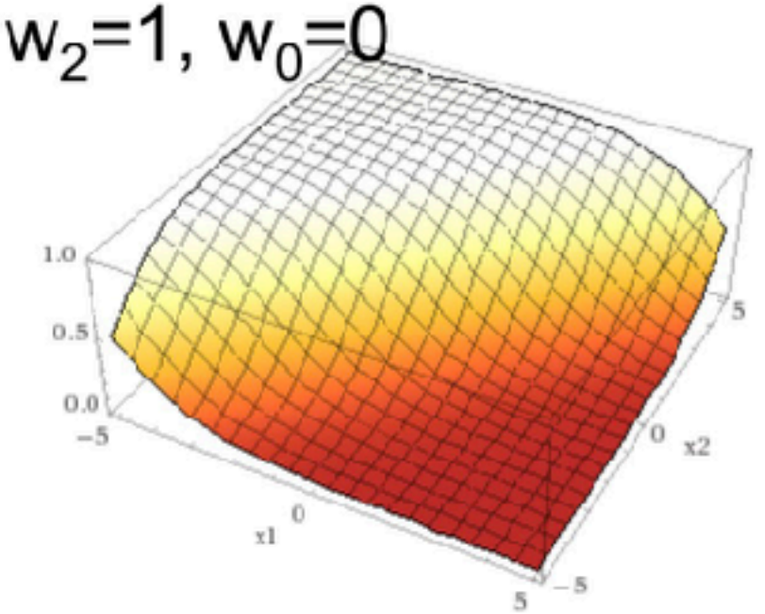
- Decision boundary: p(*Y*=0 | **X**, **w**) = 0.5
- Slope of the line defines how quickly probabilities go to 0 or 1 around decision boundary.

Visualizing $p(Y = 0|\mathbf{X}, \mathbf{w}) = \frac{1}{1 + \exp(w_0 + w_1 x_1 + w_2 x_2)}$

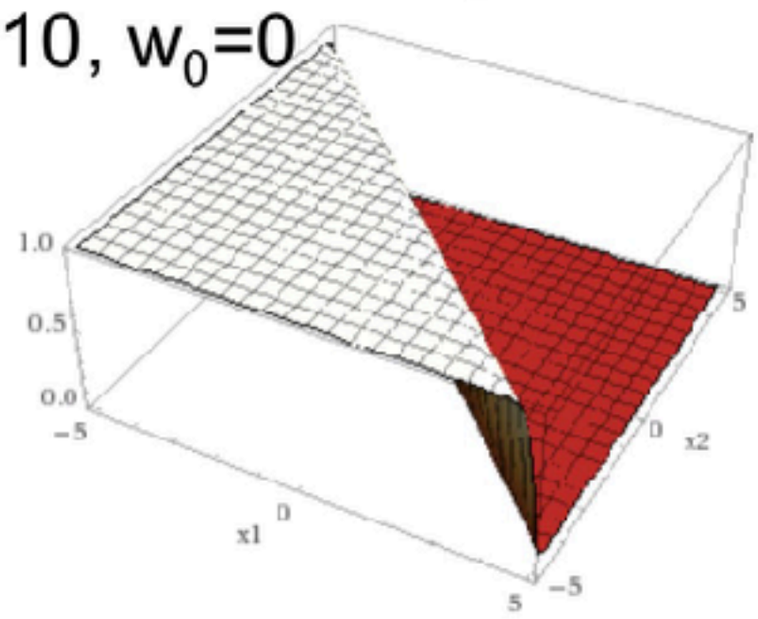
w_1=1, w_2=1, w_0=0



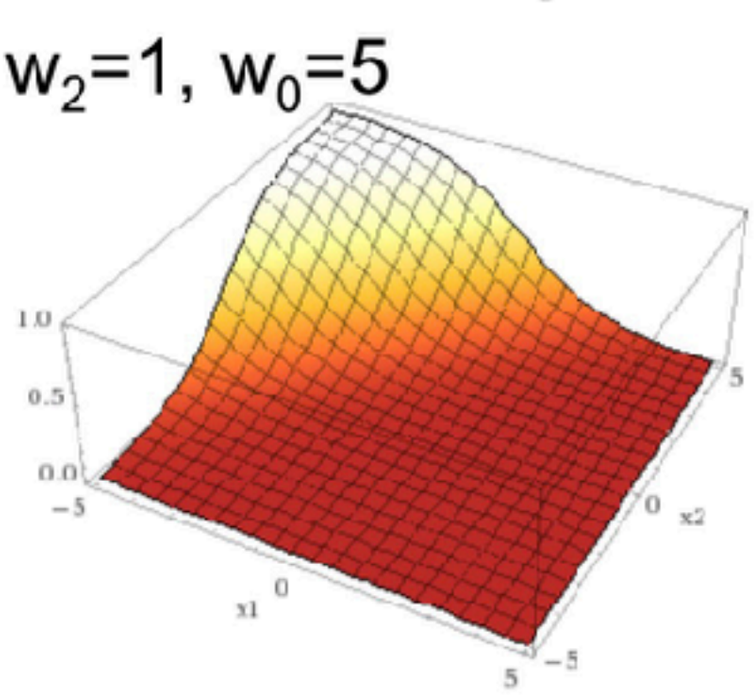
w_1=-1, w_2=1, w_0=0



w_1=10, w_2=10, w_0=0



w_1=-1, w_2=1, w_0=5



- Decision boundary is defined by y=0 hyperplane

Logistic Regression Param. Estimation

- Generative (Naïve Bayes) loss function:
  - *Data likelihood*

$$\ln\, p(\mathcal{D}|\mathbf{w}) = \sum_{j=1}^{N} \ln\, p(\mathbf{x}^j, y^j|\mathbf{w}) = \sum_{j=1}^{N} \ln\, p(y^j|\mathbf{x}^j, \mathbf{w}) + \sum_{j=1}^{N} \ln\, p(\mathbf{x}^j|\mathbf{w})$$

- Discriminative (logistic regression) loss function:
  - *Conditional Data likelihood*

$$\ln\, p(\mathcal{D}_Y|\mathcal{D}_\mathbf{X}, \mathbf{w}) = \sum_{j=1}^{N} \ln\, p(y^j|\mathbf{x}^j, \mathbf{w})$$

  - Maximize conditional log likelihood!

Logistic Regression Param. Estimation

- Maximize conditional log likelihood (*Maximum Likelihood Estimation, MLE*):

$$l(\mathbf{w}) \equiv \ln \prod_j p(y^j | \mathbf{x}^j, \mathbf{w})$$

$$= \sum_j y^j (w_0 + \sum_i w_i x_i^j) - \ln(1 + \exp(w_0 + \sum_i w_i x_i^j)$$

- No closed-form solution.
- Concave function of w → no need to worry about local optima; easy to optimize.

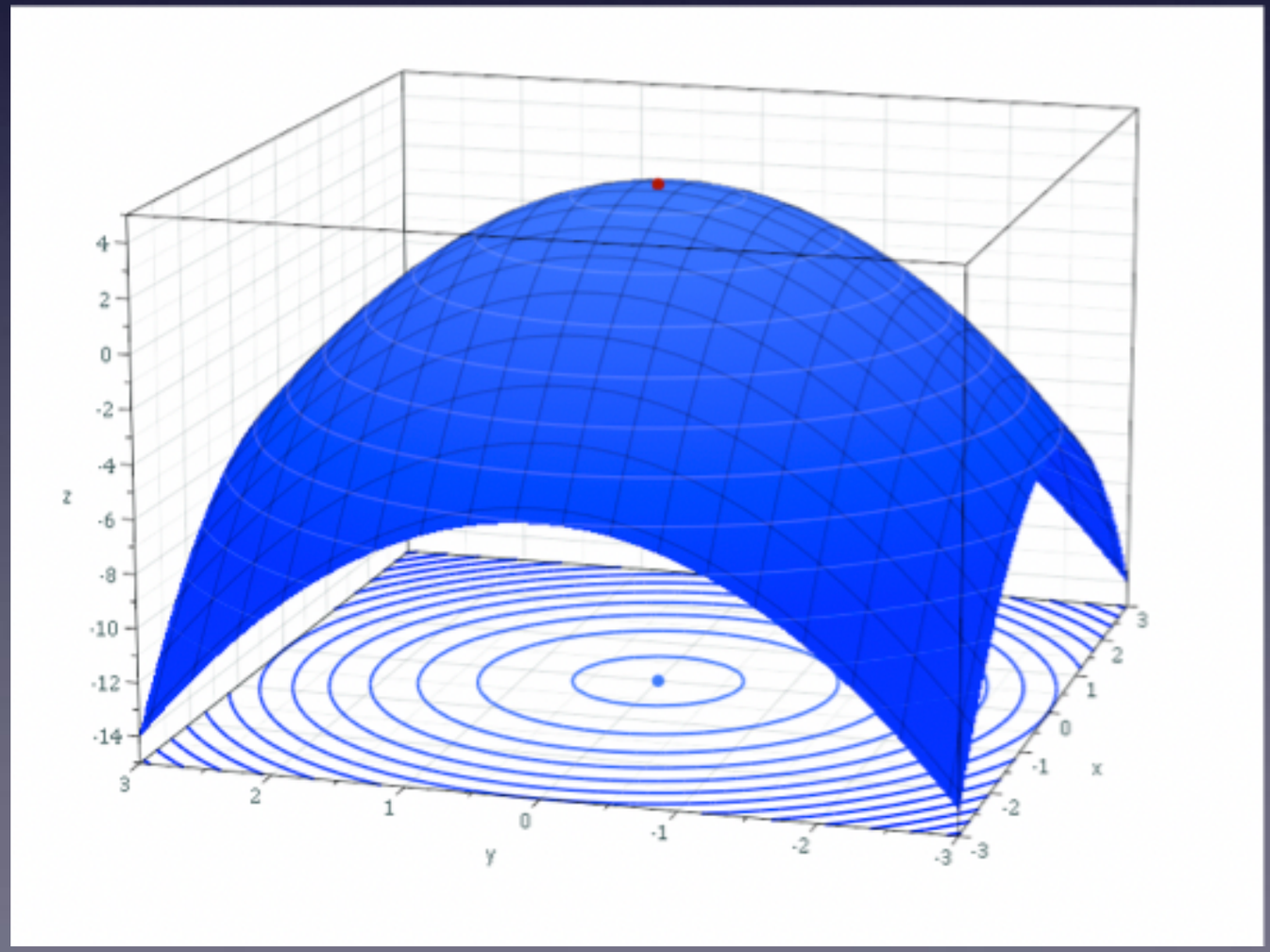
Logistic Regression Param. Estimation

- Conditional likelihood for logistic regression is convex!
- Gradient: $\nabla_{\mathbf{w}} l(\mathbf{w}) = \left[\frac{dl(\mathbf{w})}{dw_0}, \ldots, \frac{dl(\mathbf{w})}{dw_n} \right]$
- *Gradient Ascent* update rule:

$$\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w})$$

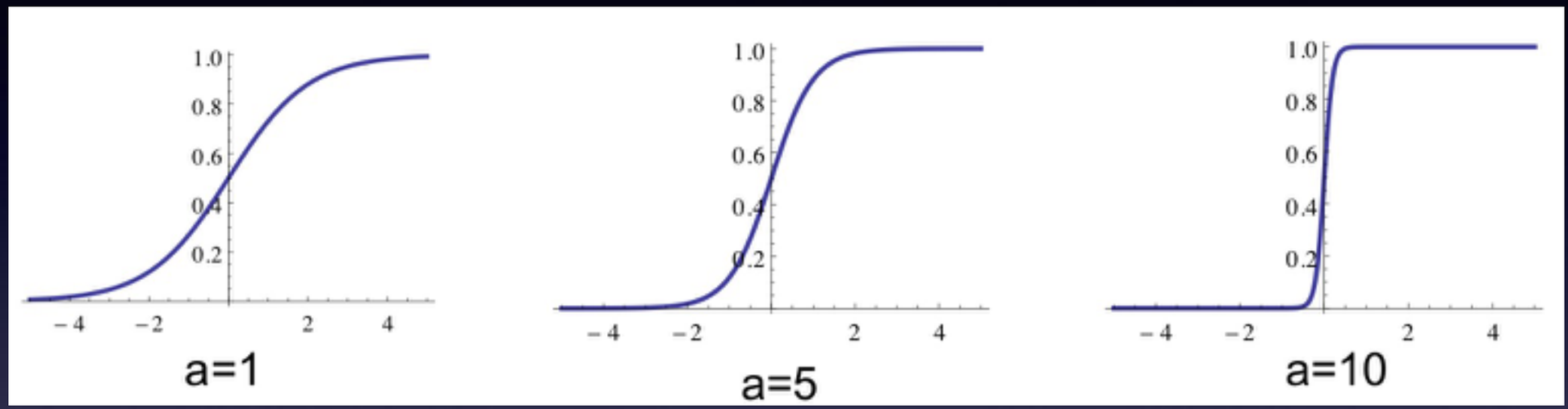
$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \frac{dl(\mathbf{w})}{dw_i}$$

- Simple, powerful, use in many places.



Logistic Regression Param. Estimation

- MLE tends to prefer large weights



- Higher likelihood of properly classified examples close to decision boundary.
- Larger influence of corresponding features on decision.
- Can cause overfitting!!!

Logistic Regression Param. Estimation

- *Regularization* to avoid large weights, overfitting.
- Add priors on **w** and formulate as *Maximum a Posteriori* (*MAP*) optimization problem.

$$p(\mathbf{w}|Y,\mathbf{X}) \propto p(Y|\mathbf{X},\mathbf{w})p(\mathbf{w})$$

- Define prior with normal distribution, zero mean, identity towards zero; pushes parameters towards zero.
- MAP estimate:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}} \ln \left[p(\mathbf{w}) \prod_{j=1}^{N} p(y^j|\mathbf{x}^j,\mathbf{w}) \right]$$

Logistic Regression for Discrete Classification

- Logistic Regression in more general case, where Y = { y_1, ..., y_R}. Define a weight vector wi for each yi, i=1,...,R-1.

$$p(Y = 1|X) \propto \exp(w_{10} + \sum_i w_{1i} X_i)$$

$$p(Y = 2|X) \propto \exp(w_{20} + \sum_i w_{2i} X_i)$$

$$\vdots$$

$$p(Y = r|X) = 1 - \sum_{j=1}^{r-1} p(Y = j|X)$$

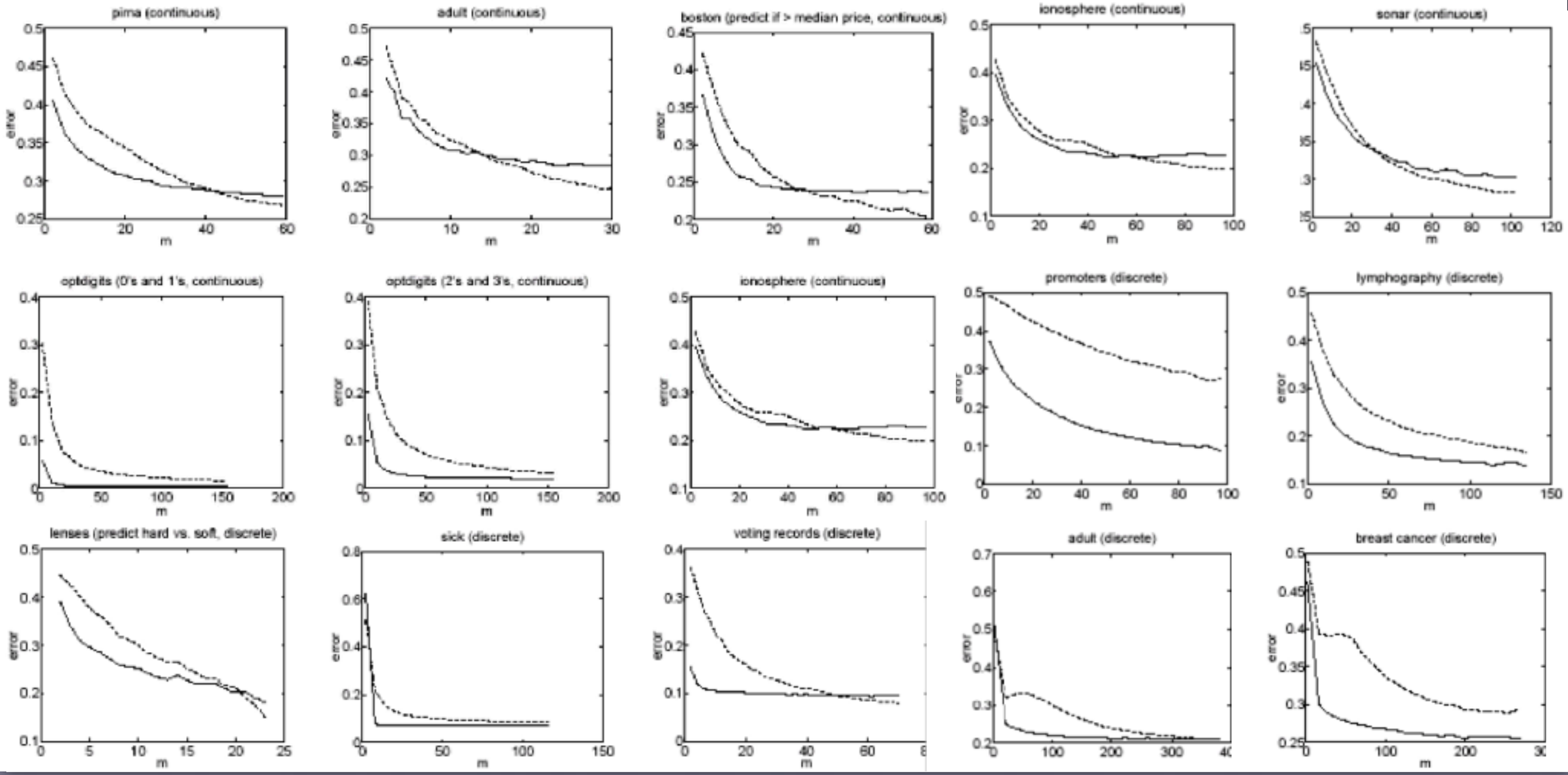
Naïve Bayes vs Logistic Regression

- E.g., Y={0,1}, X = <X_1, ..., X_n>, X_i continuous.

	Naïve Bayes (generative)	Logistic Regression (discriminative)
Number of parameters	4n+1	n+1
parameter estimation	uncoupled	coupled
when # training samples → infinite & model correct	good classifier	good classifier
when # training samples → infinite & model incorrect	biased classifier	less-biased classifier
Training samples needed	O(log N)	O (N)
Training convergence speed	faster	slower

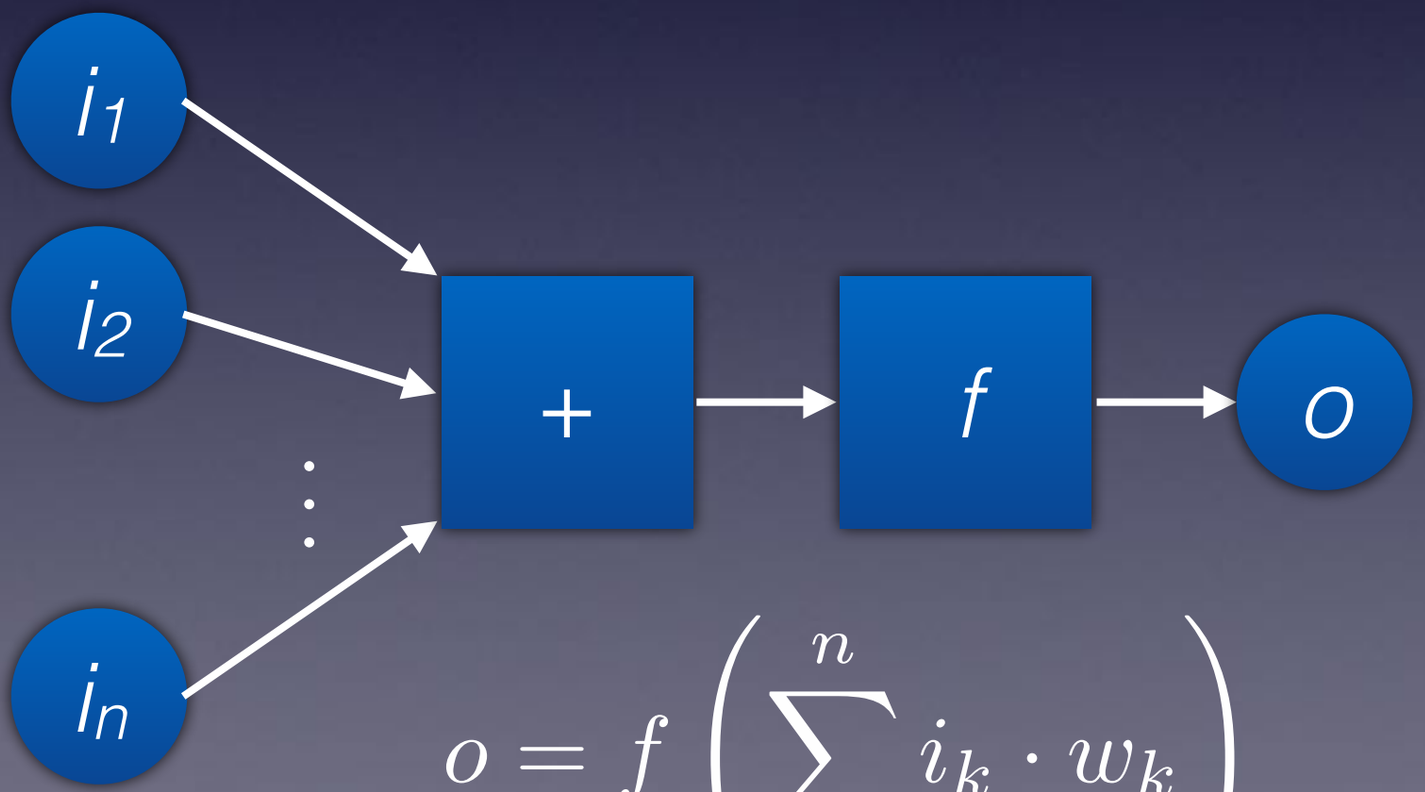
Naïve Bayes vs Logistic Regression

- Examples from UCI Machine Learning dataset

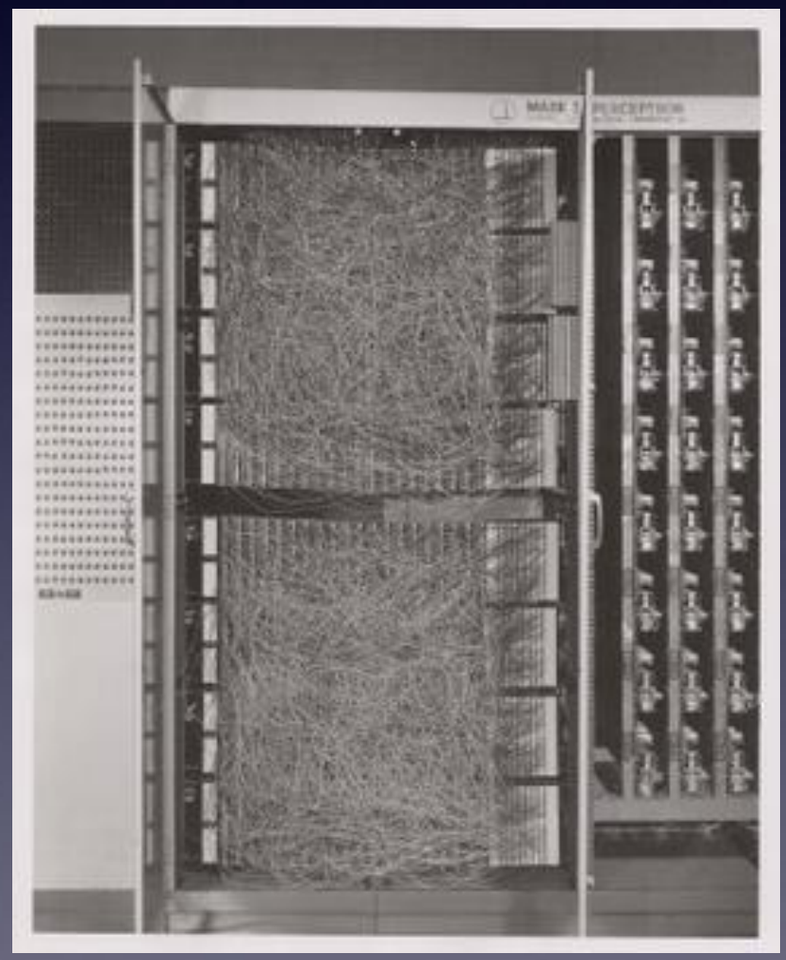


Perceptron

- Invented in 1957 at the Cornell Aeronautical Lab. Intended to be a machine instead of a program that is capable of recognition.
- A linear (binary) classifier.



