Machine learning in computer vision

Lesson 5

Model Evaluation

Metrics for Performance Evaluation

- How to evaluate the performance of a model?

Methods for Performance Evaluation

- How to obtain reliable estimates?

Methods for Model Comparison

- How to compare the relative performance among competing models?

Classification as Binomial Experiment

Classification:

A fixed number of trials: n

Only two outcomes ("success" == $h(\mathbf{x}_i) \neq y_i$)

Probability of success in one trial: $p = error(h|P)$

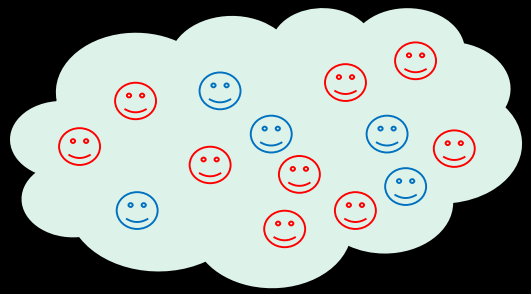
Each trial is independent

$$error(h|X) = \frac{r}{n}$$

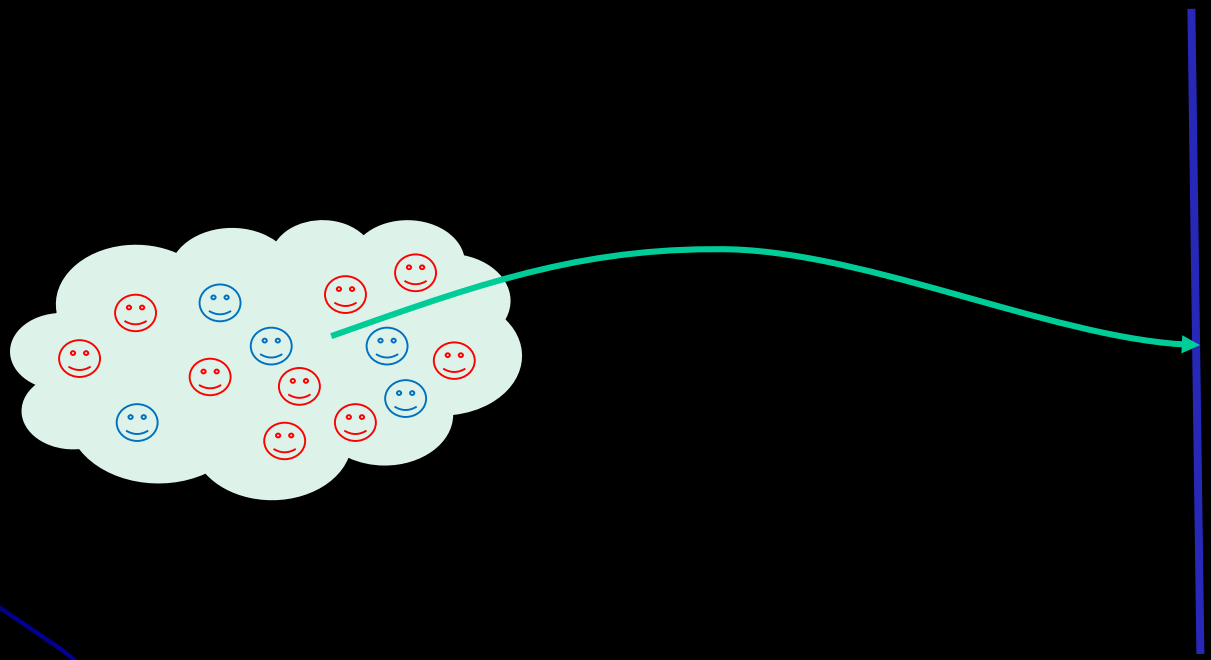
$$E\big(error(h|X)\big) = E\left(\frac{r}{n}\right) = \frac{E(r)}{n} = \frac{np}{n} = p = error(h|P)$$

$\rightarrow$ an unbiased estimator

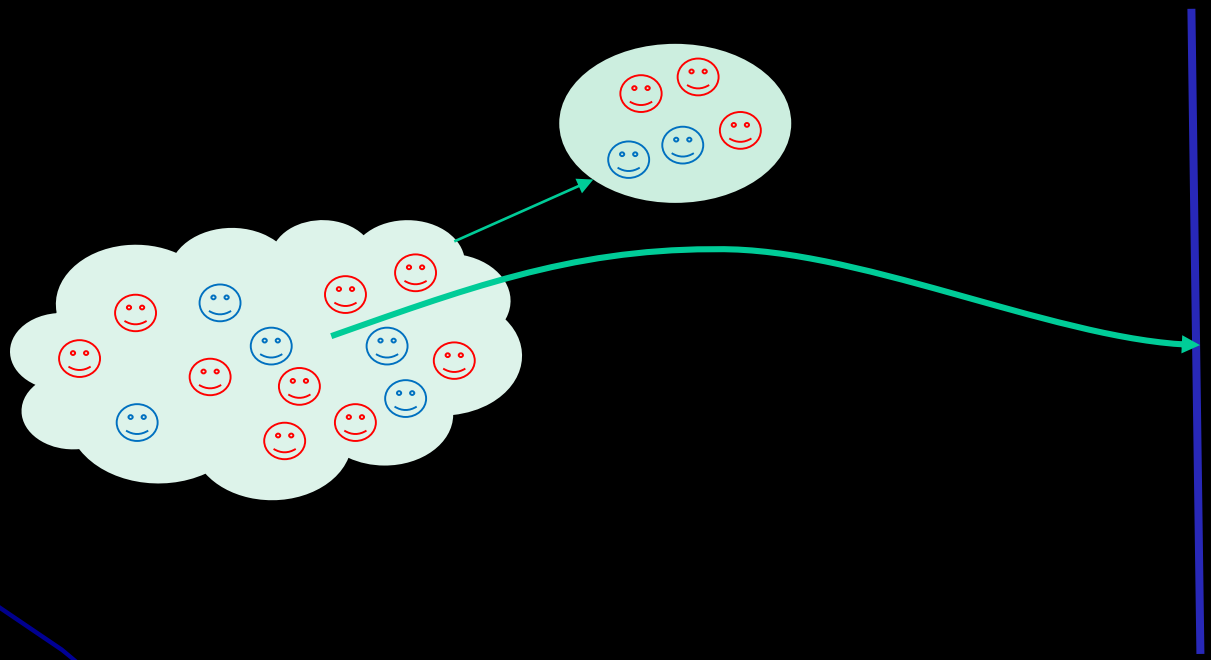
Parameter estimation



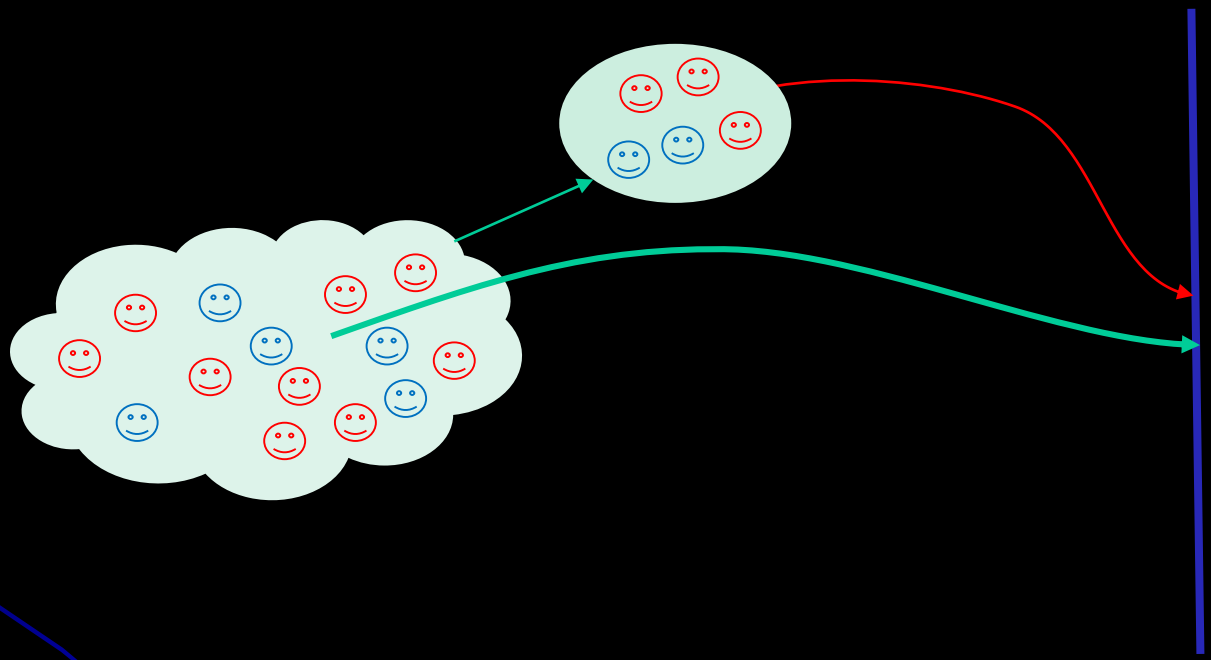
Parameter estimation



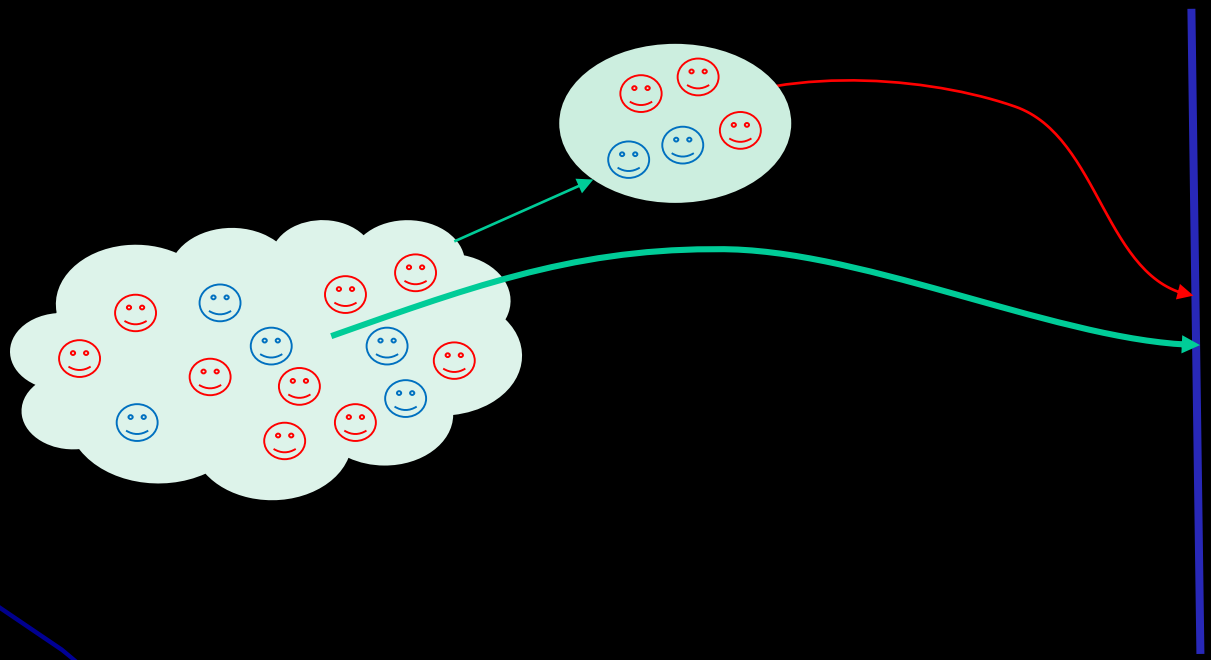
Parameter estimation



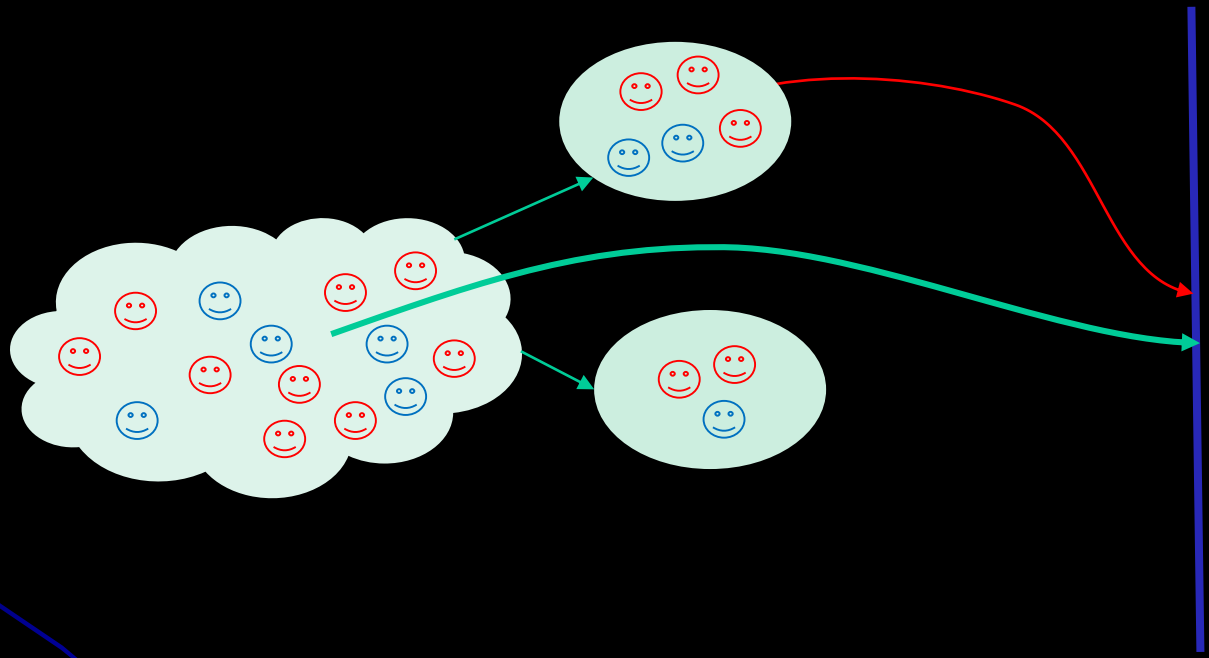
Parameter estimation



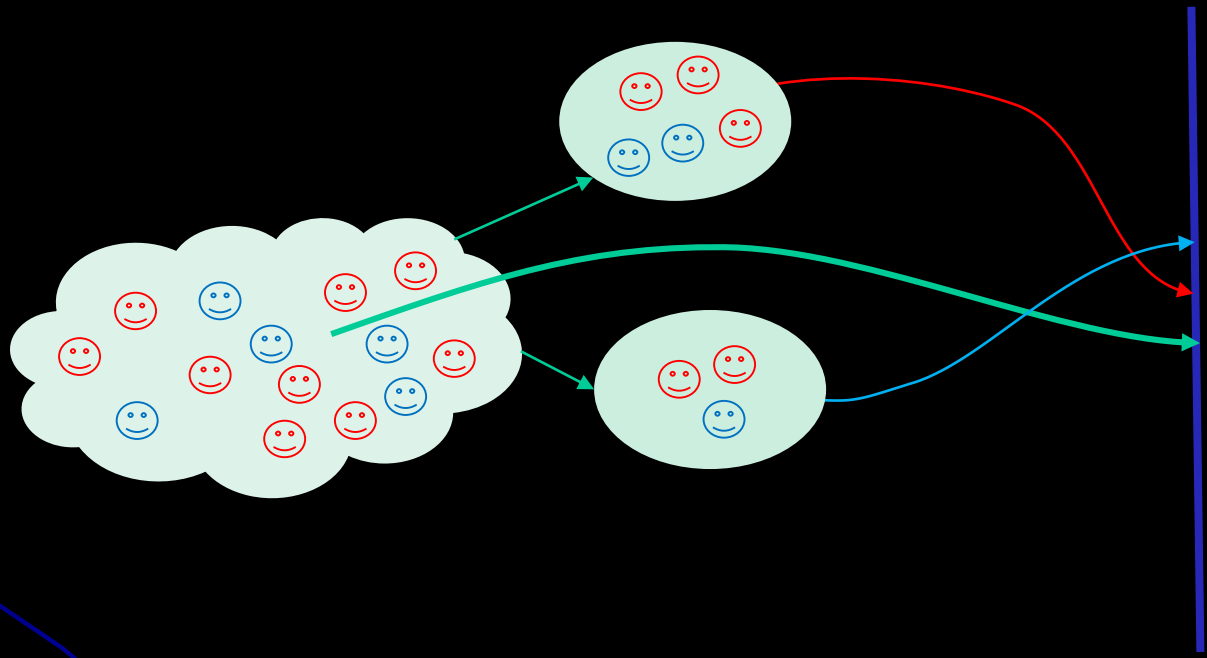
Parameter estimation



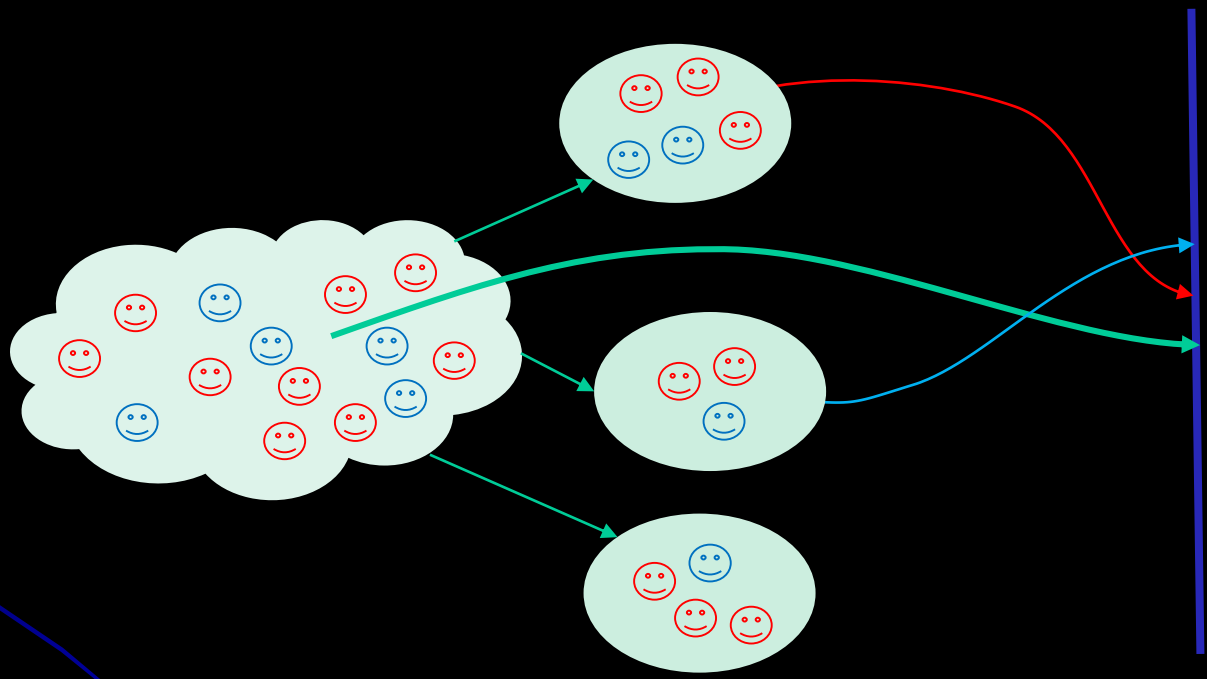
Parameter estimation



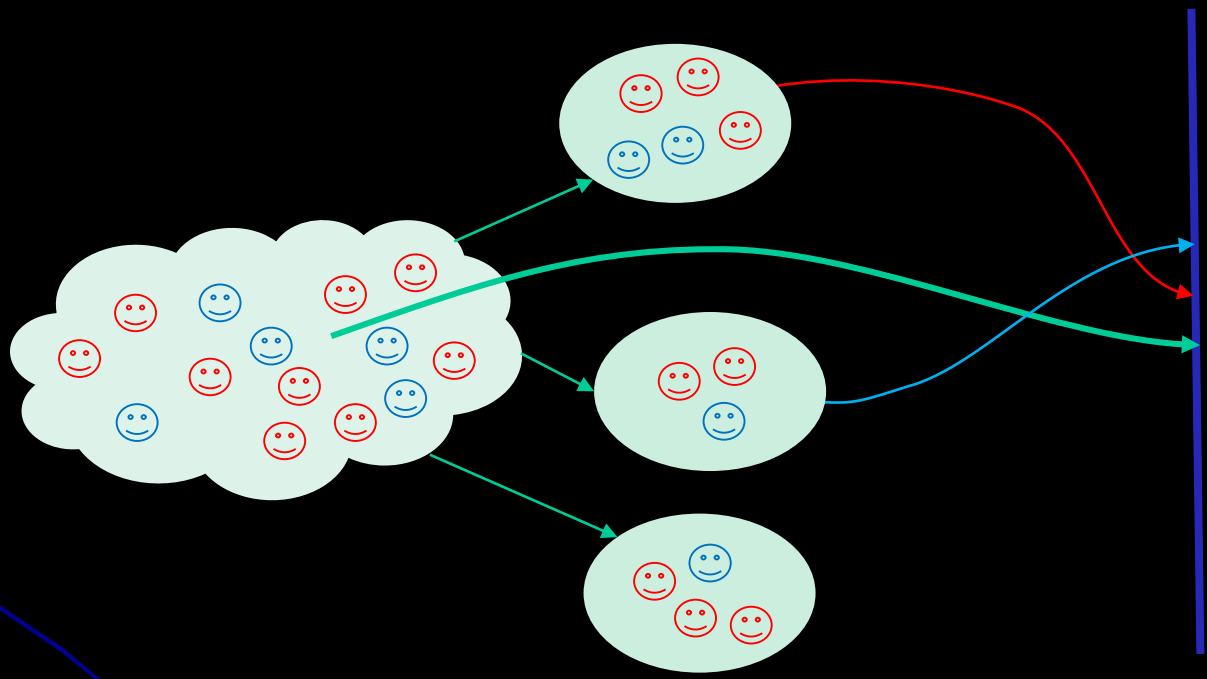
Parameter estimation