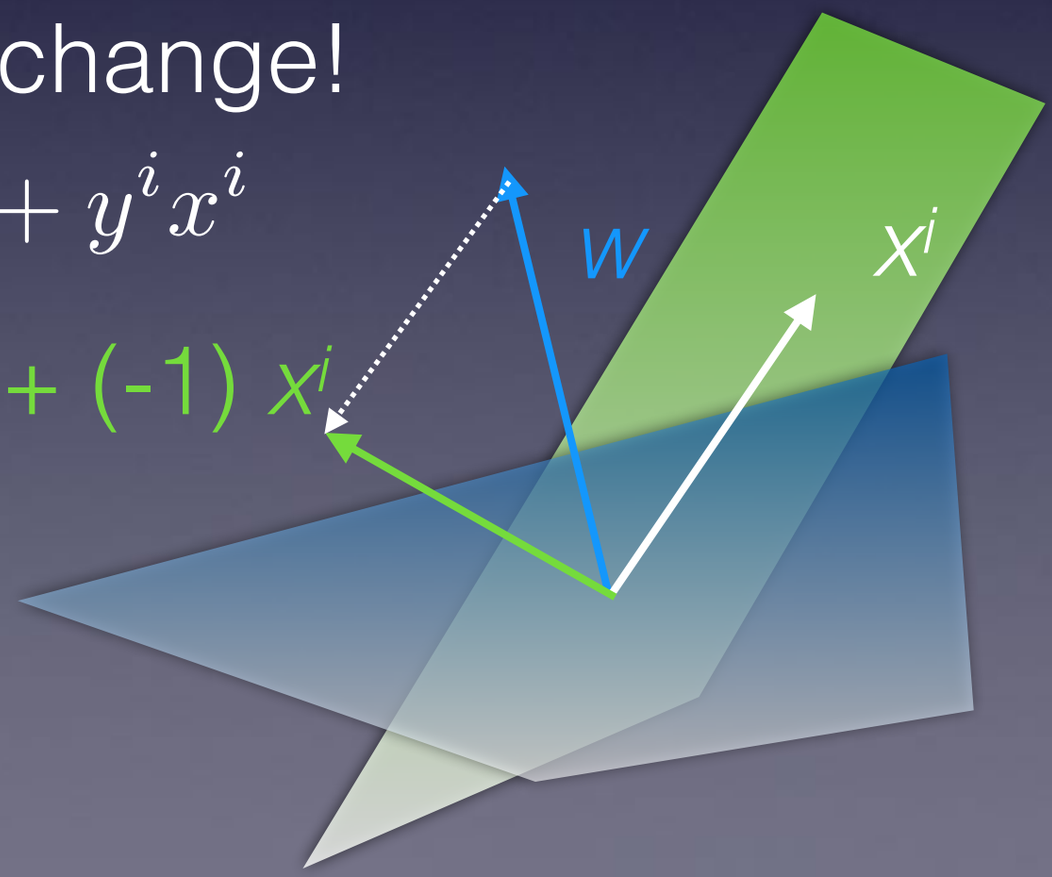
$$o = f\left(\sum_{k=1}^{n} i_k \cdot w_k\right)$$



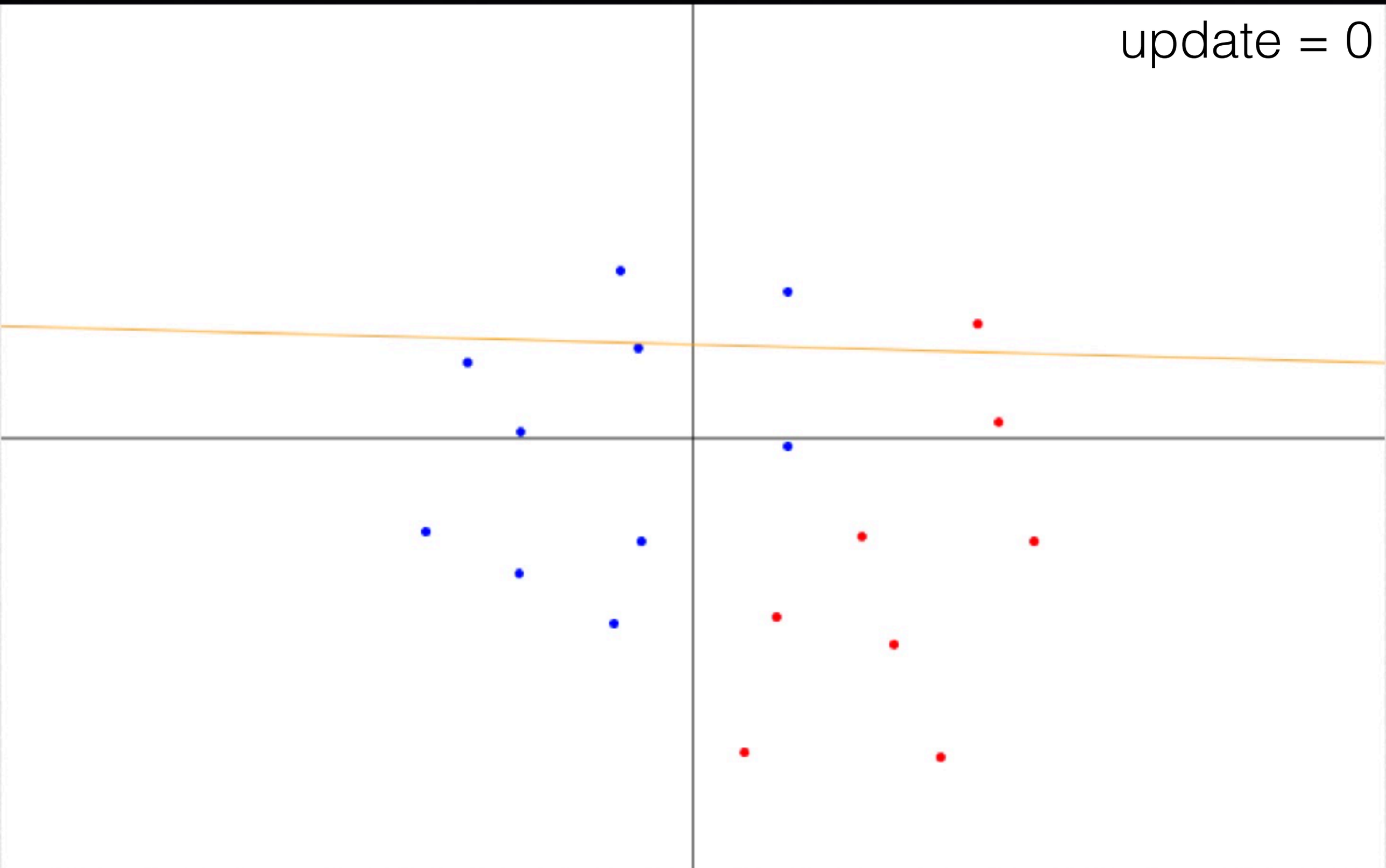
Mark I perceptron machine

Binary Perceptron Algorithm

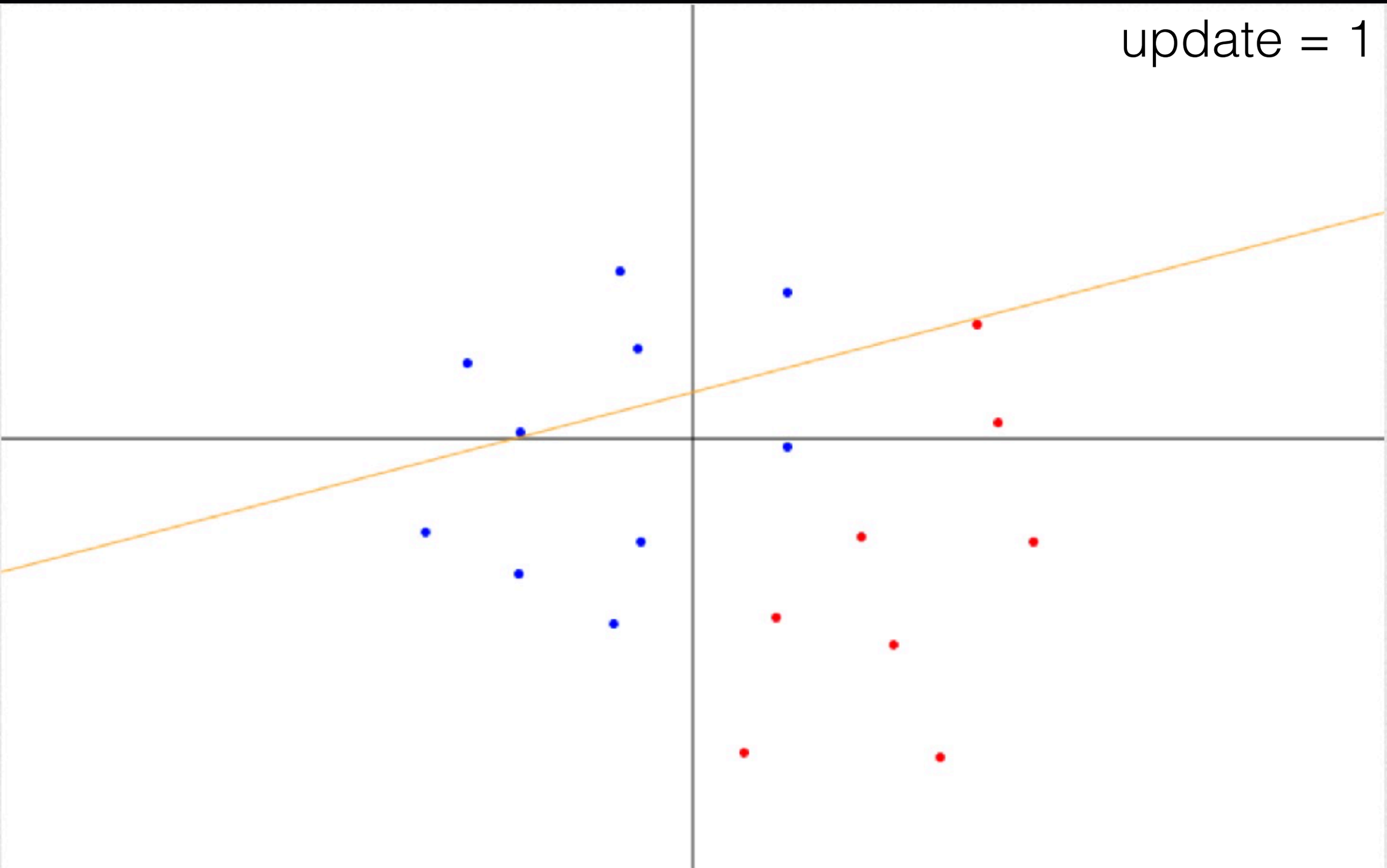
- Start with zero weights: w=0
- For t=1...T (T passes over data)
  - For i=1...n (each training sample)
    - Classify with current weights $y = \text{sign}(w \cdot x^i)$
      (sign(x) is +1 if x>0, else -1)
- If correct, (i.e., y=yi), no change!
- If wrong, update $w = w + y^i x^i$



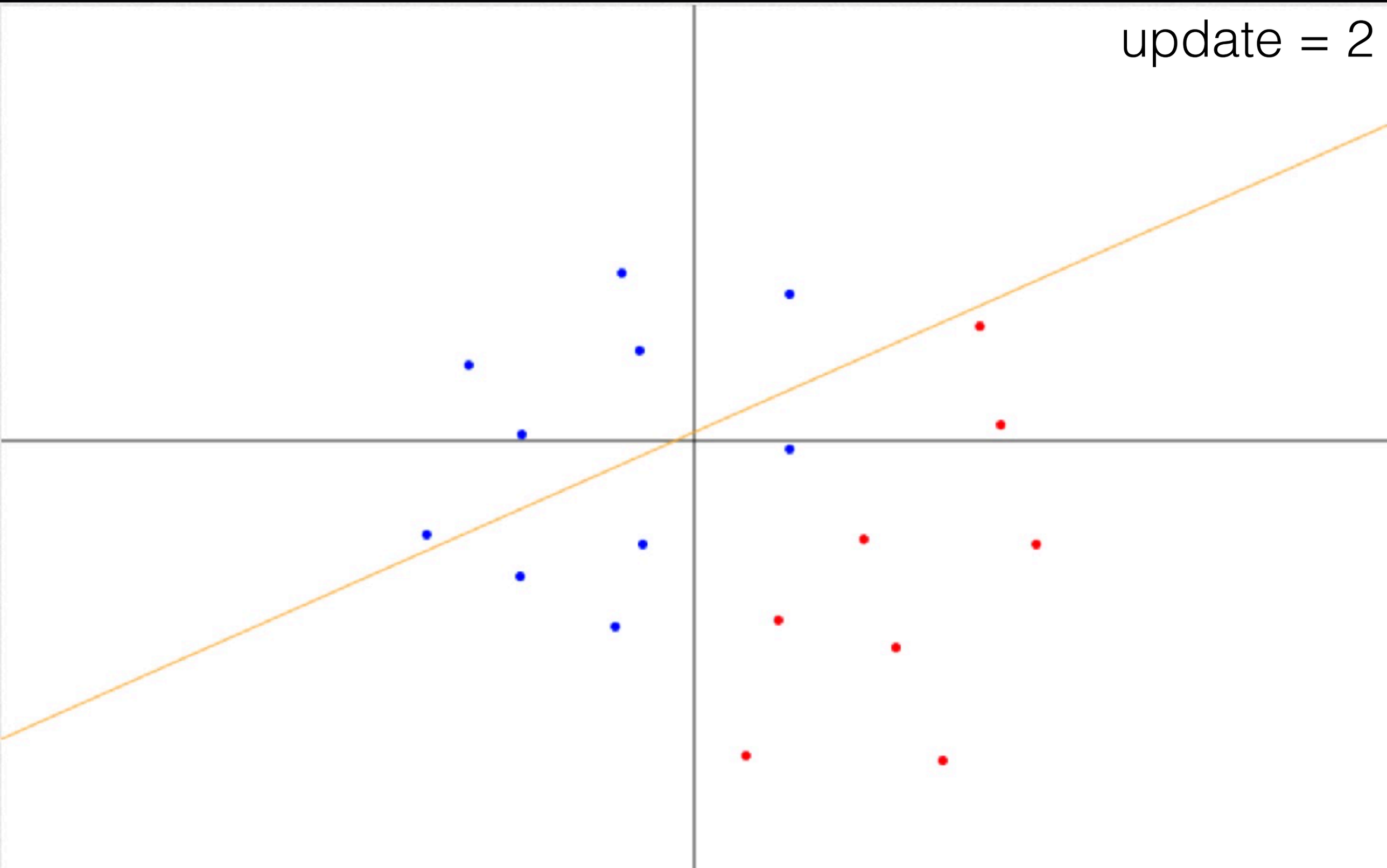
Binary Perceptron example



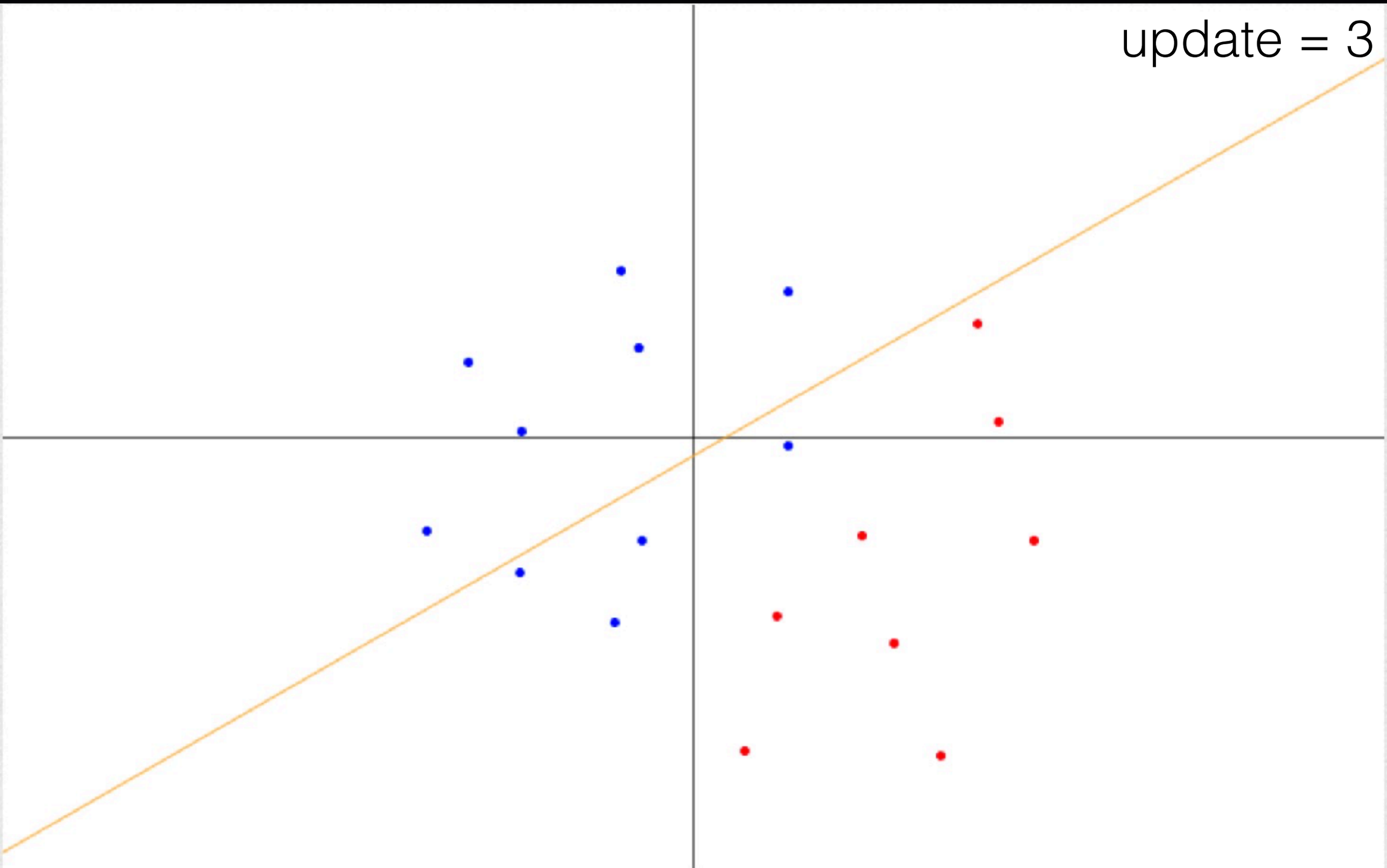
Binary Perceptron example



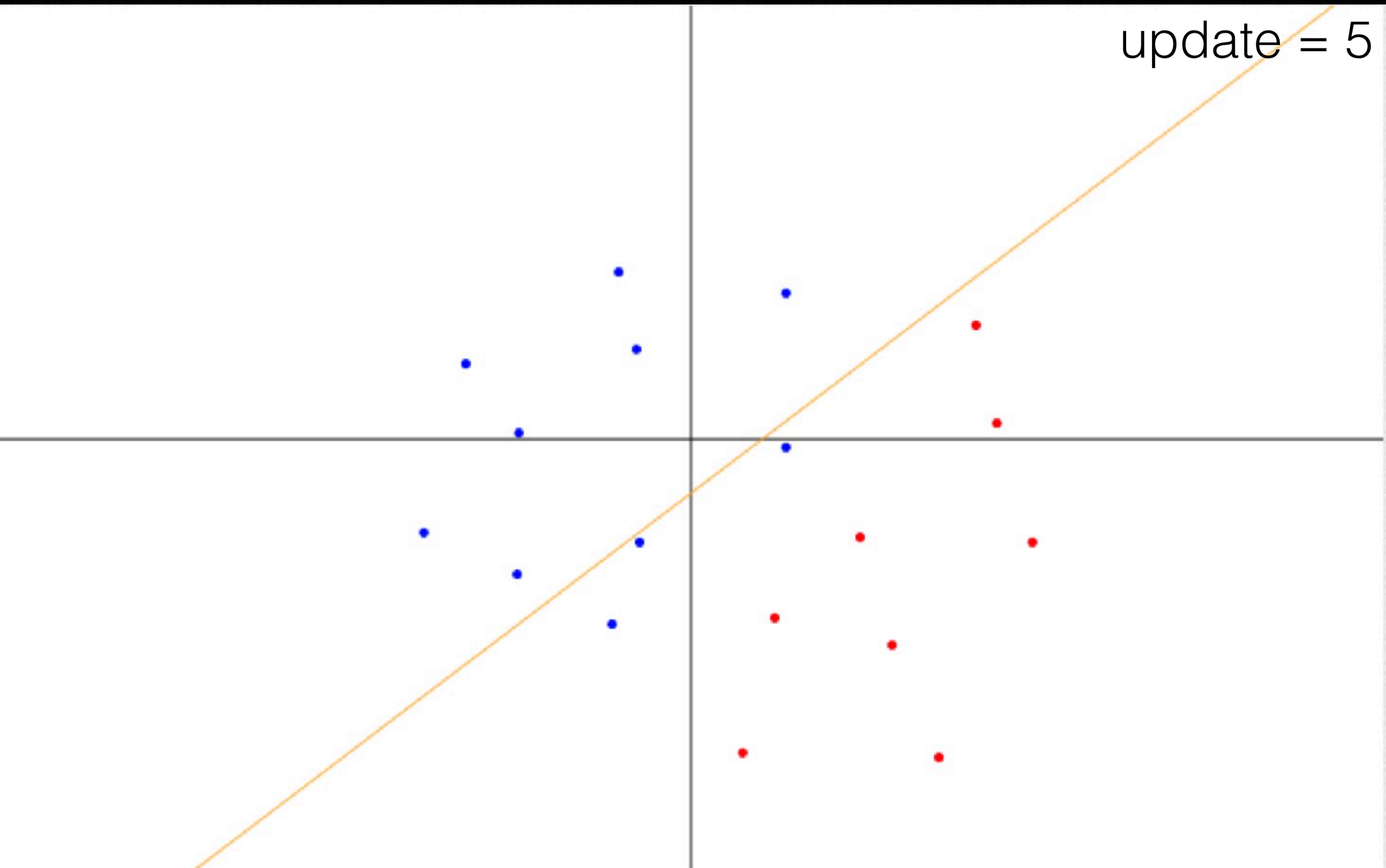
Binary Perceptron example



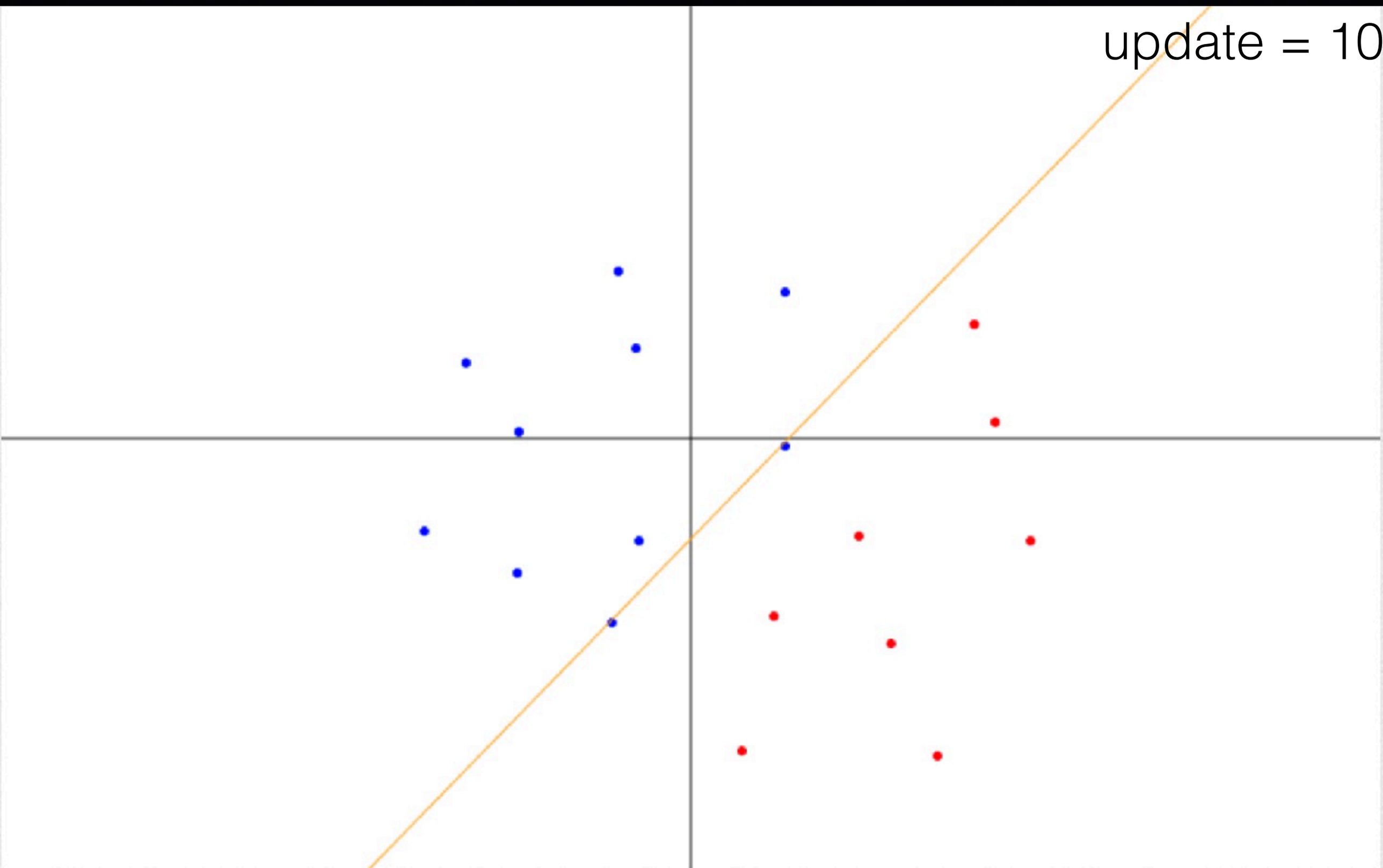
Binary Perceptron example



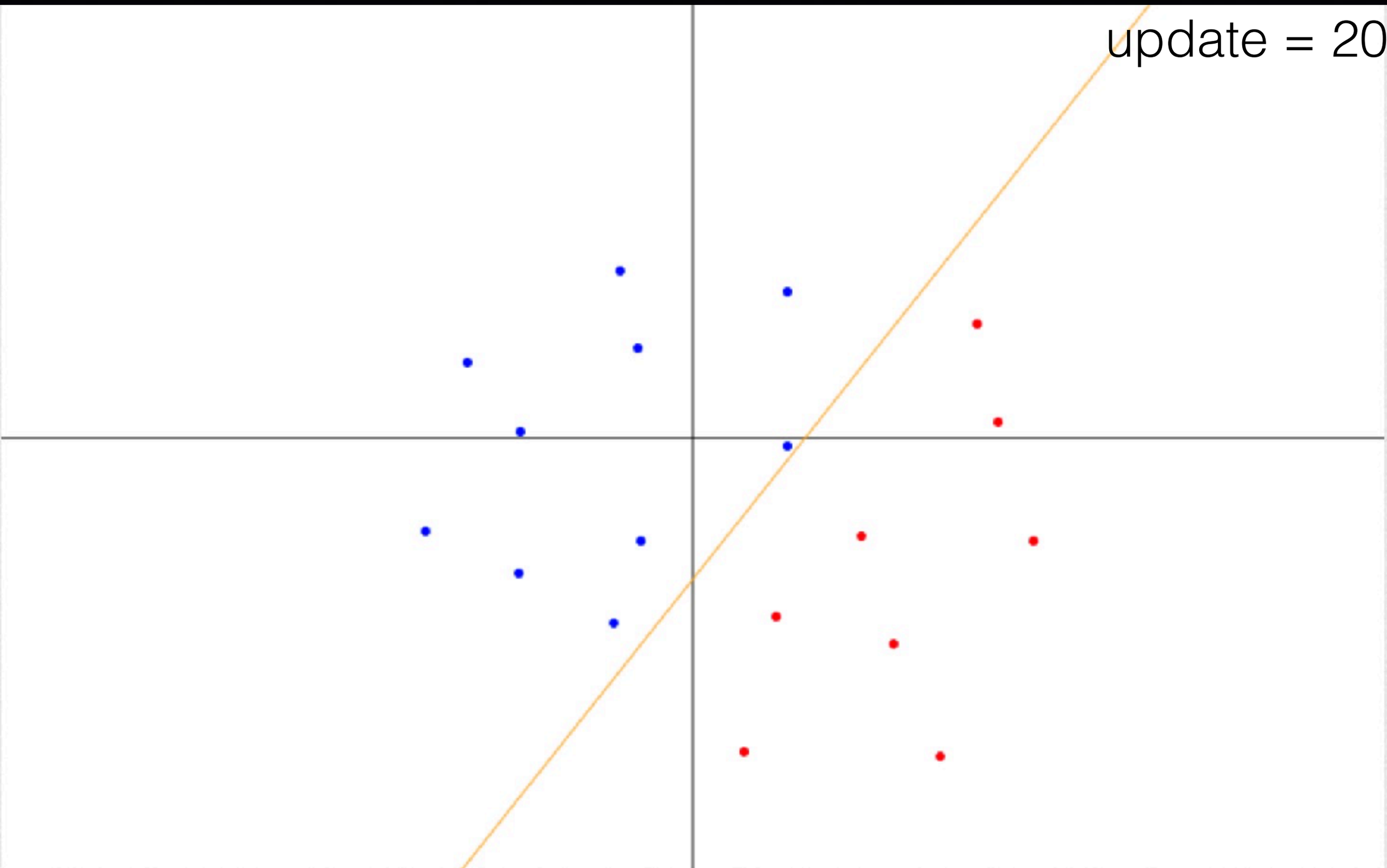
Binary Perceptron example



Binary Perceptron example



Binary Perceptron example



Multiclass Perceptron

- If we have more than two classes:
  - Have a weight vector for each class w_y
  - Calculate an activation function for each class

$$\text{activation}_w(x, y) = w_y \cdot x$$

- Highest activation wins

$$y^* = \underset{y}{\text{argmax}}(\text{activation}_w(x, y))$$

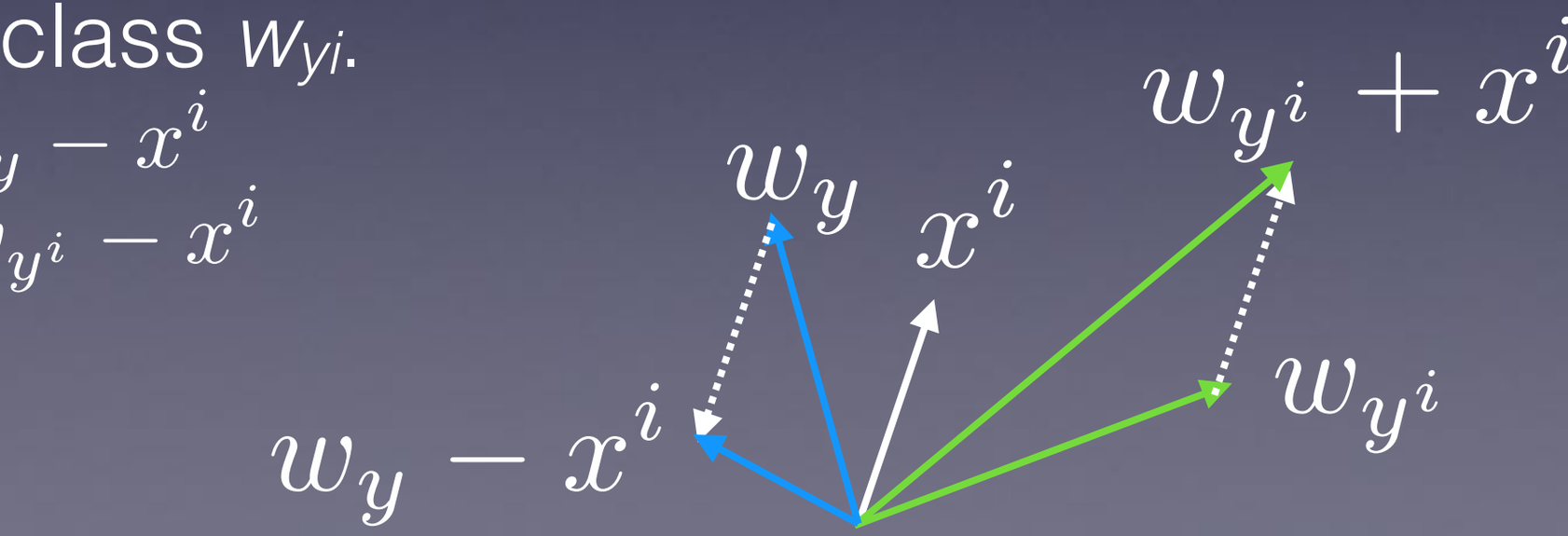
Multiclass Perceptron

- Starts with zero weights
- For t=1, ..., T, i=1, ..., n (T times over data)
  - Classify with current weights

$$y = \operatorname{argmax} w_y \cdot x^i$$

  - If correct ($\hat{y}$=y_i), no change!
  - If wrong: subtract features x^i from weights for predicted class w_y and add them to weights for correct class w_{y^i}.

$$w_y = w_y - x^i$$
$$w_{y^i} = w_{y^i} + x^i$$



Multiclass Perceptron Example

- Text classification example:
  x = "win the vote" sentence

x

BIAS	1
win	1
game	0
vote	1
the	1
...	

$\cdot$

w_{sports}

BIAS	-2
win	4
game	4
vote	0
the	0
...	

$x \cdot w_{\text{sports}} = 2$

w_{politics}

BIAS	1
win	2
game	0
vote	4
the	0
...	

$x \cdot w_{\text{politics}} = 7$

w_{tech}

BIAS	2
win	0
game	2
vote	0
the	0
...	

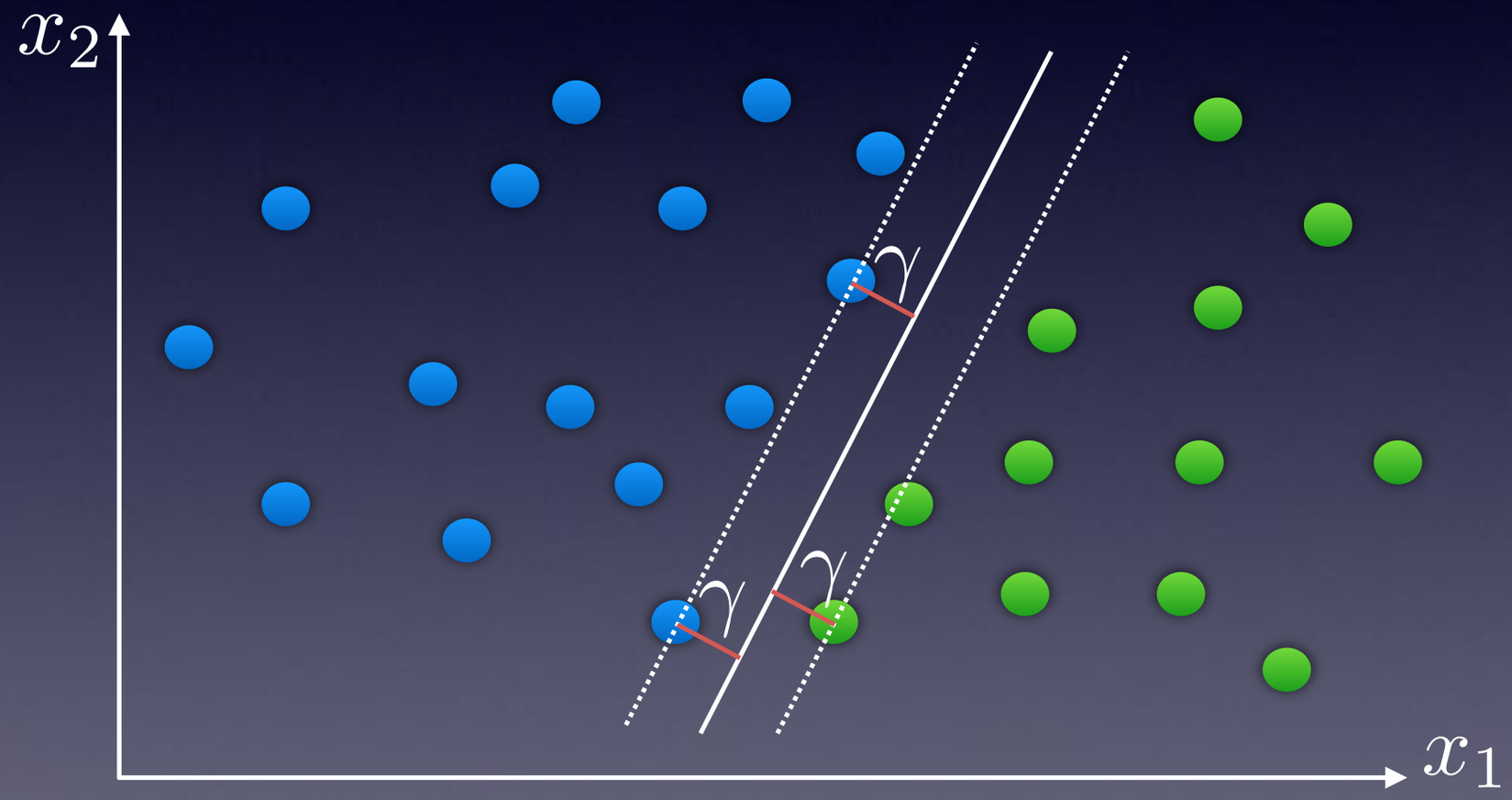
$x \cdot w_{\text{tech}} = 2$

Classified as "politics"

Linearly separable (binary)

- The data is linearly separable with margin γ if

$$\exists w \, \forall t \;\; y^t (w \cdot x^t) \geq \gamma > 0$$



Mistake Bound for Perceptron

- Assume data is separable with margin γ

$$\exists w^* \text{ s.t.} ||w^*||_2 = 1 \text{ and } \forall t \; y^t(w^* \cdot^t) \geq \gamma$$

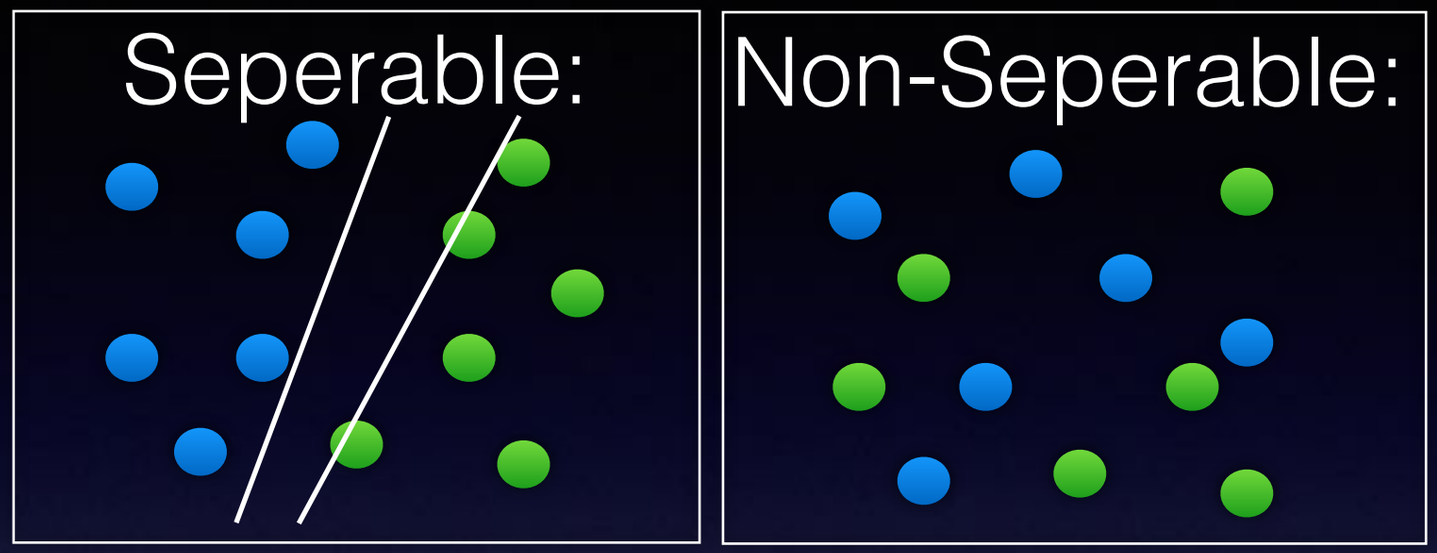
- Also assume there is a number *R* such that