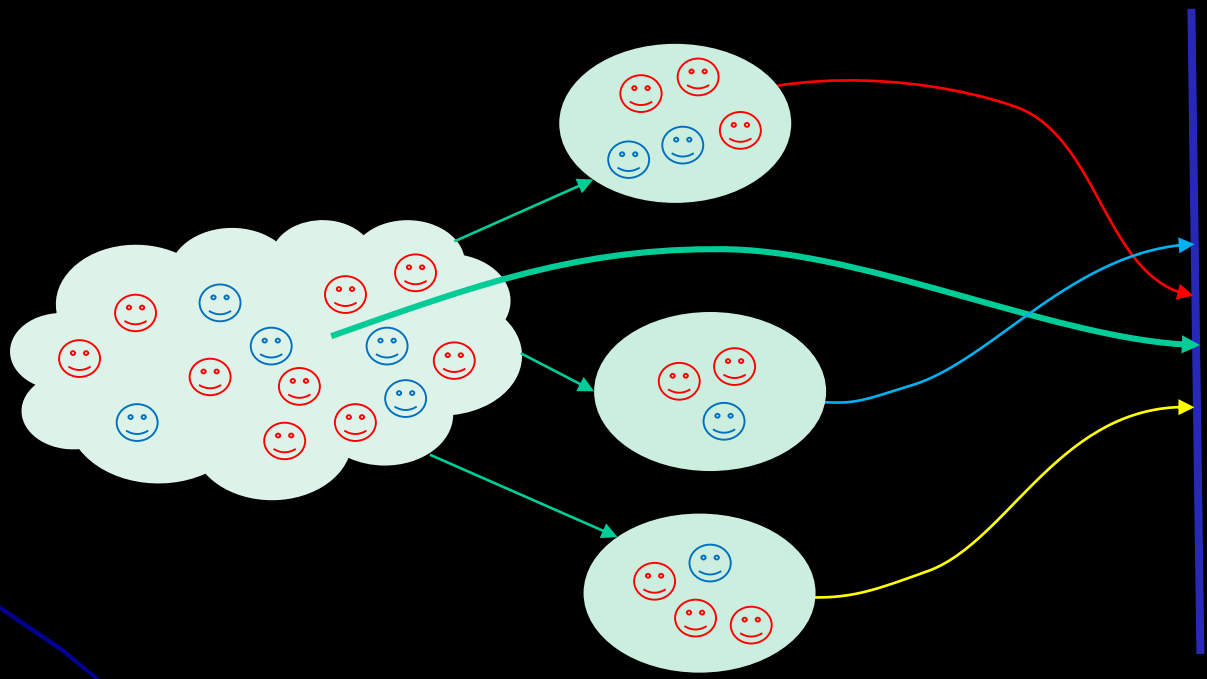
Parameter estimation



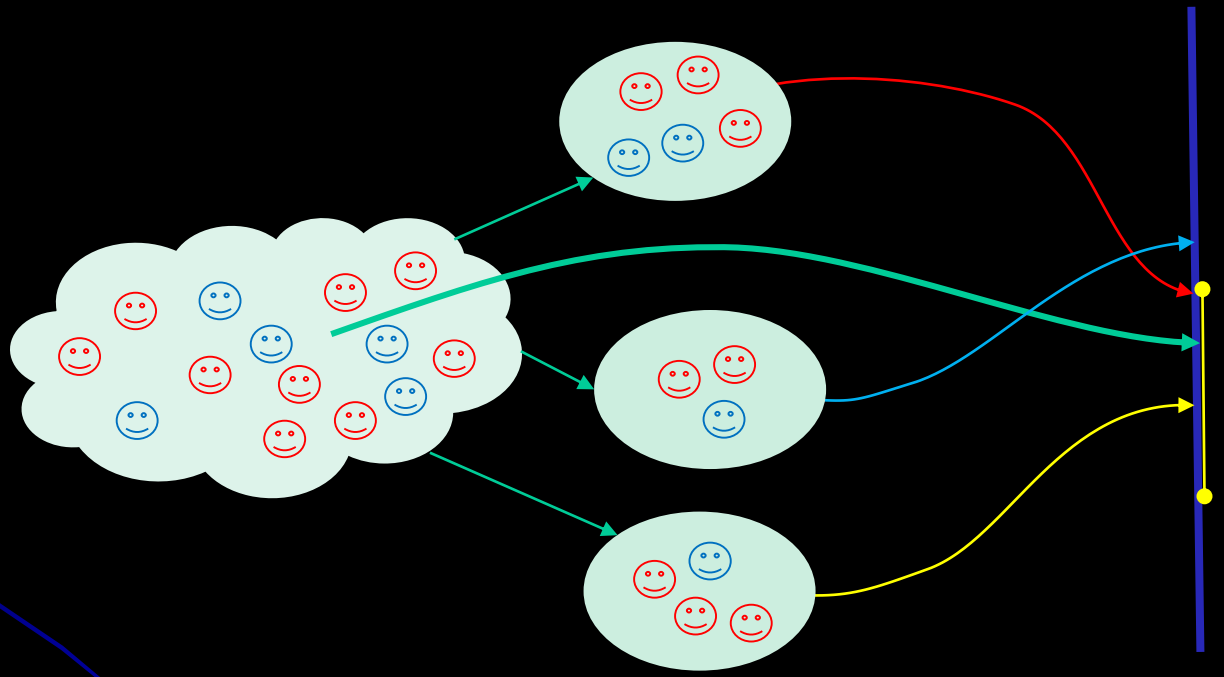
Parameter estimation



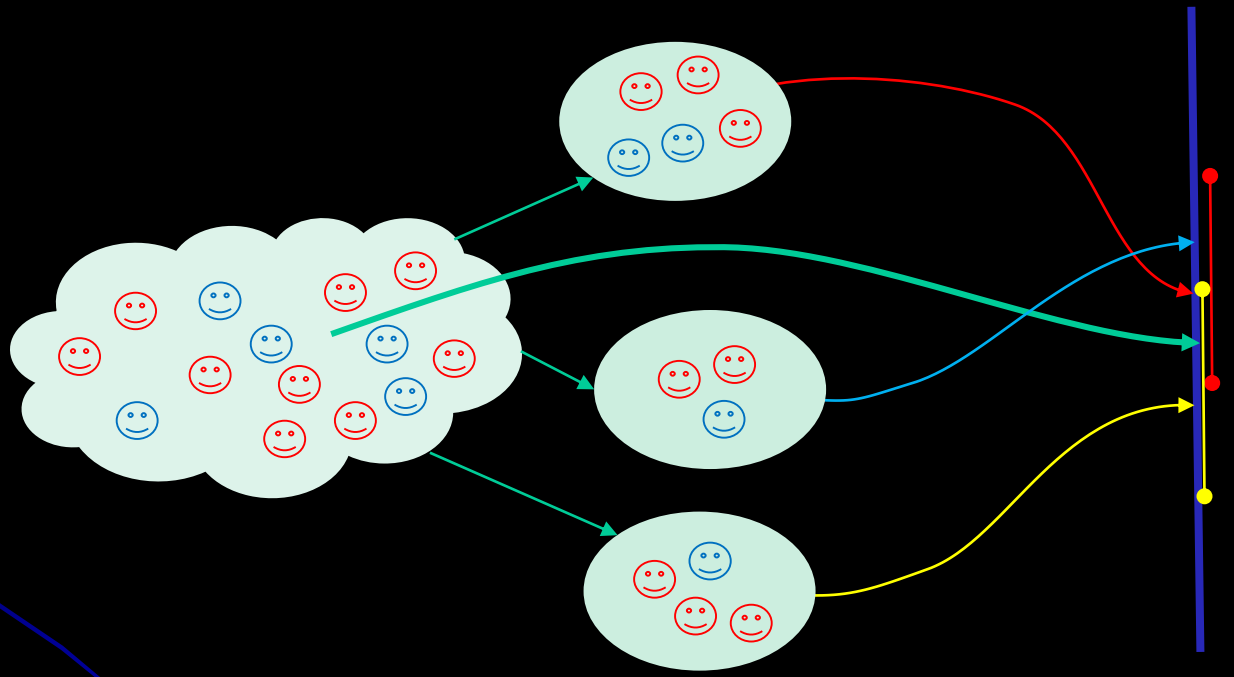
Parameter estimation



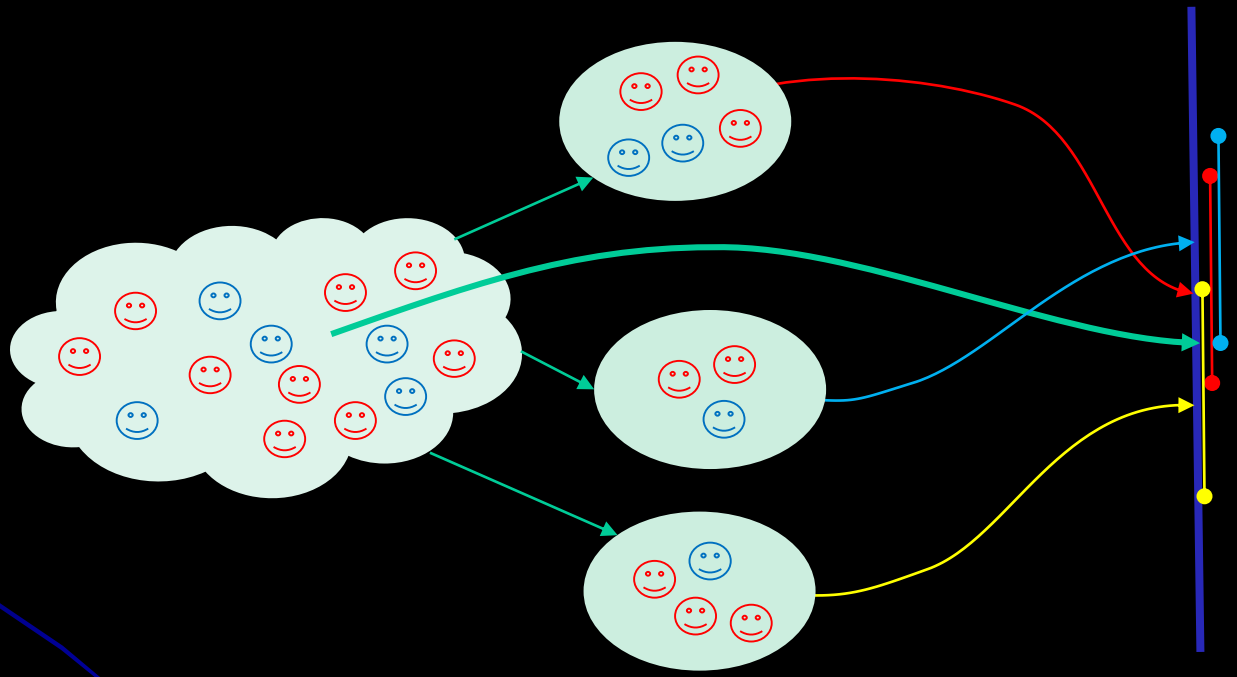
Parameter estimation



Parameter estimation



Parameter estimation



Confidence Intervals

If X contains n examples, drawn independently of *h* and each other and n≥30

Then with approximately 100(1−α)% probability, $error(h|P)$ lies in interval