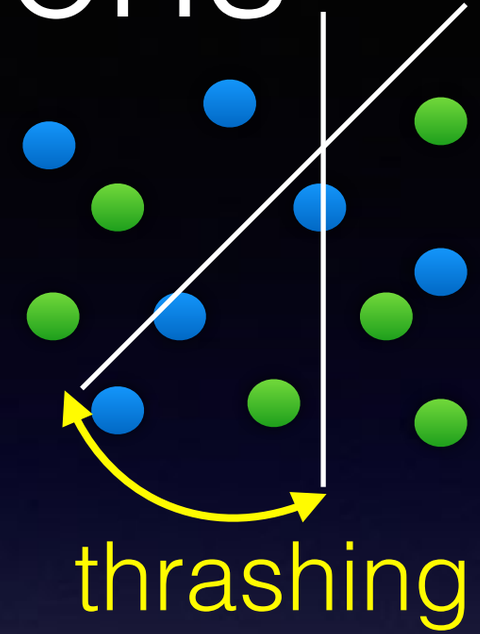
$$\forall t \; ||x^t||_2 \leq R$$

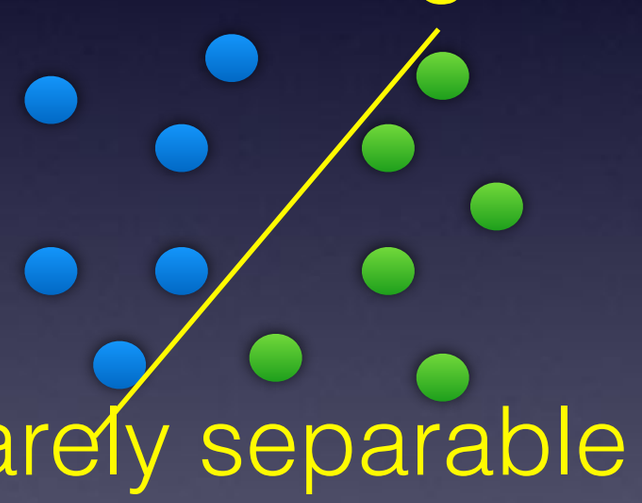
- Theorem: the number of mistakes (parameter updates) made by the perceptron is bounded:

$$\text{mistakes} \leq \frac{R^2}{r^2}$$

Issues with Perceptrons



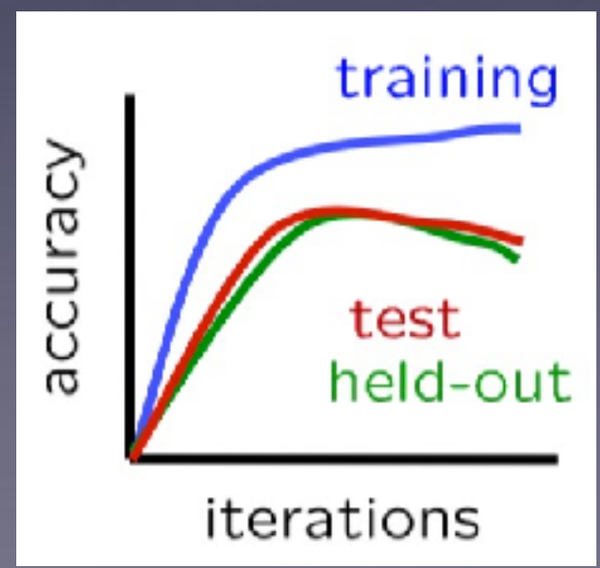




- Noise: if the data isn't separable, weights might thrash (averaging weight vectors over time can help).

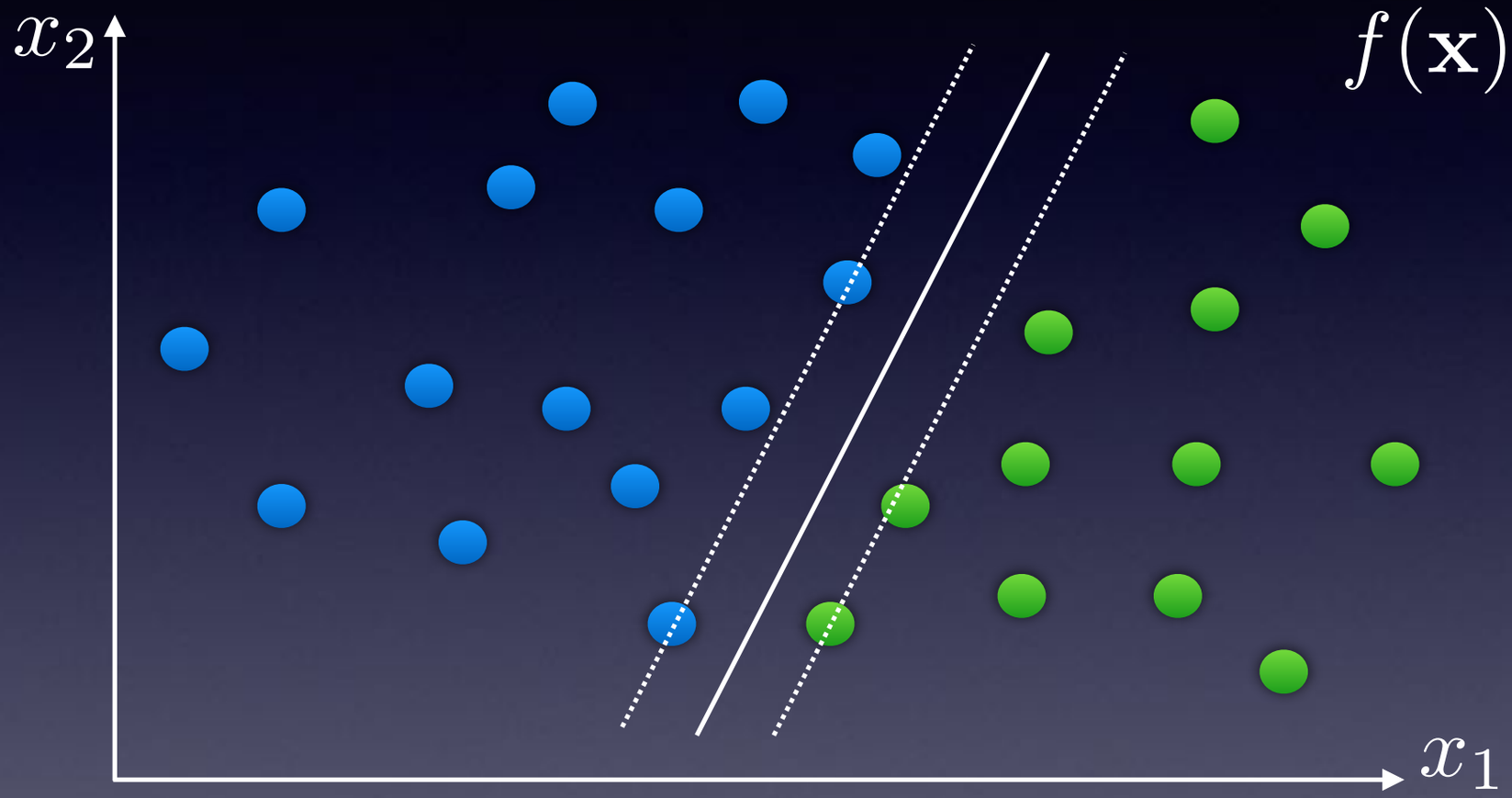
- Mediocre generalization: finds a barely separating solution.

- Overtraining: test / hold-out accuracy usually rises then falls.



Linear SVM Classifier

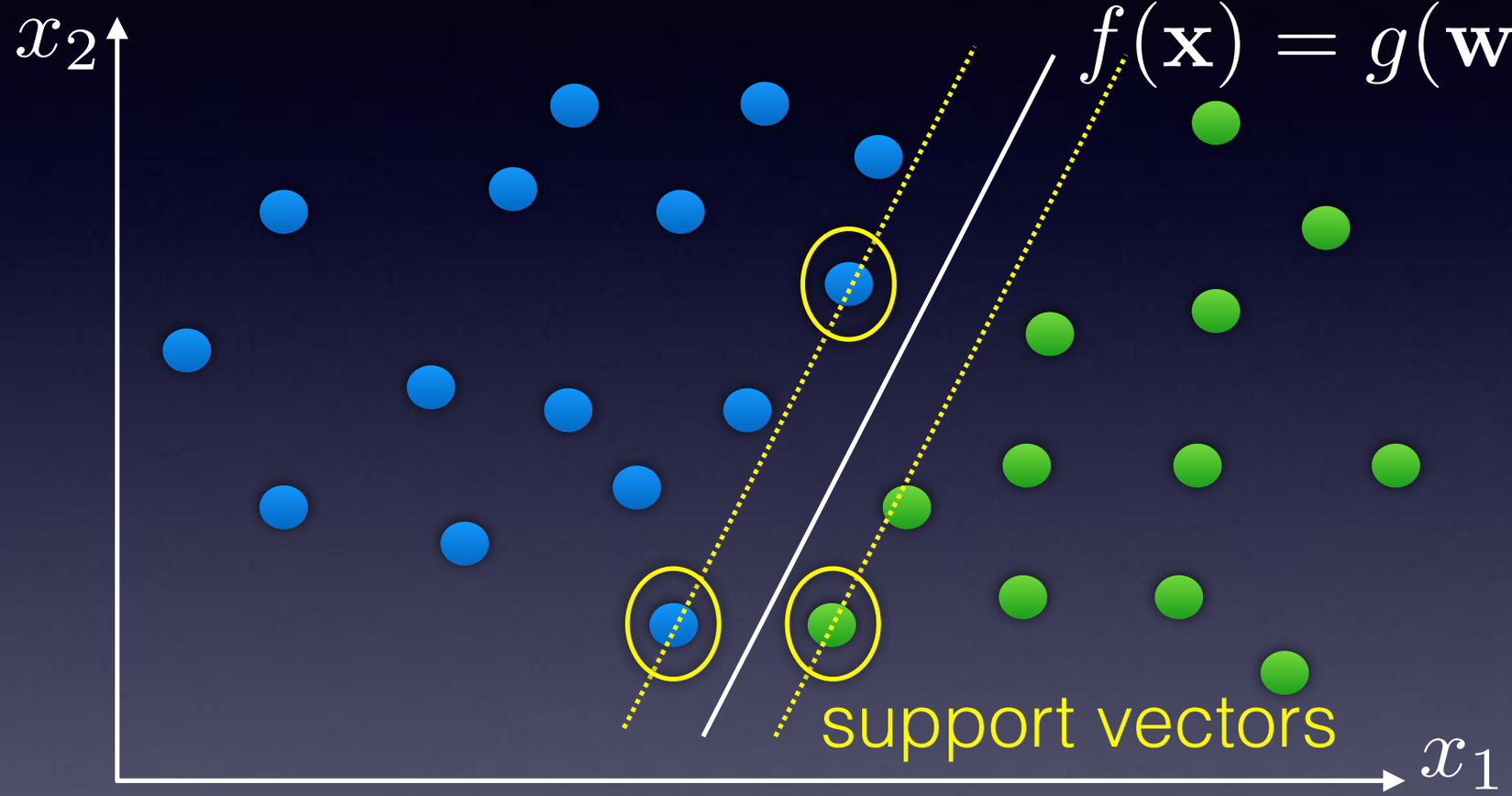
- Find a linear function to separate the classes



- Define hyperplane $\mathbf{tX} - b = 0$ where t is the tangent to hyperplane, $\mathbf{X}$ is the matrix of all data points. Minimize $||\mathbf{t}||$ s.t. $\mathbf{tX} - b$ produces correct label for all $\mathbf{X}$.

Linear SVM Classifier

- Find a linear function to separate the classes



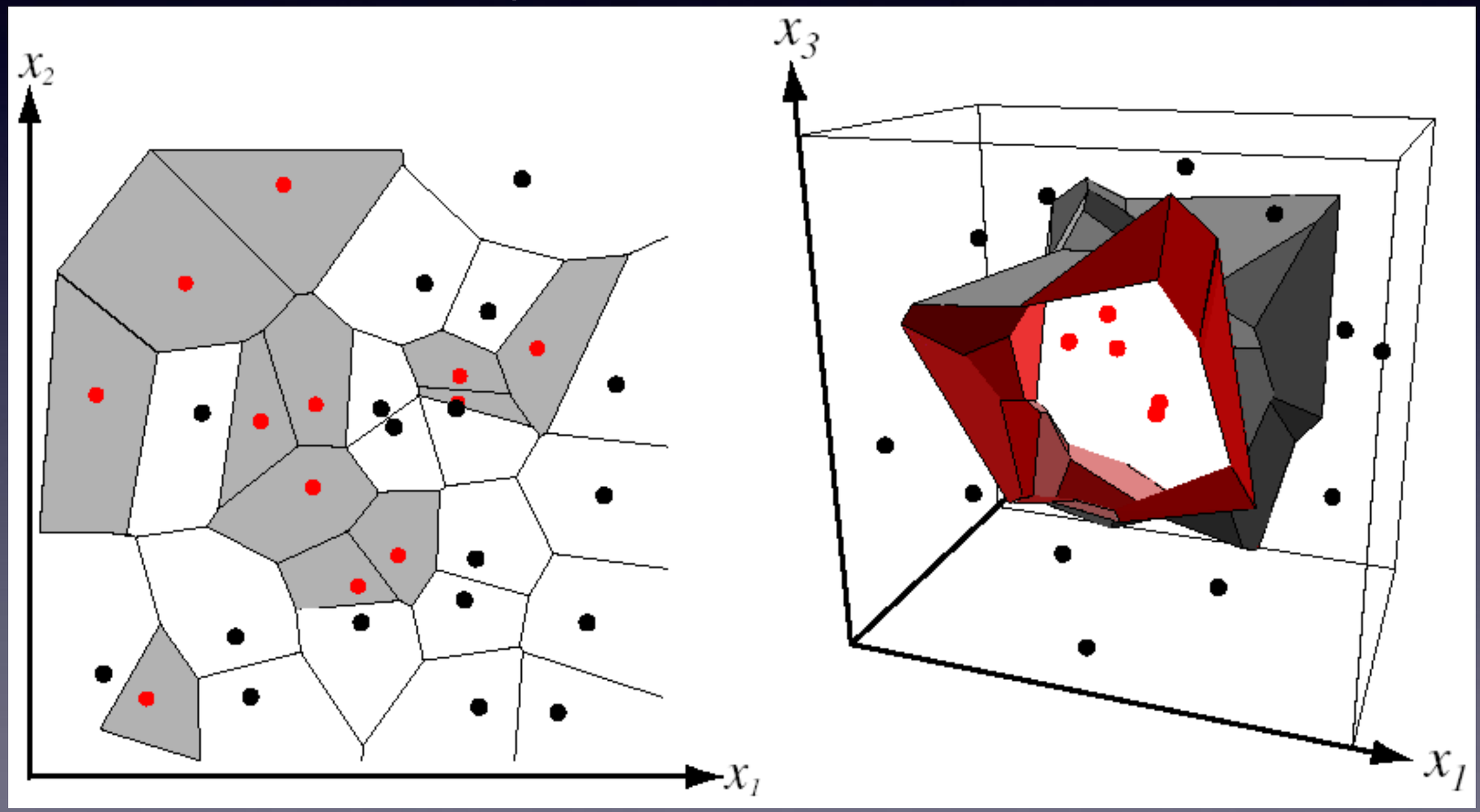
- Define hyperplane $\mathbf{t}\mathbf{X} - b = 0$ where t is the tangent to hyperplane, $\mathbf{X}$ is the matrix of all data points. Minimize $||\mathbf{t}||$ s.t. $\mathbf{t}\mathbf{X} - b$ produces correct label for all $\mathbf{X}$.

Nonlinear Classifiers

- Some data sets are not linearly separable!
- Option 1:
  - Use non-linear features, e.g., polynomial basis functions
    - Learn linear classifers in a transformed, non-linear feature space
- Option 2:
  - Use non-linear classifiers (decision trees, neural networks, nearest neighbors)

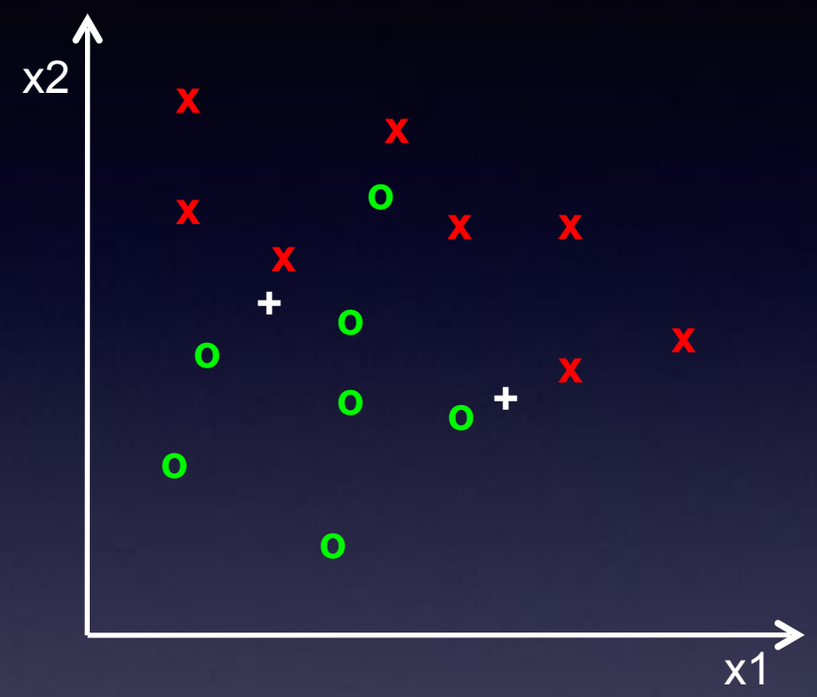
Nearest Neighbor Classifier

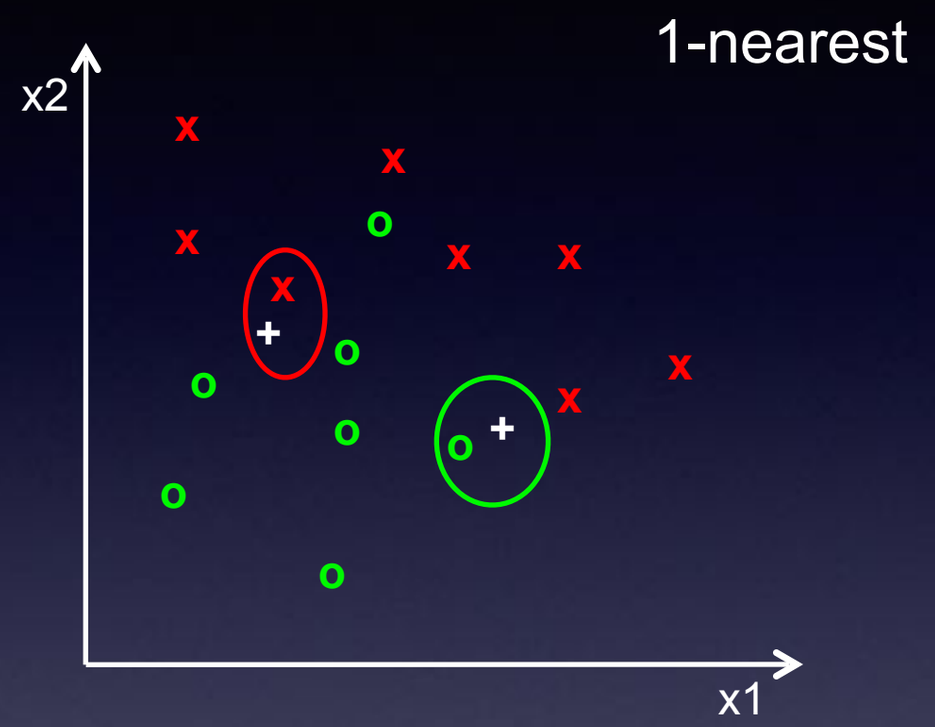
- Assign label of nearest training data point to each test data point.

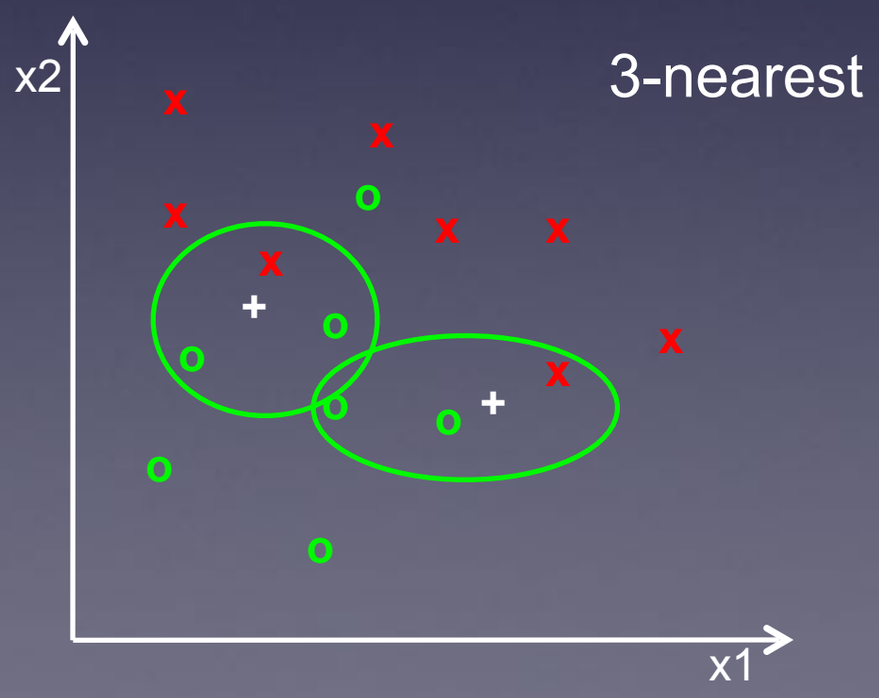


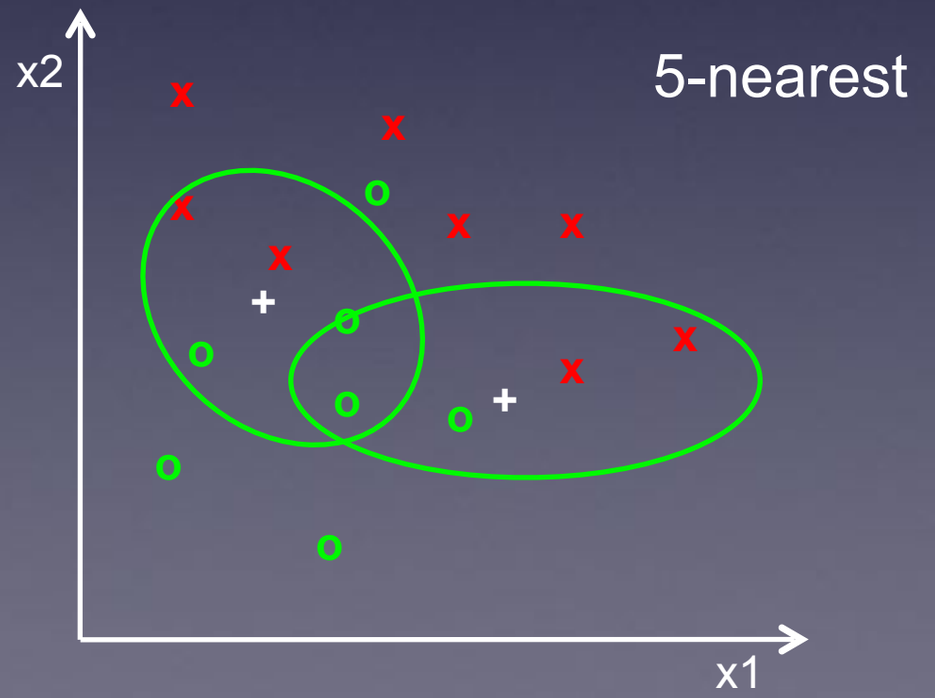
Duda, Hart and Stork, Pattern Classification

K-Nearest Neighbor Classifier



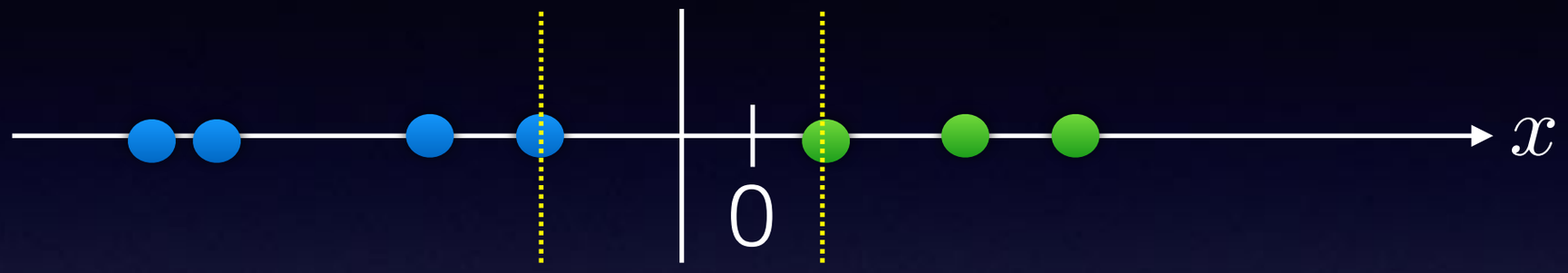




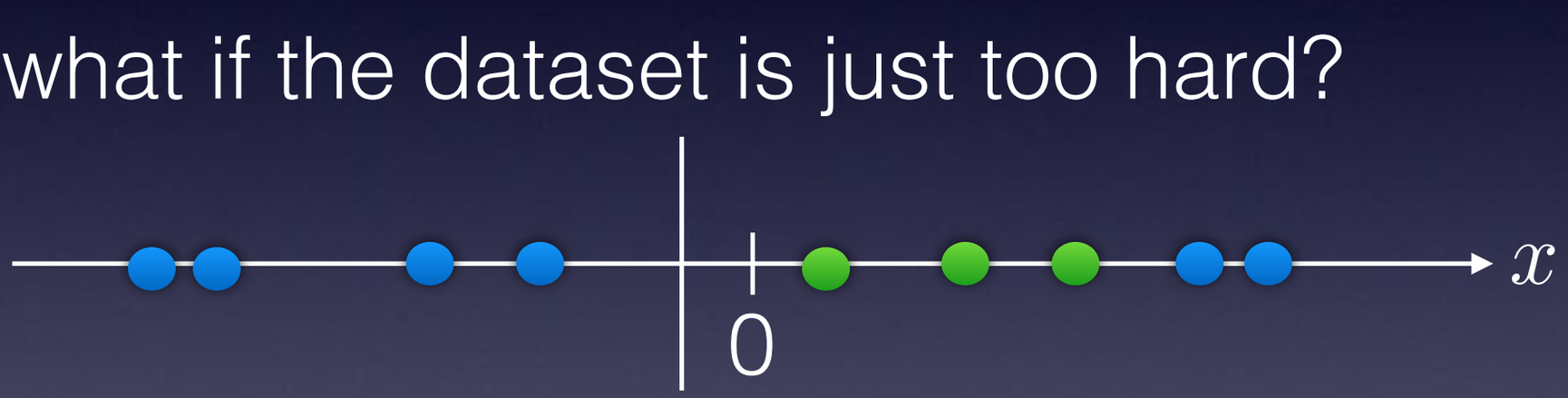


Nonlinear SVMs

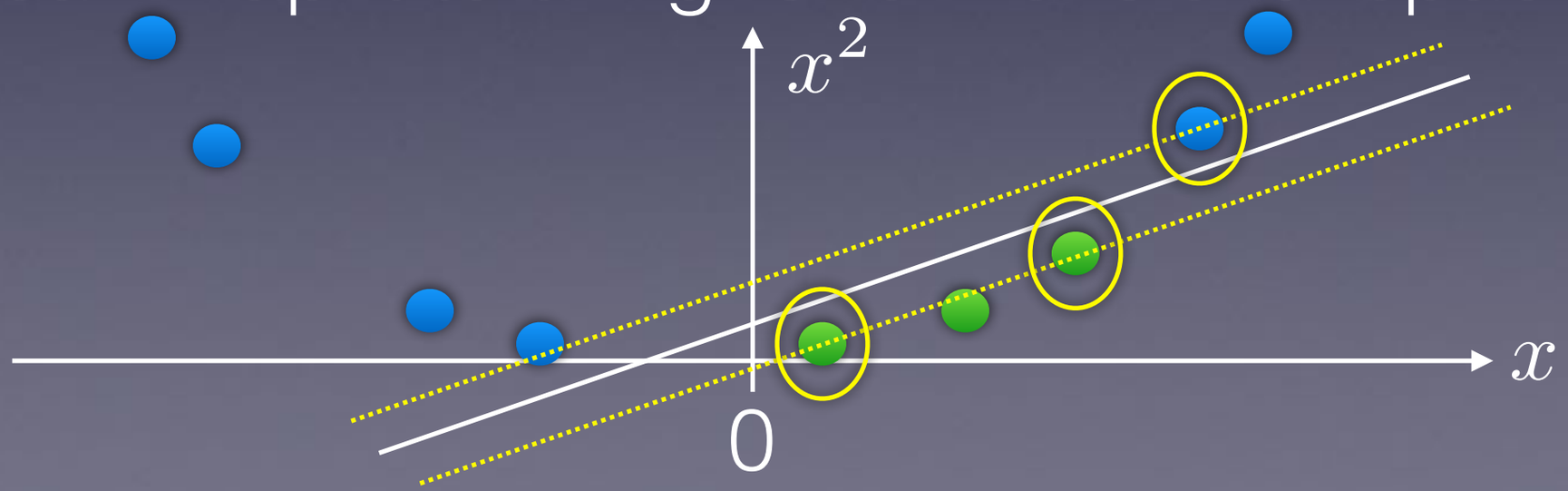
- Data that are linearly separable work out great:



- But what if the dataset is just too hard?

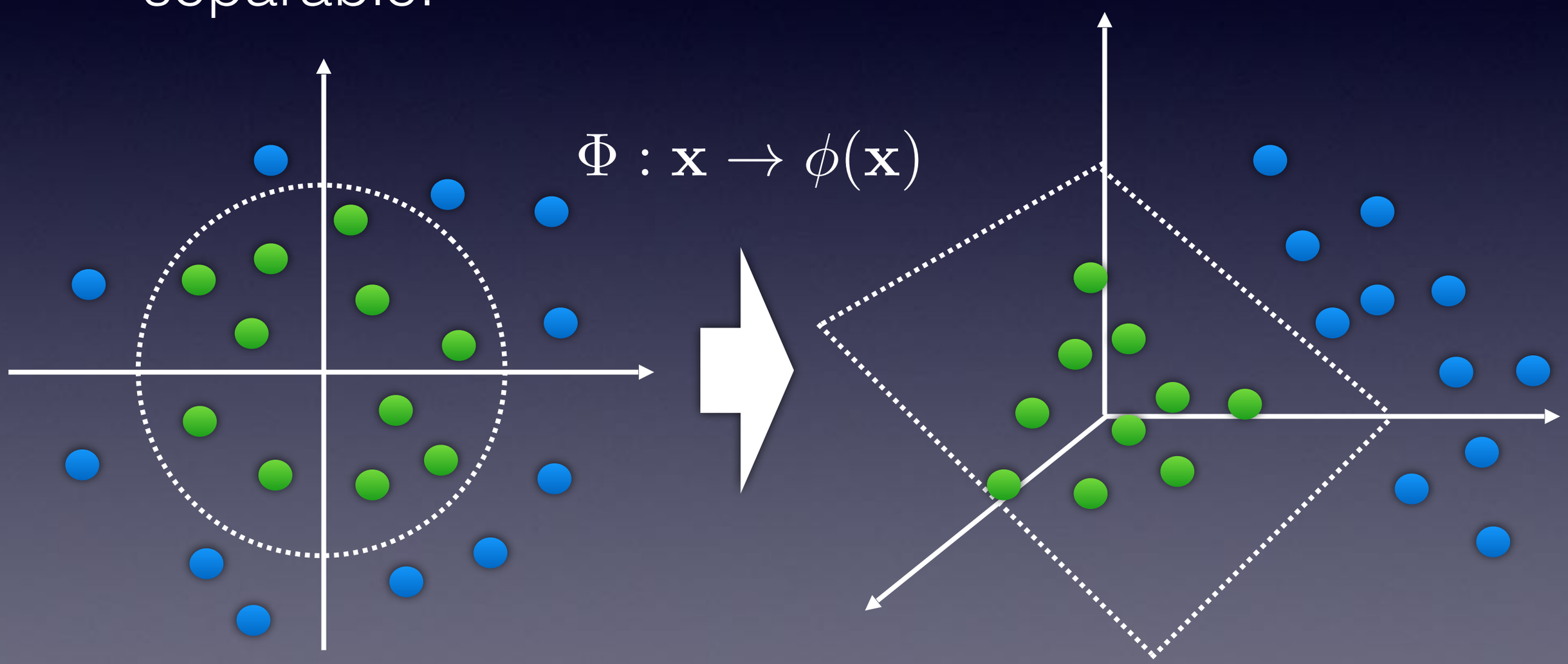


- We can map it to a higher-dimensional space!



Nonlinear SVMs

- Map the input space to some higher dimensional feature space where the training set is separable:



Nonlinear SVMs

- The kernel trick: instead of explicitly computing the lifting transformation

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$$

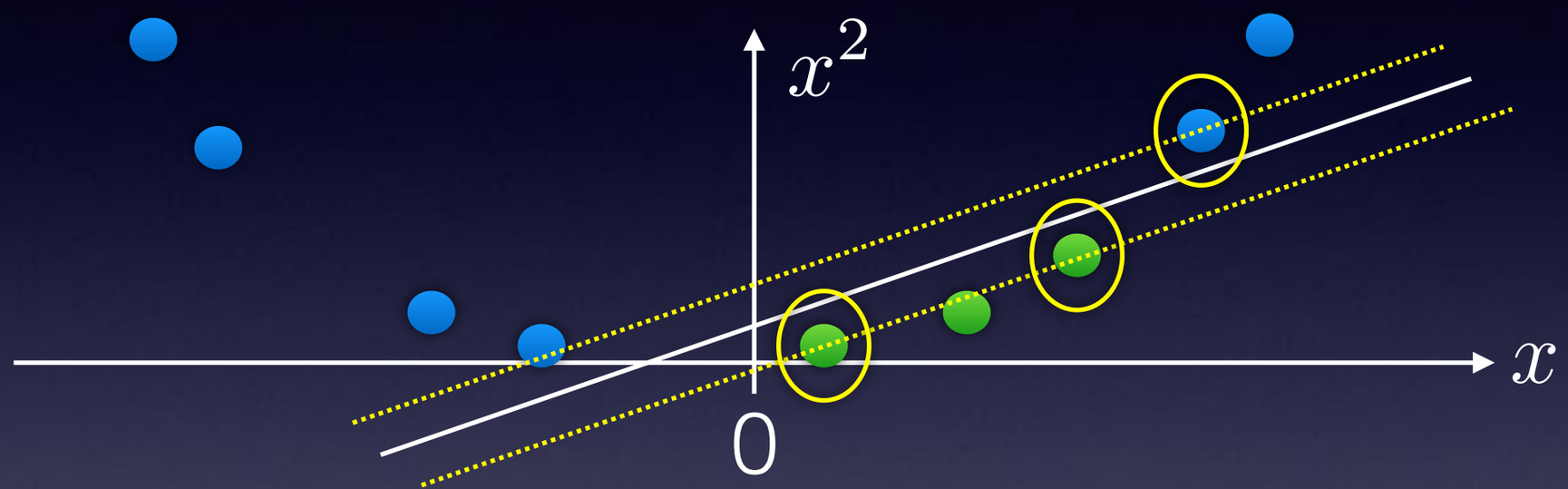
- This gives a non-linear decision boundary in the original feature space:

$$\sum_i \alpha_i y_i \phi(x_i) \cdot \phi(x) + b = \sum_i \alpha_i y_i K(x_i, x) + b$$

- Common kernel function: Radial basis function kernel.

Nonlinear kernel example

- Consider the mapping: $\phi(x) = (x, x^2)$



$$\phi(x) \cdot \phi(y) = (x, x^2) \cdot (y, y^2) = xy + x^2y^2$$

$$K(x, y) = xy + x^2y^2$$

Kernels for bags of features

- Histogram intersection kernel:

$$I(h_1, h_2) = \sum_{i=1}^{N} \min(h_1(i), h_2(i))$$

- Generlized Gaussian kernel:

$$K(h_1, h_2) = \exp\left(-\frac{1}{A} D(h_1, h_2)^2\right)$$

D can be (inverse) L1 distance, Euclidean distance, $\mathcal{X}^2$ distance, etc.

Multi-class SVM

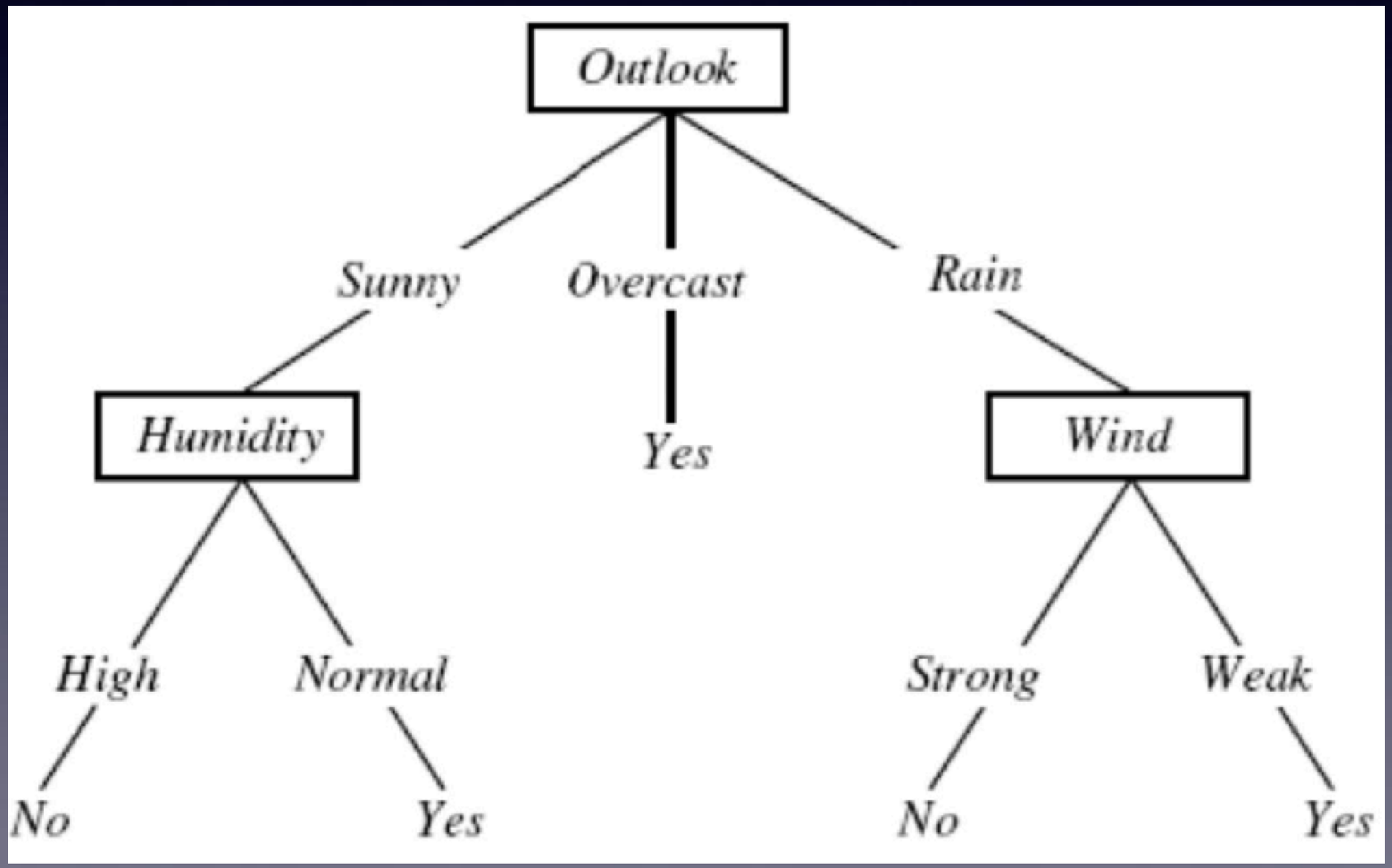
- Combine multiple two-class SVMs
  - One vs others:
    - Training: learn an SVM for each class vs the others.
    - Testing: apply each SVM to test example and assign it to the class of the SVM that returns the highest decision value.
  - One vs one:
    - Training: learn an SVM for each pair of classes
    - Testing: each learned SVM votes for a class to assign to the test example.

SVMs: Pros & Cons

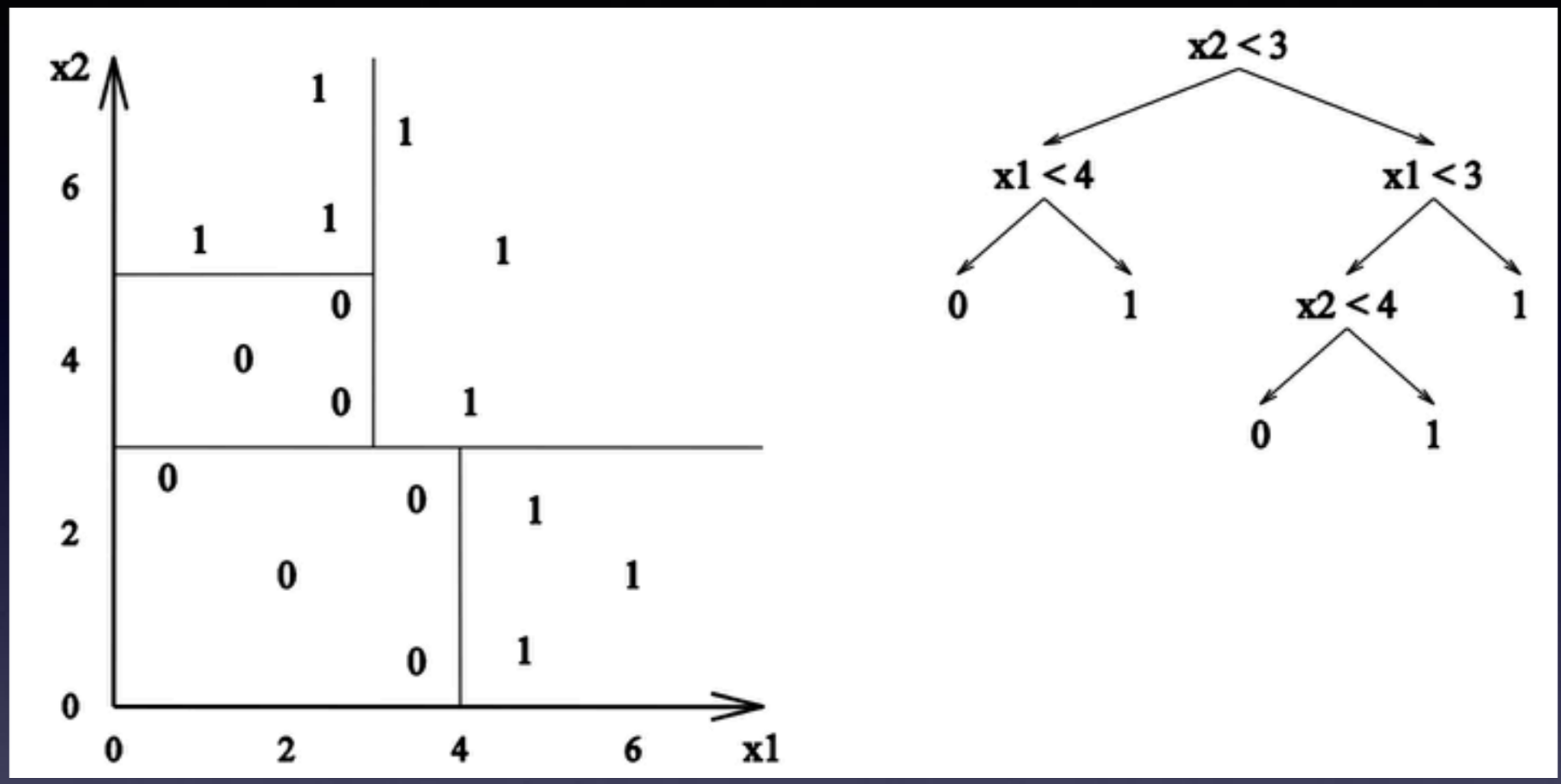
- Pros:
  - SVMs work very well in practice, even with very small training sample sizes.
- Cons:
  - No direct multi-class SVM; must combine two-class SVMs.
  - Computation and memory usage:
    - Must compute matrix of kernel values for each pair of examples.
    - Learning can take a long time for large problems.