$$error(h|X) \pm z_{1-\alpha/2}\sqrt{\frac{error(h|X)(1-error(h|X))}{n}}$$

where $z_{1-\alpha/2}$ is the critical value of normal distribution

p	0.999	0.995	0.990	0.975	0.950	0.900
Zp	3.090	2.576	2.326	1.960	1.645	1.282

Confidence Intervals

$error(h|X) = 0.15$

$n = 50$

95% CI? (α=5%)

$$error(h|P) = 0.15 \pm 1.96\sqrt{(0.15 * 0.85)/50}$$
$$= 0.15 \pm 0.099$$

$$error(h|P) \in \langle 0.051, 0.249 \rangle$$

General approach to classification

Training set consists of records with known class labels

Training set is used to build a classification model

A labeled test set of previously unseen data records is used to evaluate the quality of the model

The classification model is applied to new records with unknown class labels

The three-way split

Training set

A set of examples used for learning

Validation set

A set of examples used to tune the parameters of a classifier

Test set

A set of examples used only to assess the performance of fully-trained classifier. After assessing the model with the test set, YOU *MUST NOT* further tune your model (in order to prevent ‘learning the test set’ and ‘overfitting’)

Hold out

Problems:

For small or “unbalanced” datasets, instances might not be representative.

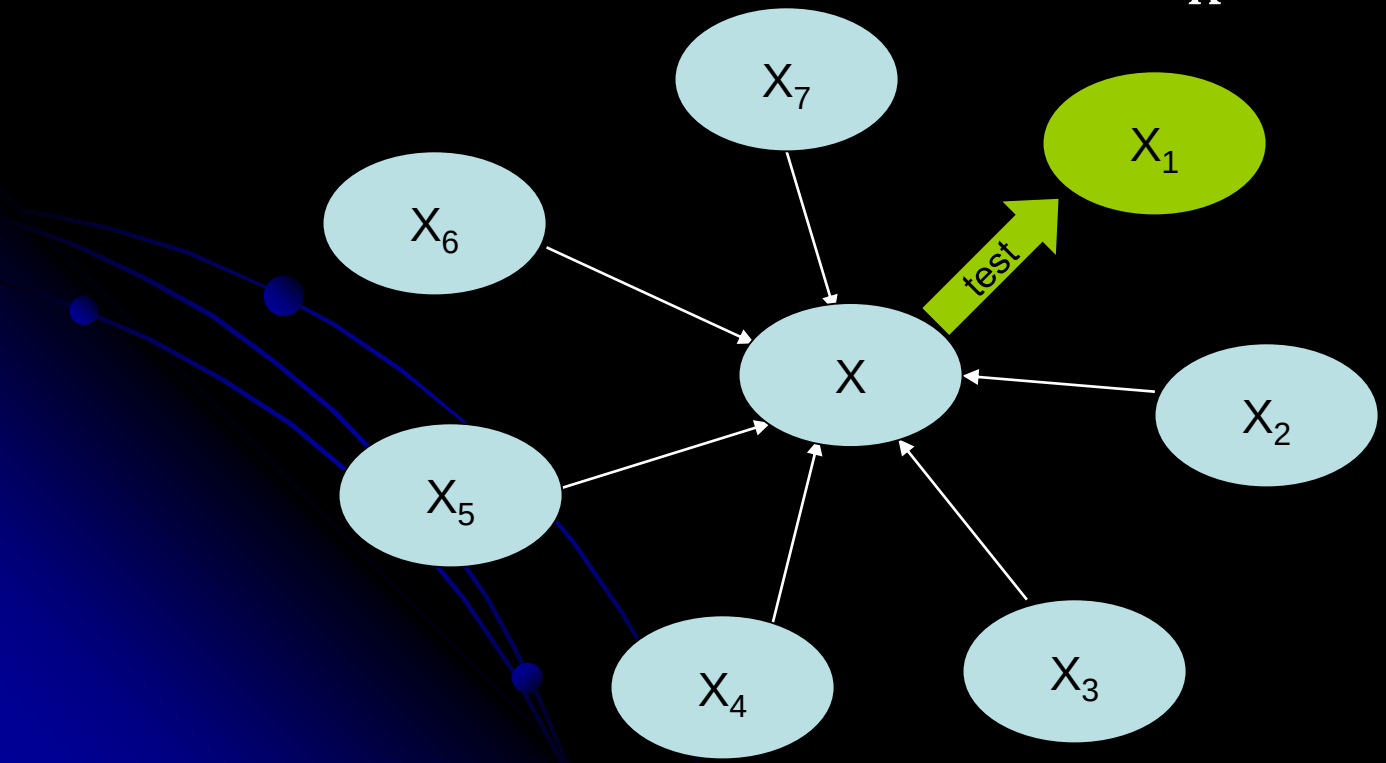
The data used for training and testing may vary significantly.

Solution:

Generate new subsets of instances with an approximately equal proportions of classes.