Decision Tree Classifier

- Prediction is done by sending the example down the tree until a class assignment is reached.



Decision Tree Classifier



- *Internal Nodes*: each test a feature
- *Leaf nodes*: each assign a classification
- Decision Trees divide the feature space into axis-parallel rectangles and label each rectangle with one of the K classes.

Training Decision Trees

- Goal: find a decision tree that achieves minimum misclassification errors on the training data.
- Brute-force solution: create a tree with one path from root to leaf for each training sample. (problem: just memorizing, won't generalize.)
- Find the smallest tree that minimizes error. (problem: this is NP-hard.)

Top-down induction of Decision Tree

1. Choose the best feature a* for the root of the tree.
2. Split training set **S** into subsets $\{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_k\}$ where each subset $\mathbf{S}_i$ contains examples having the same value for a*.
3. Recursively apply the algorithm on each new subset until all examples have the same class label.

The problem is, what defines the "best" feature?

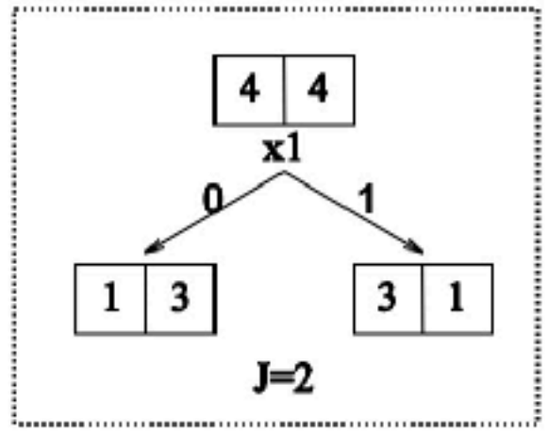
Choosing Best Feature

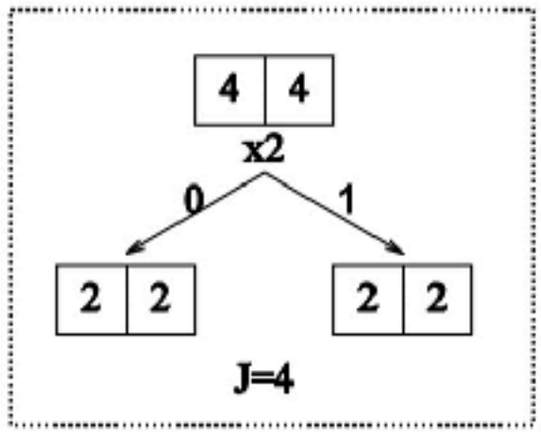
- Decision Tree feature selection based on classification error.

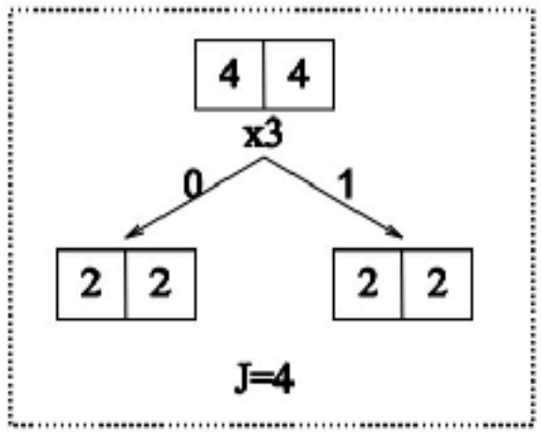
- Perform 1-step look-ahead search and choose the attribute that gives the lowest error rate on the training data

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Training examples







Does not work well, since it doesn't reflect progress towards a good tree.

Choosing Best Feature

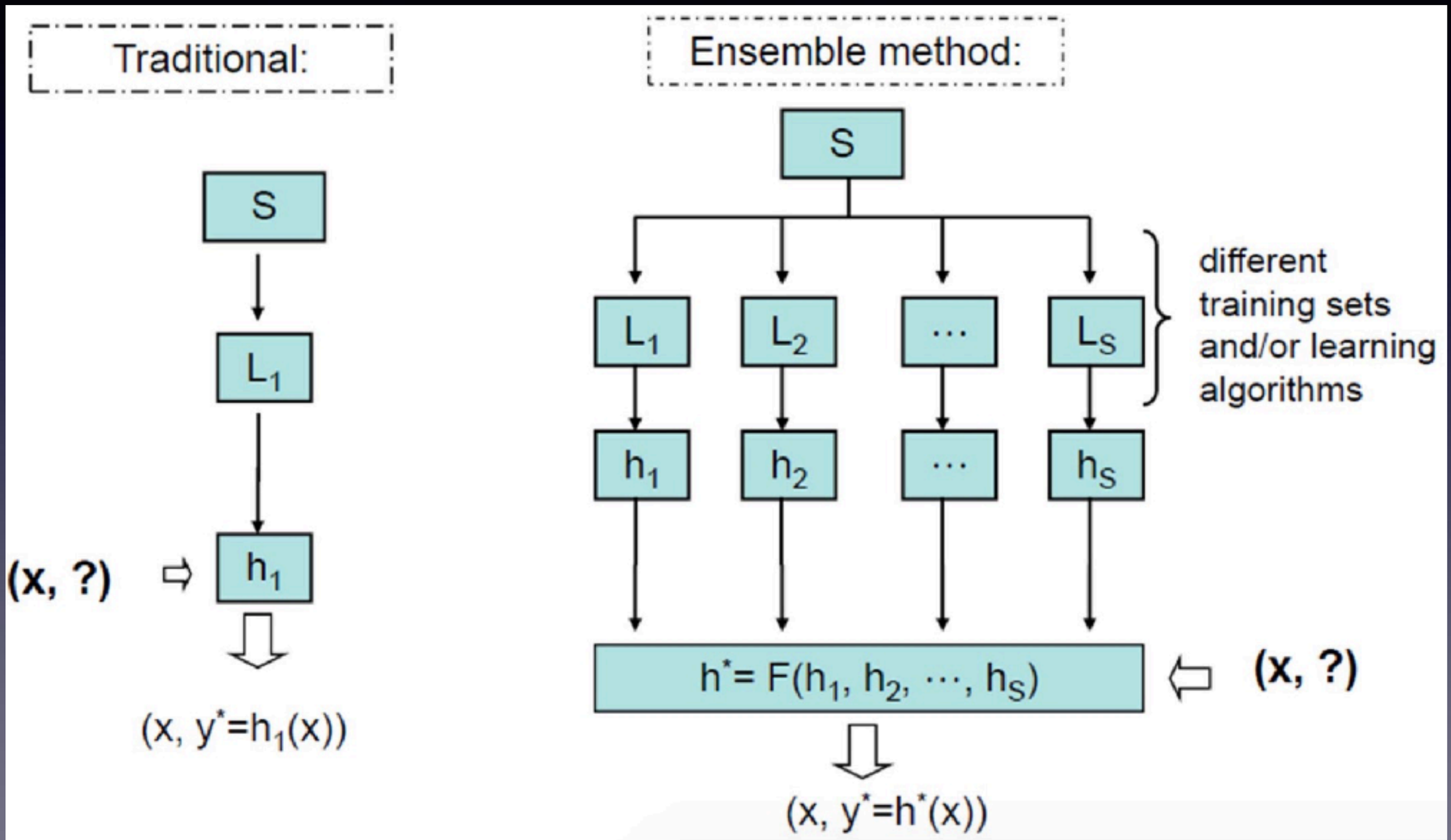
- Choose feature that gives the highest *information gain* (*X* that has the highest mutual information with *Y*).

$$\begin{aligned}\underset{j}{\operatorname{argmax}}\, I(X_j; Y) &= \underset{j}{\operatorname{argmax}}\, H(Y) - H(Y|X_j) \\ &= \underset{j}{\operatorname{argmin}}\, H(Y|X_j)\end{aligned}$$

- Define $\tilde{J}(j)$ to be the expected remaining uncertainty about *y* after testing $\mathbf{x}_j$.

$$\tilde{J}(j) = H(Y_X)j) = \sum_x p(X_j = x) H(Y|X_j = x)$$

Ensembles: Combining Classifiers



Bootstrap Aggregating (Bagging)

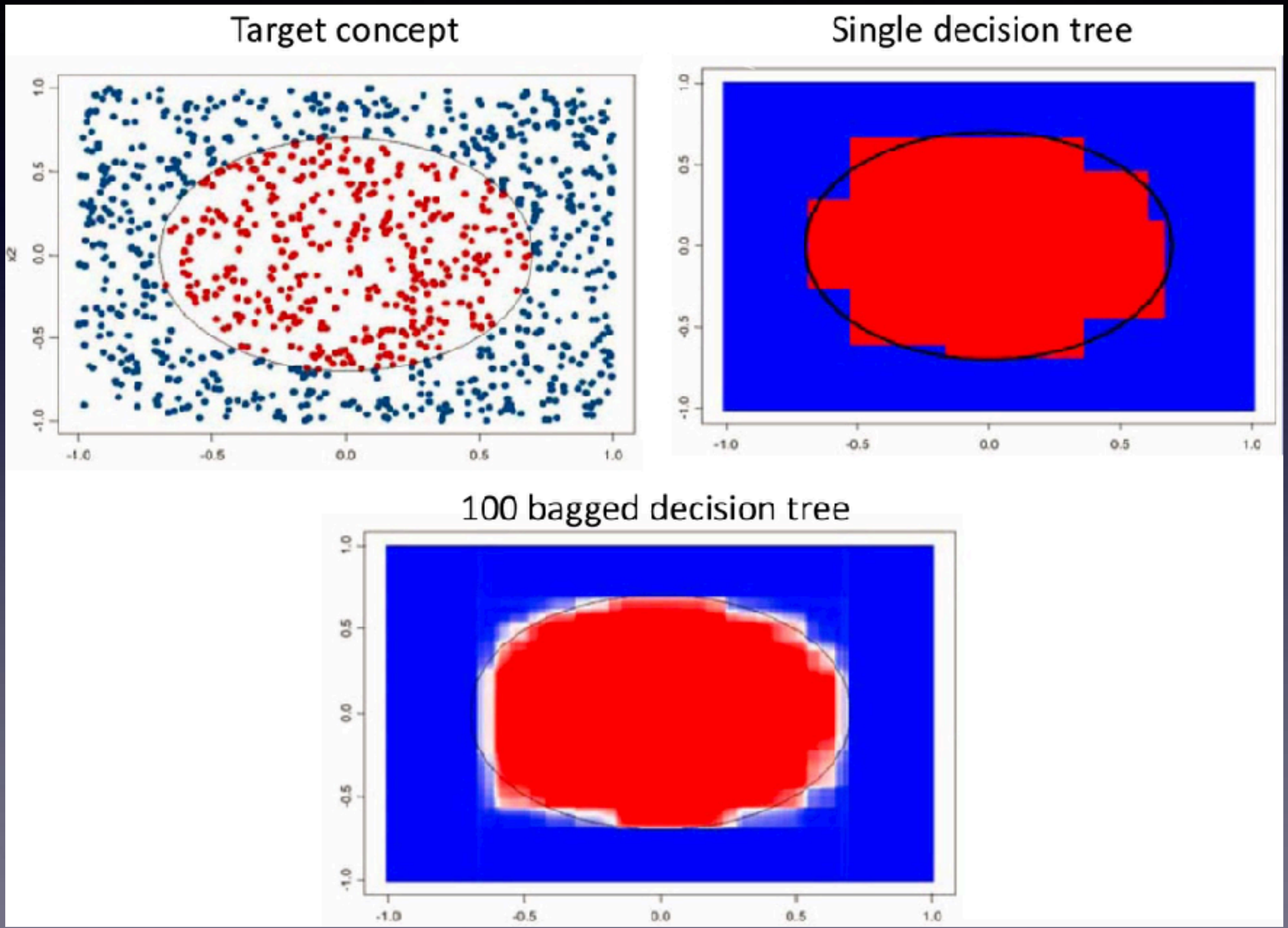
1. Create T bootstrap samples, $\{S_1, ..., S_T\}$ of S as follows:
   - For each Si, randomly draw $|S|$ examples from S with replacement.
   - With large $|S|$, each Si will contain 1 - 1/e = 63.2% unique examples.
2. For each i=1, ..., T, h_i = Learn (S_i)
3. Output H = $<\{h_1, ..., h_T\}$, majority vote >

Leo Breiman, "Bagging Predictors", Machine Learning, 24, 123-140 (1996)

Learning Algorithm Stability

- A learning algorithm is unstable if small changes in the training data produces large changes in the output hypothesis.
- Bagging will have little benefit when used with stable learning algorithms.
- Bagging works best when used with unstable yet relatively accurate classifiers.

100 bagged decision trees



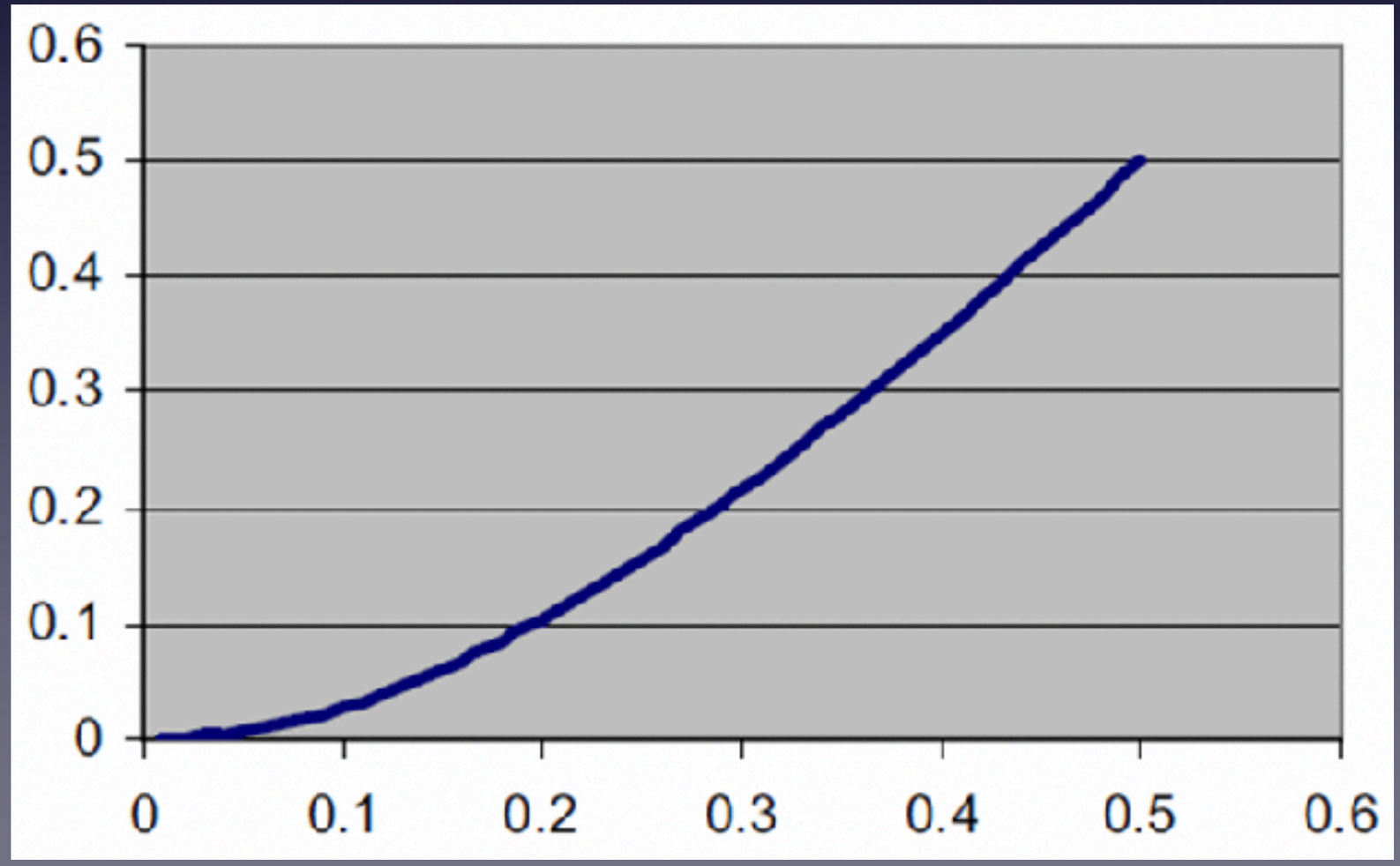
Boosting

- Bagging: individual classifiers are independent
- Boosting: classifiers are learned iteratively
  - Look at errors from previous classifiers to decide what to focus on for the next iteration over data.
  - Successive classifiers depends upon its predecessors.
  - Result: more weights on "hard" examples, i.e., the ones classified incorrectly in the previous iterations.

Error Upper Bound

- Consider $E = <\{h_1, h_2, h_3\}$, majority vote>
- If h_1, h_2, h_3 have error rates less than e, the error rate of E is upper-bounded by g(a): $3e^2-2e^3 < e$

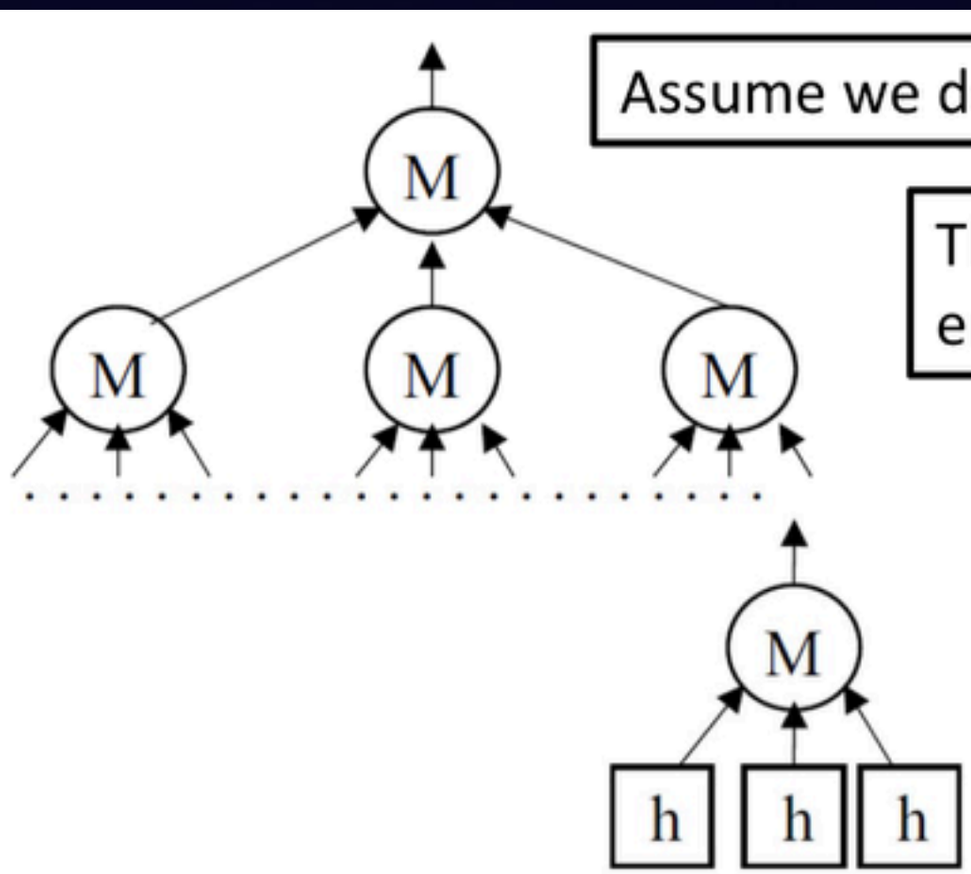
$3e^2-2e^3$



e

Arbitrary Accuracy from Weak Classifiers

- Hypothesis of getting a classifier ensemble of arbitrary accuracy, from weak classifiers.



Assume we desire an error rate less than e.

These need only achieve an error rate less than $g^{-1}(e)$

As we move down the tree, the error we needs to achieve increases according to g^{-1}

Eventually the error rate needed will be attainable by the weak learner

The original formulating of boosting learns too slowly. Empirical studies show that Adaboost is highly effective.

Adaboost

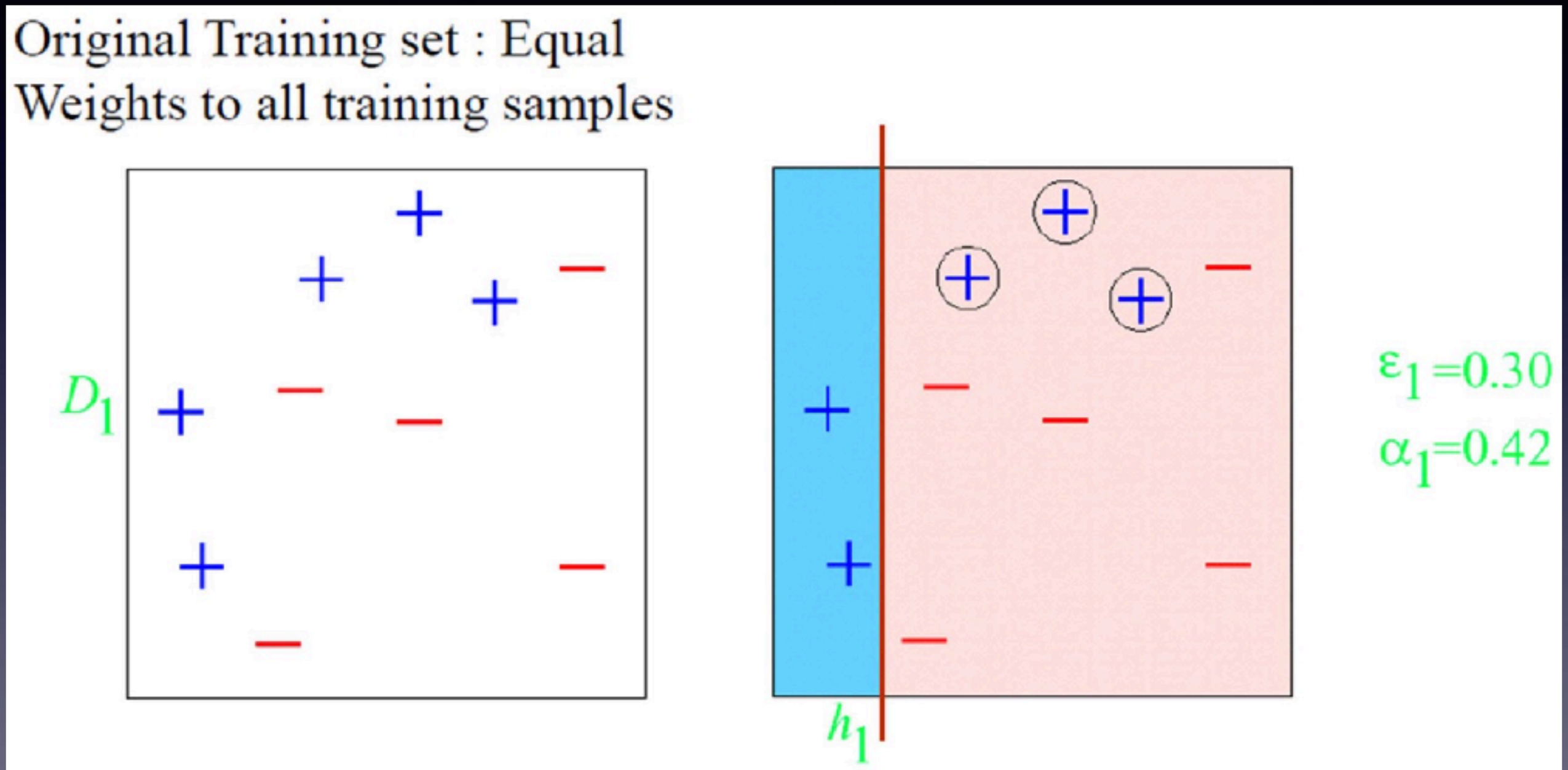
- Adaboost works by learning many times on different distributions over the training data.
- Modify learner to take distribution as input.

  1. For each boosting round, learn on data set S with distribution D_j to produce j^{th} ensemble member h_j.

  2. Compute the $j+1^{th}$ round distribution D_{j+1} by putting more weight on instances that h_j made mistake on.

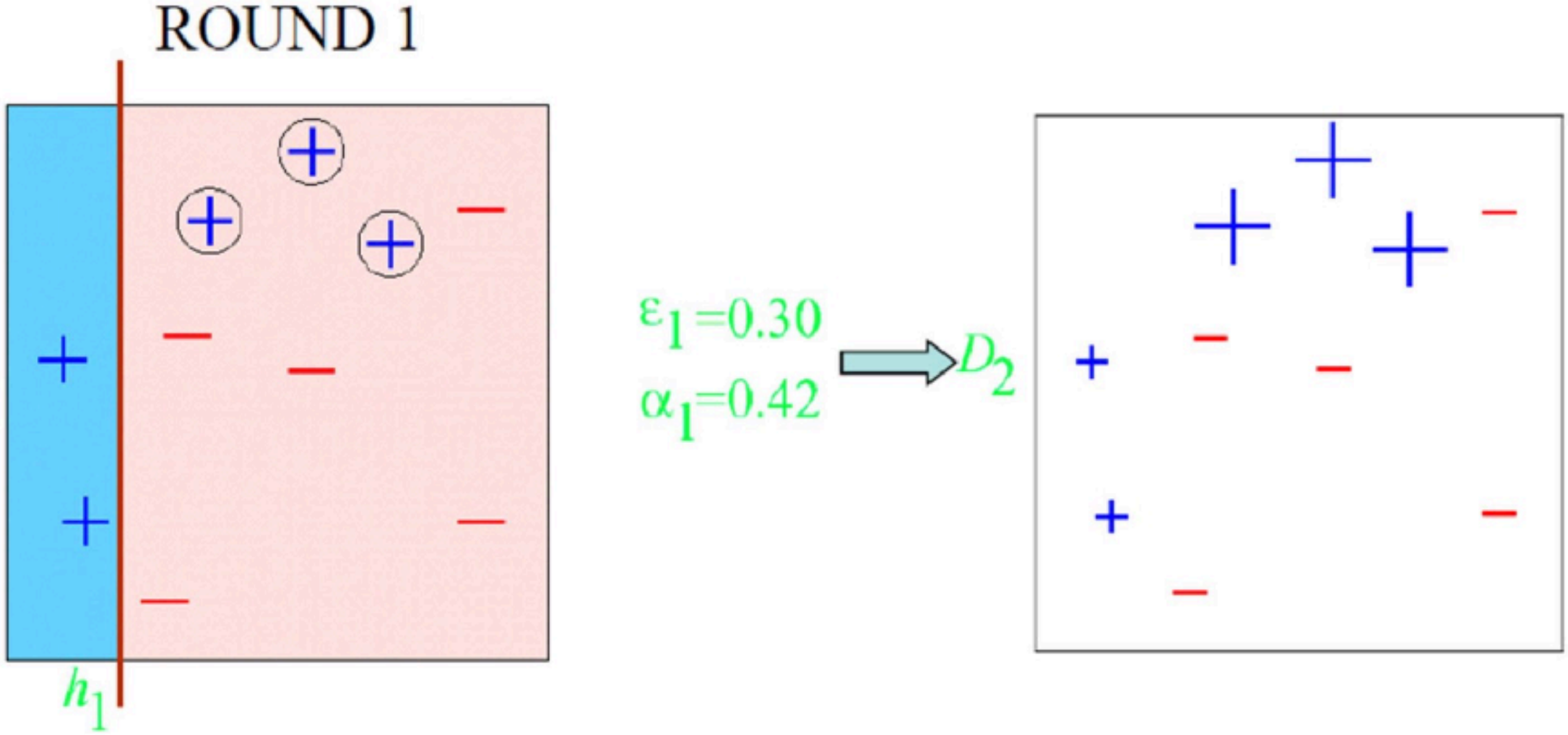
  3. Compute a voting weight w_j for h_j.

Adaboost Example



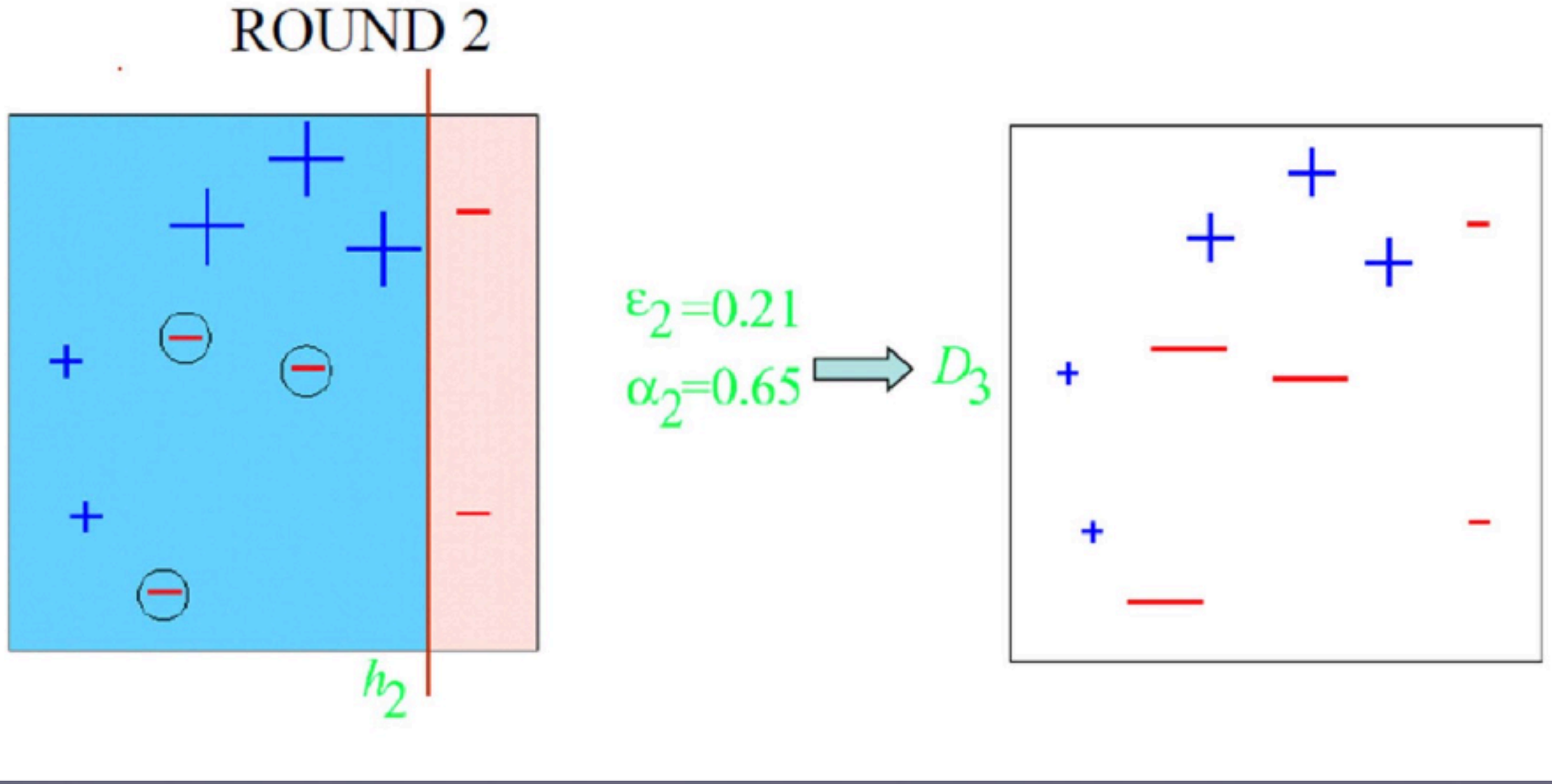
Credit: "A tutorial on boosting" by Yoav Freund and Rob Schapire.

Adaboost Example



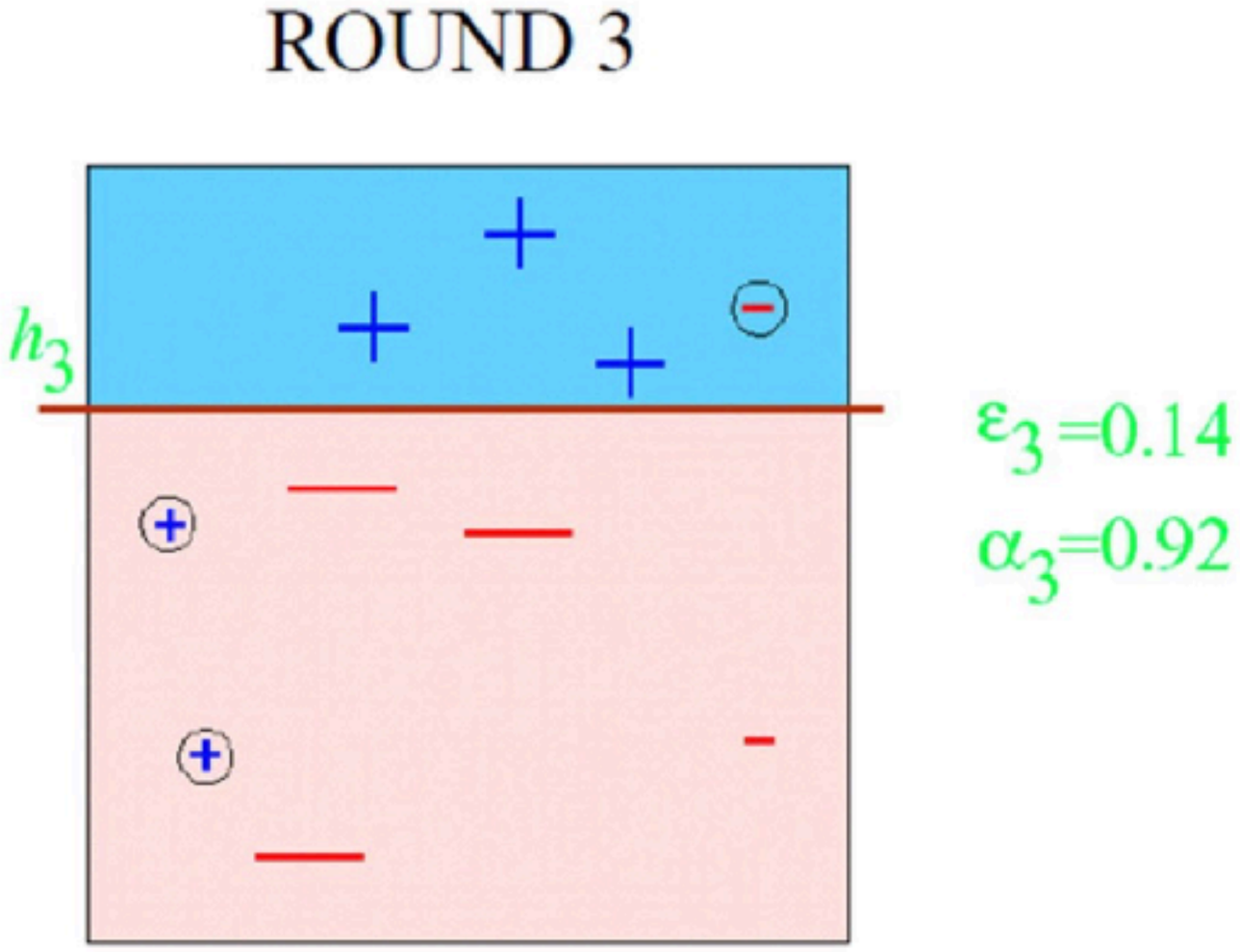
Credit: "A tutorial on boosting" by Yoav Freund and Rob Schapire.

Adaboost Example



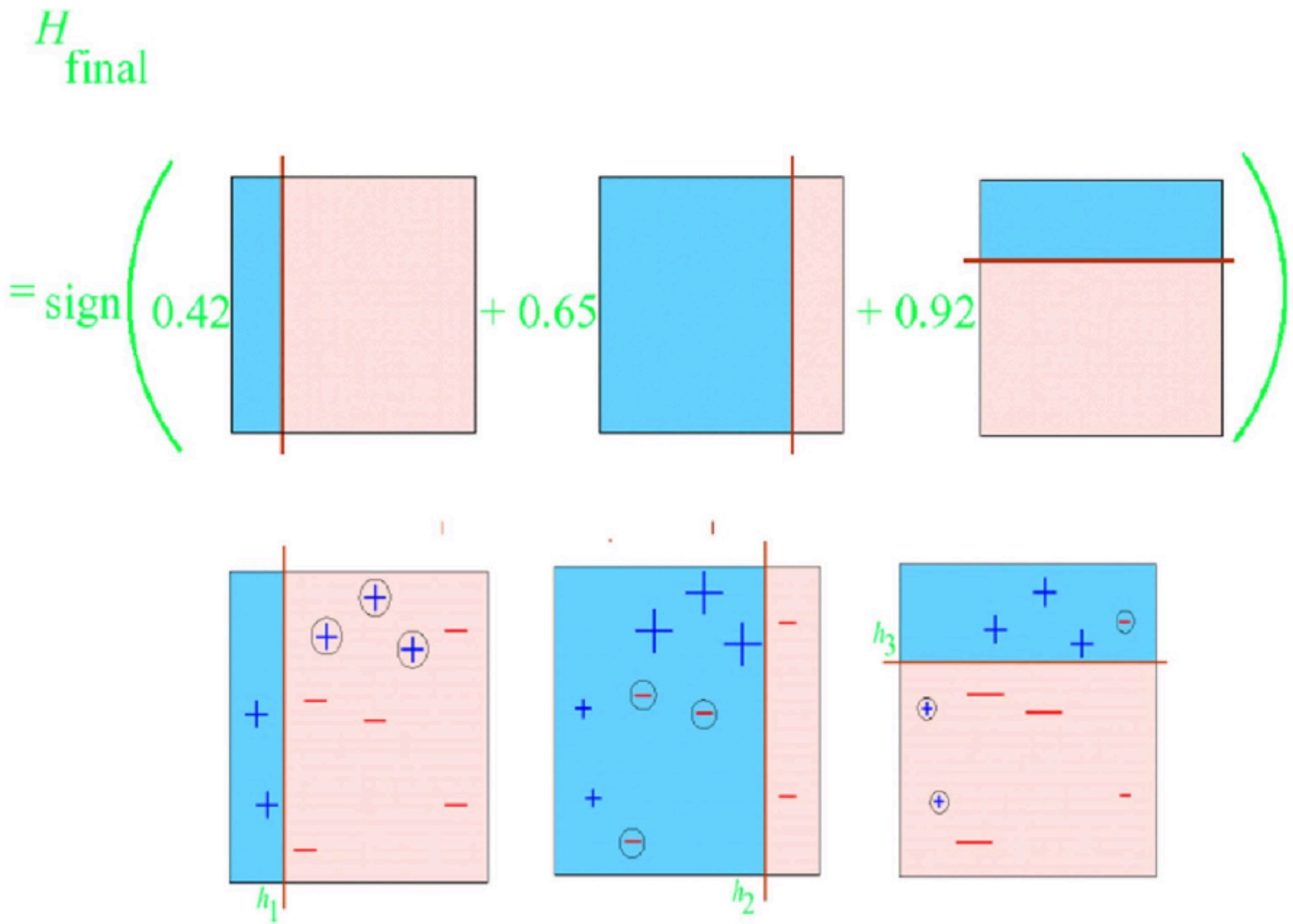
Credit: "A tutorial on boosting" by Yoav Freund and Rob Schapire.

Adaboost Example



Credit: "A tutorial on boosting" by Yoav Freund and Rob Schapire.

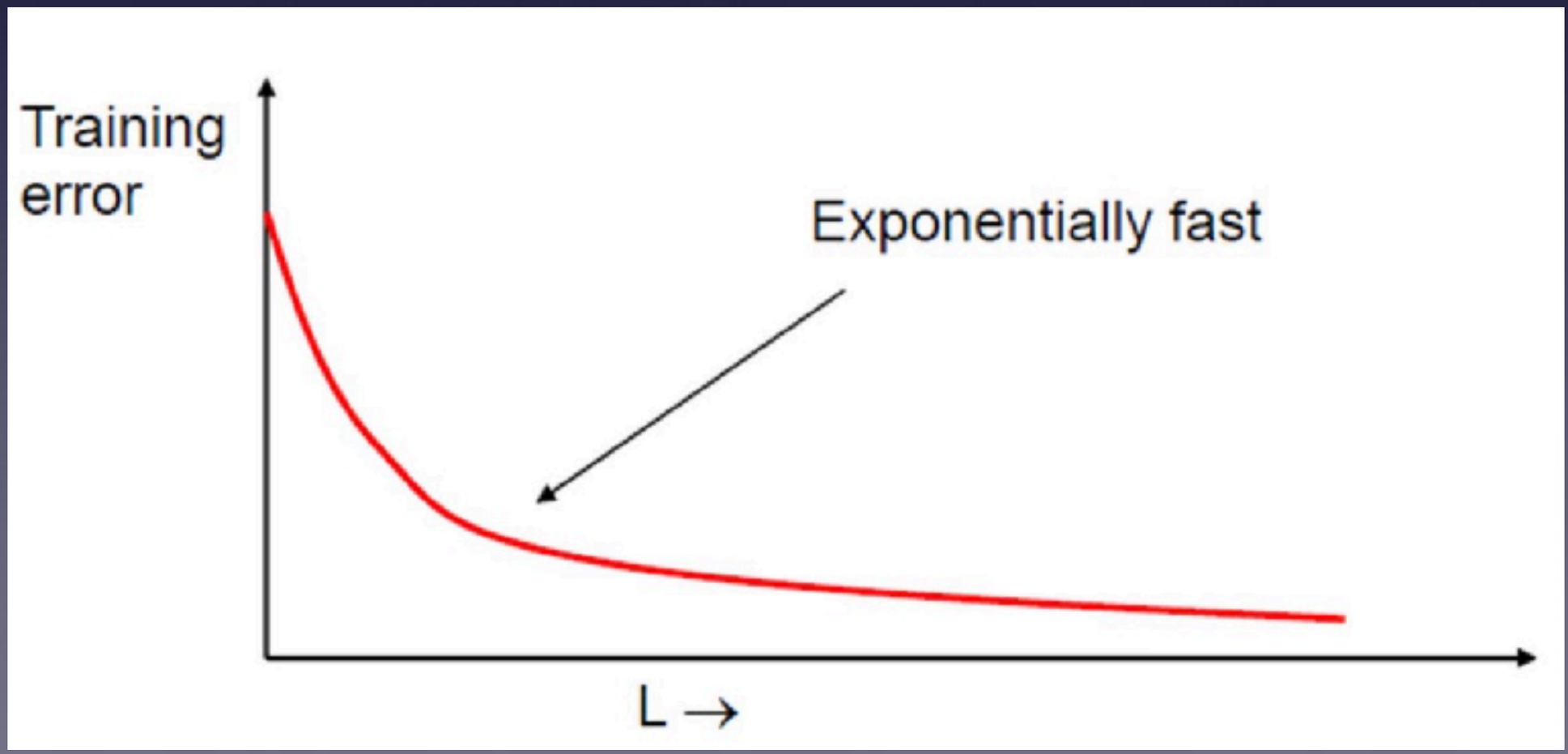
Adaboost Example



Adaboost Properties

- Suppose the base learner L is a weak learner, with error rate slightly less than 0.5 (better than random guess)

- Training error goes to zero exponentially fast!!!