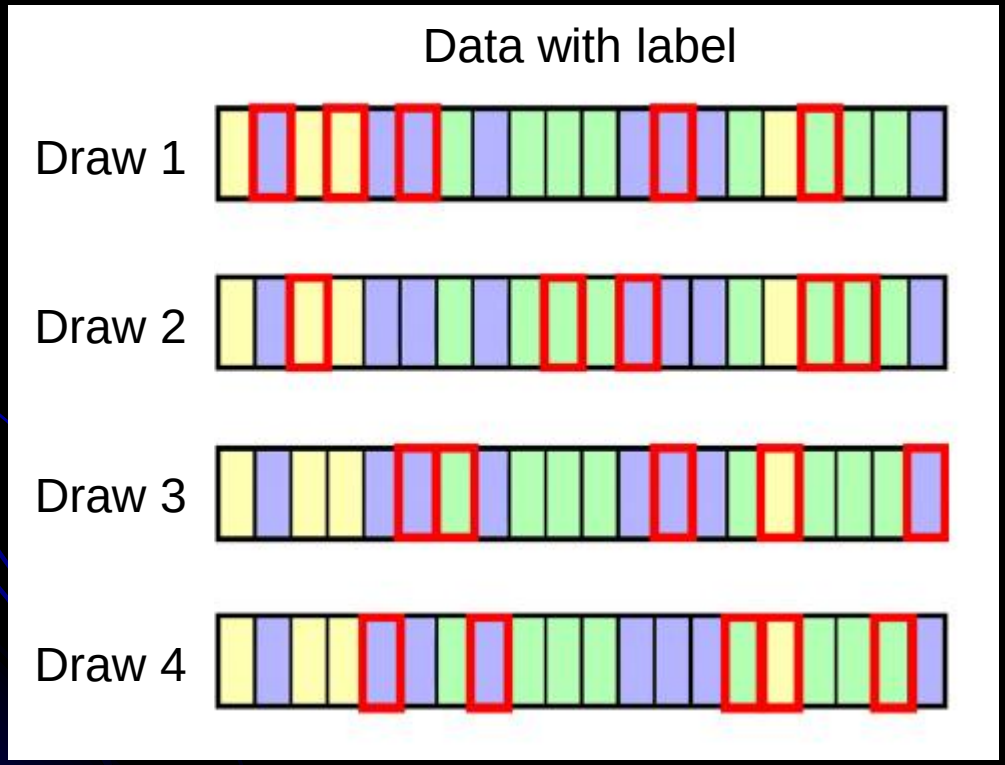
Cross validation

Divide data into K subsets $\{X_1, \ldots X_K\}$
Use each subset for error estimation
Compute average error $E = \frac{1}{K}\sum_{k=1}^{K} E_k$