Semi-supervised Learning

Machine Learning Roadmap

unsupervised

supervised

Dimension Reduction

Clustering

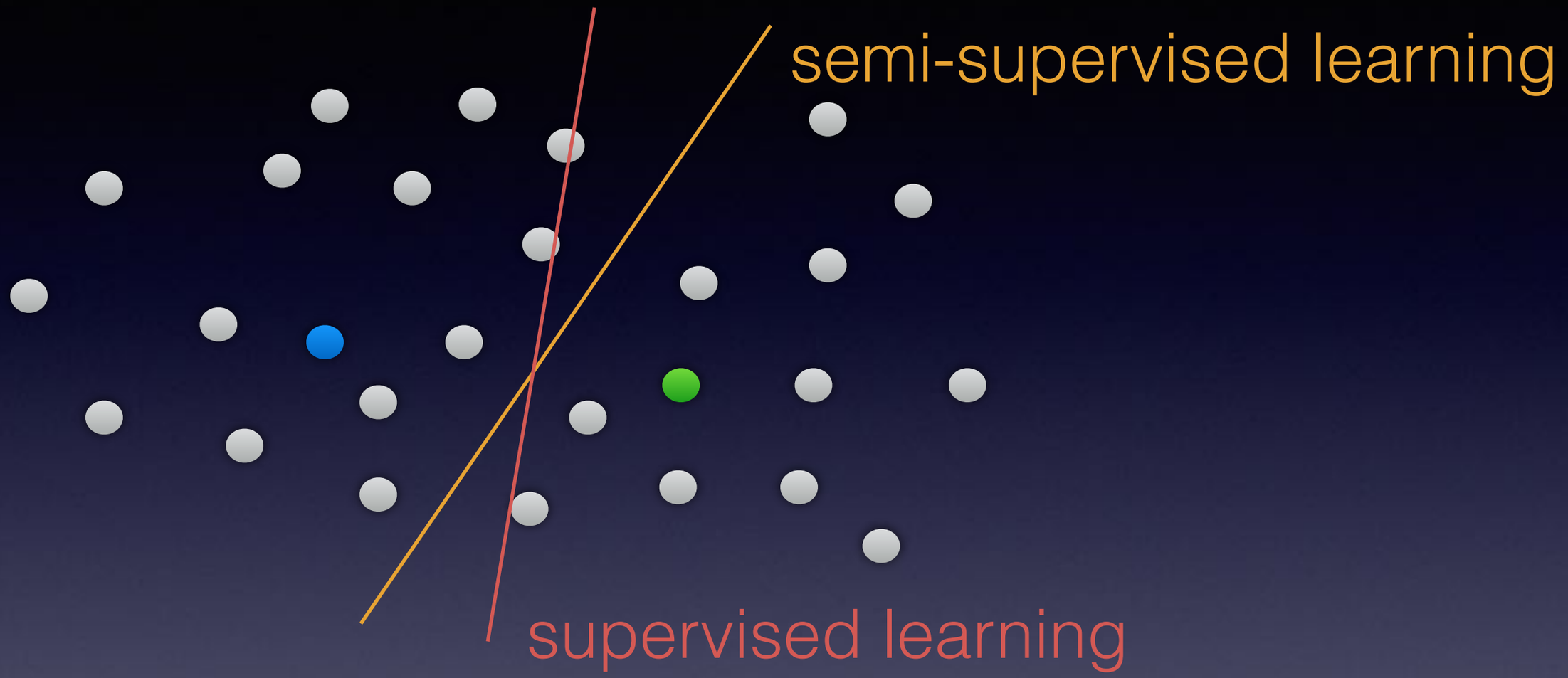
Regression

Classification

continuous
(predicting a quantity)

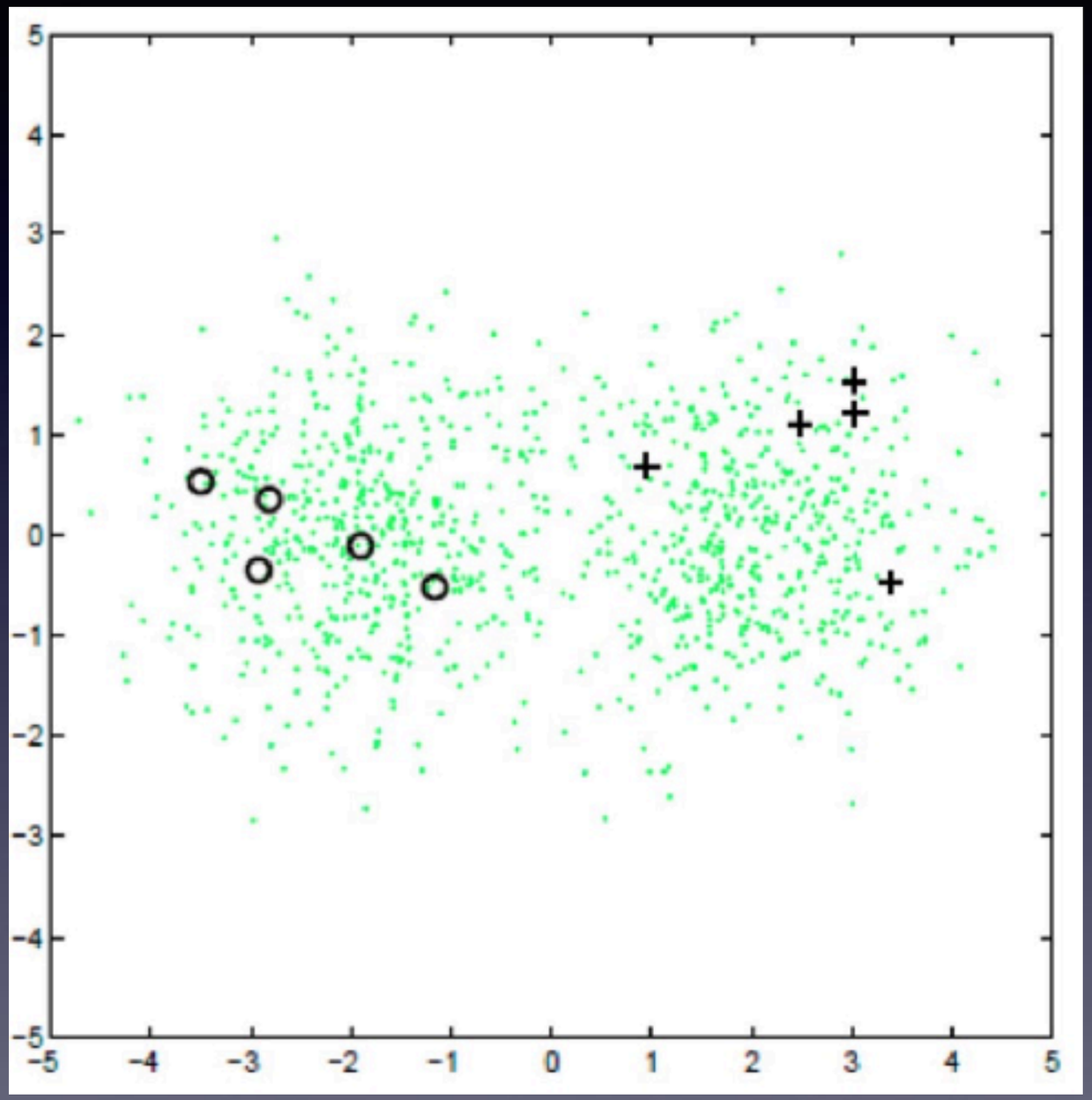
discrete
(predicting a category)

Why can unlabeled data help?



- Assume that class boundary should go through low density areas.
- Having unlabeled data helps getting better decision boundary.

Why can unlabeled data help?

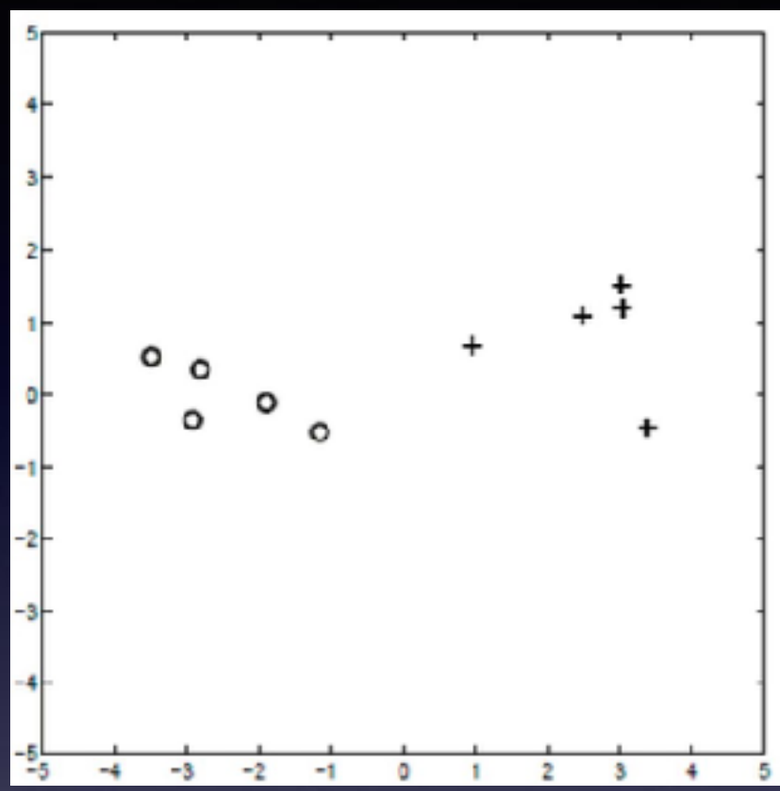


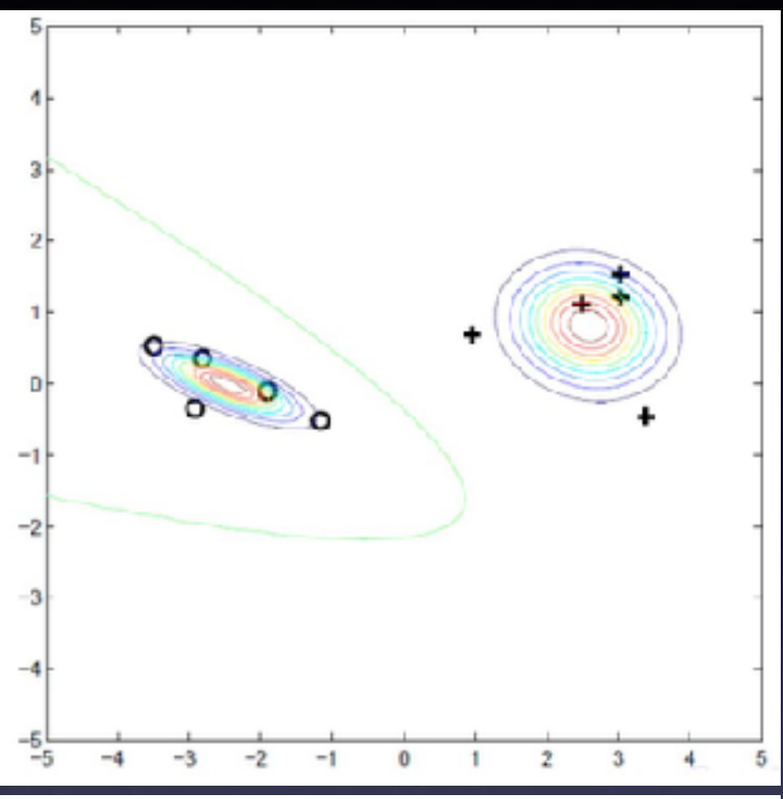
- Assume that each class contains a coherent group of points (e.g., Gaussian)
- Having unlabeled data points can help learn the distribution more accurately.

Semi-Supervised Learning (SSL)

- **Generative models:**
  - Use unlabeled data to more accurately estimate the models.
- **Discriminative models:**
  - Assume that p(y|x) is locally smooth
  - Graph/manifold regularization
- **Multi-view approach:** multiple independent learners that agree on unlabeled data
  - Cotraining

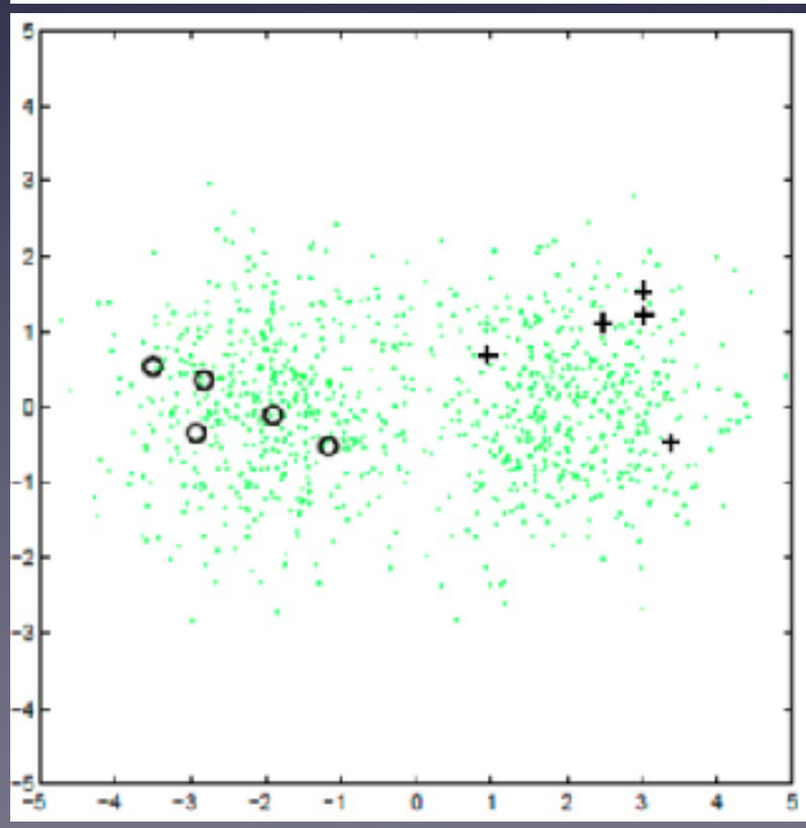
SSL Bayes Gaussian Classifier

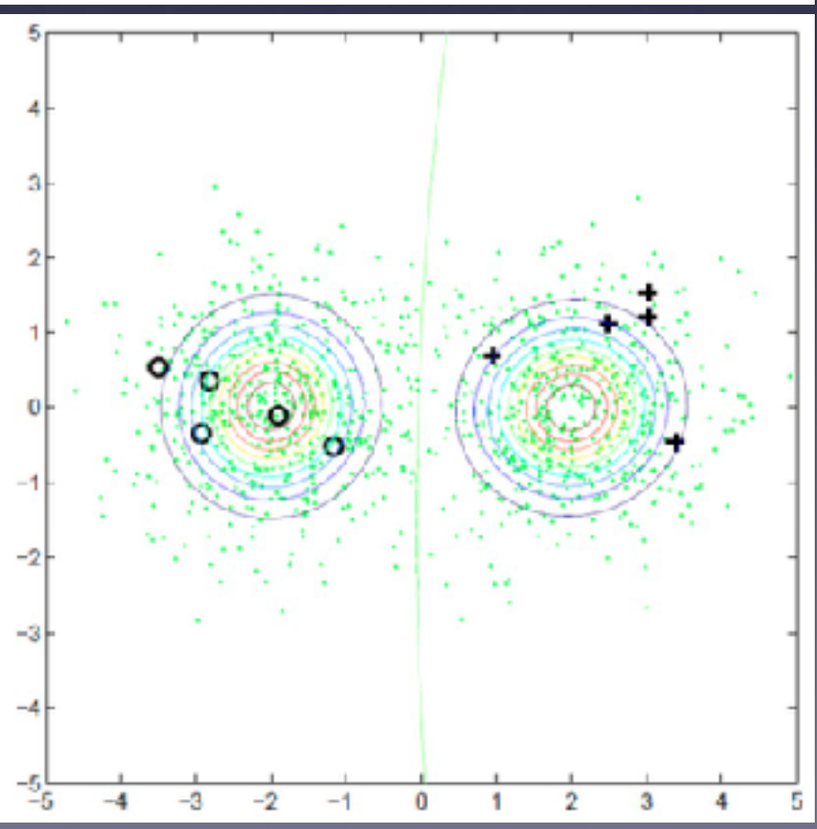




Without SSL: optimize

$$p(X_l, Y_l|\theta)$$





With SSL: optimize

$$p(X_l, Y_l, X_u|\theta)$$

SSL Bayes Gaussian Classifier

- In SSL, the learned θ needs to explain the unlabeled data well, too.
- Find MLE or MAP estimate of joint and marginal likelihood:

$$p(X_l, Y_l, X_u|\theta) = \sum_{Y_u} p(X_l, Y_l, X_u, Y_u|\theta)$$

- Common mixture models used in SSL:
  - GMM
  - Mixture of Multinomials

Estimating SSL GMM params

- Binary classification with GMM using MLE
  - Using labeled data only, MLE is trivial:

$$\log\, p(X_l, Y_l|theta) = \sum_{i=1}^{l} \log\, p(y_i|\theta)\, p(x_i|y_i, \theta)$$

- With both labeled and unlabeled data, MLE is harder---use EM:

$$\log\, p(X_l, Y_l|theta) = \sum_{i=1}^{l} \log\, p(y_i|\theta)\, p(x_i|y_i, \theta)$$

$$+ \lambda \sum_{i=l+1}^{l+u} \log\,(\sum_{y=1}^{2} p(y|\theta)\, p(x_i|y, \theta))$$

Semi-Supervised EM for GMM

- Start with MLE $\theta = \{w, \mu, \Sigma\}_{1:2}$ on (X_l, Y_l)
  - w_c = proportion of class *c*
  - μ_c = sample mean of class c
  - Σ_c = sample covariance of class c

- The E-step: compute the expected label

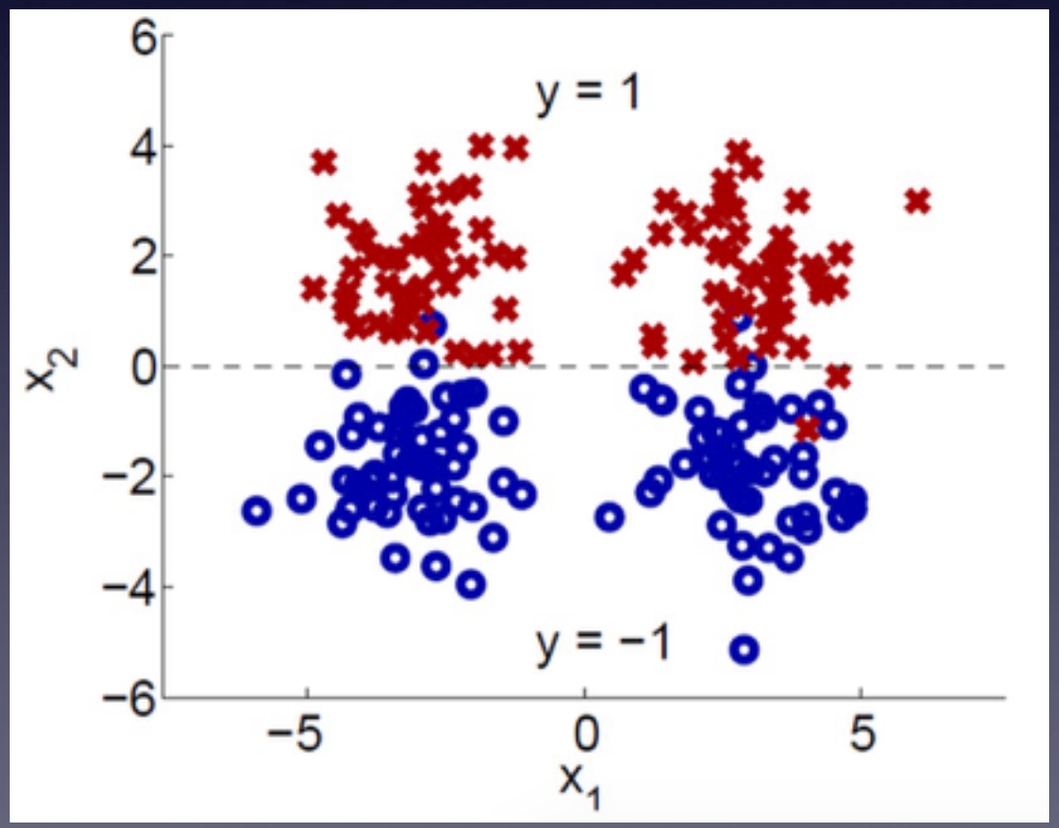
$$p(y|x,\theta) = \frac{p(x,y|\theta)}{\sum_{y'} p(x,y'|\theta)}$$

  for all $x \in X_\mu$.

- The M-step: update MLE θ with (now labeled) X_μ

SSL GMM Discussions

- SSL is sensitive to assumptions!!!
- Cases when the assumption is wrong:



博觀而約取

厚積而薄發

-- 蘇軾 <<稼說>>