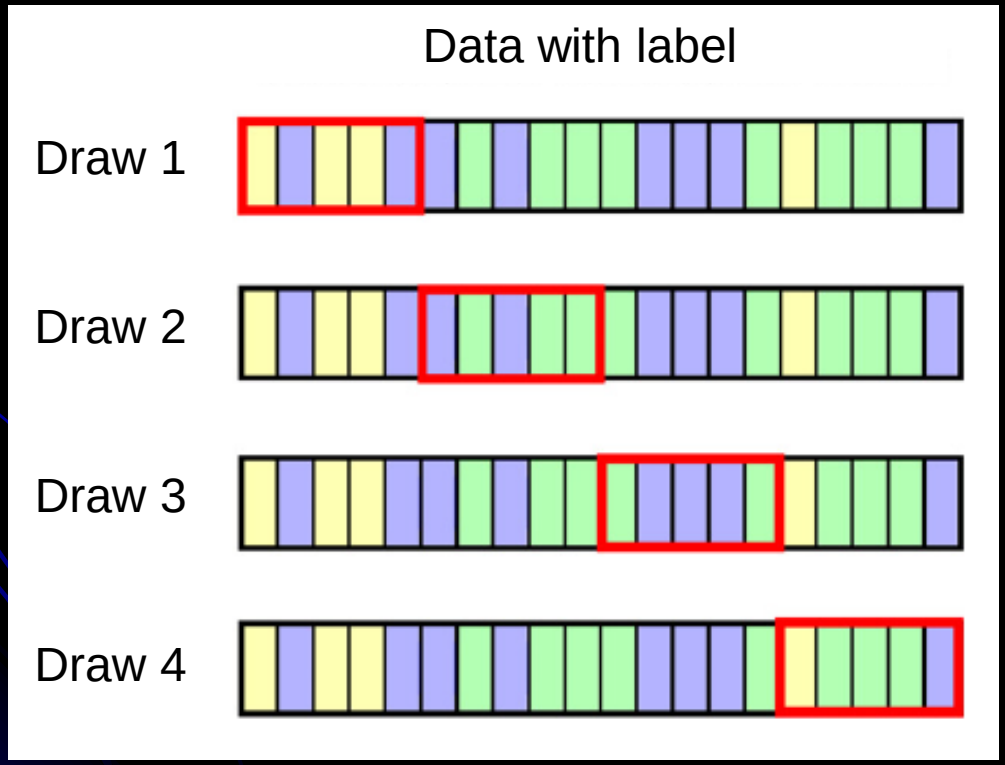
Random subsampling

Random sampling of test data (no return)
But can be used in several draws



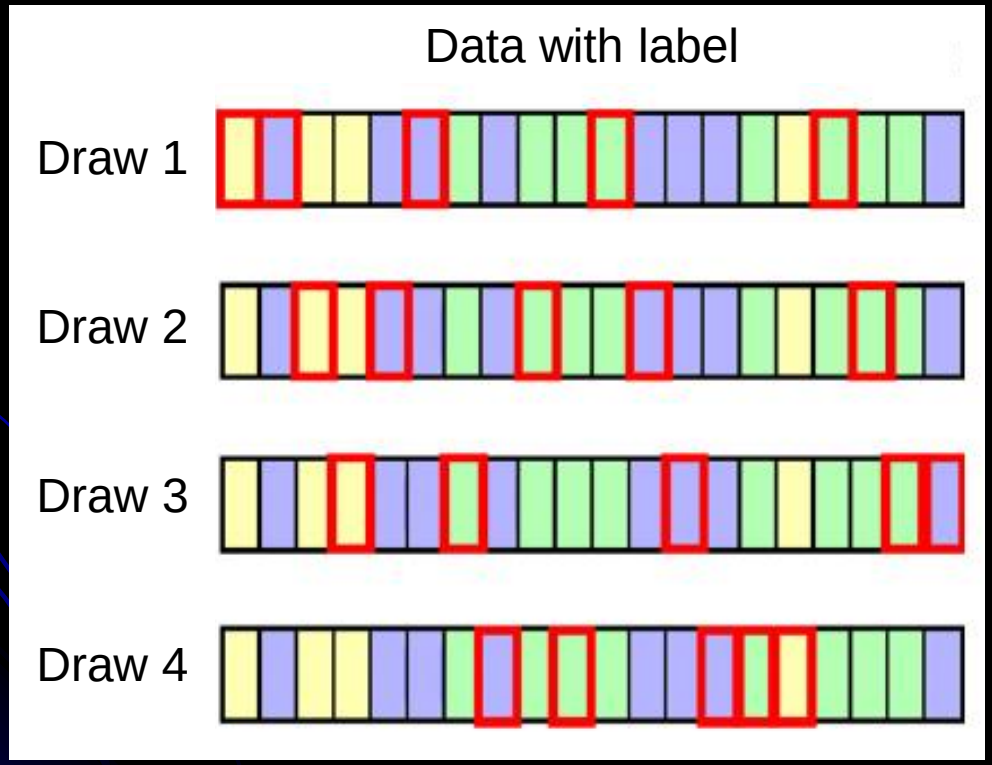
K-fold cross validation

Divide into *K* equally sized subsets



Stratified K-fold cross validation

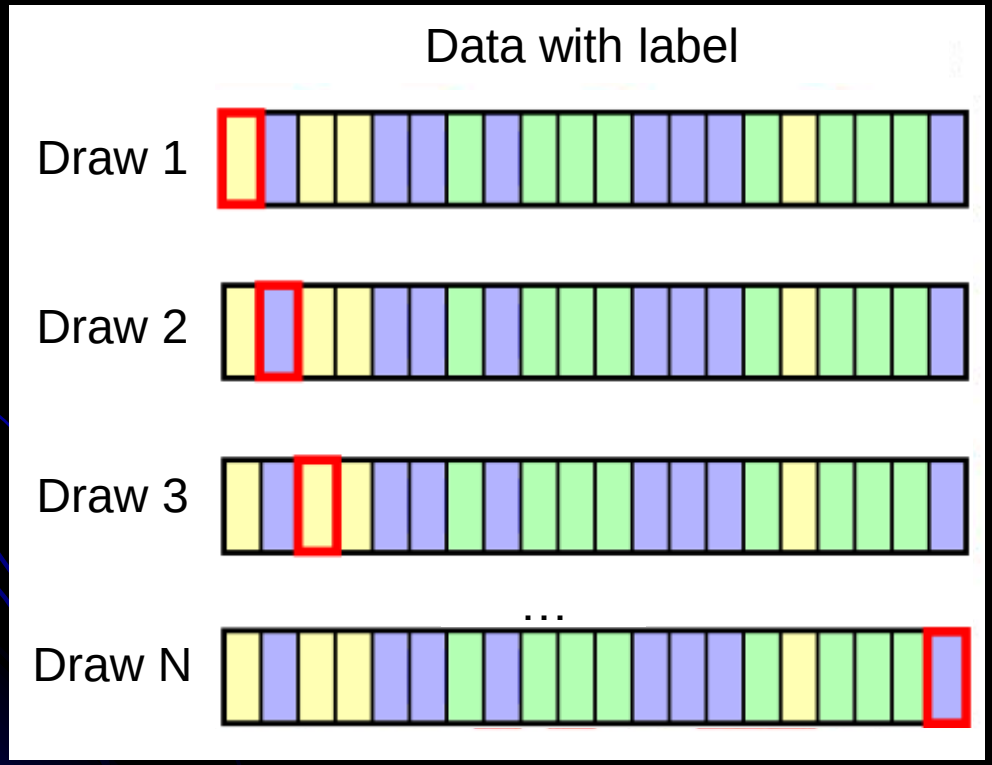
Create K subsets that follow the class distribution of the whole set



Leave one out

Test on one sample only

K=N



Choice of K

Mostly K=10

For large datasets, K=3 is enough

For small datasets, leave one out

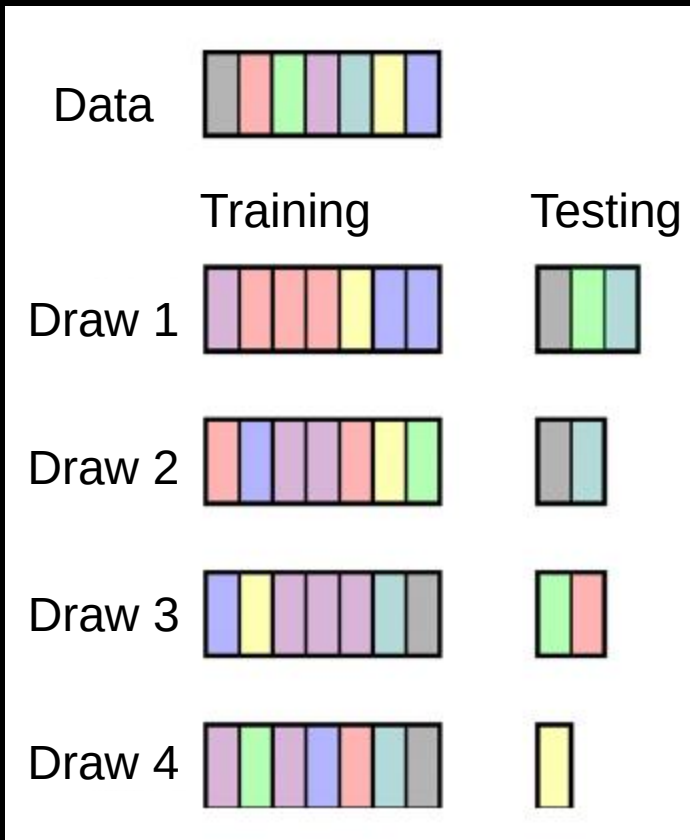
Bootstrap

Sample a dataset of *N* instances *N* times *with replacement* to form a new dataset of *N* instances – the training set

Use the instances from the original dataset that do not occur in the new training set for testing

$E = \frac{1}{K}\sum_{k=1}^{K} E_k$

CV – sampling w/o replacement



0.632 Bootstrap

A particular instance has a probability of $1 - \frac{1}{N}$ of *not* being picked

The probability of ending up in the test data:

$$\left(1 - \frac{1}{N}\right)^N \approx e^{-1} = 0.368$$

This means the training data will contain approximately 63.2% of the instances

0.632 Bootstrap

$E = \frac{1}{K}\sum_{k=1}^{K} E_k$ tends to be pessimistic estimate

Possible solution:

$E = 0.632 * \frac{1}{K}\sum_{k=1}^{K} E_k + 0.368 * E_{data}$

E_{data} – error when trained and tested on all available data

Model Evaluation

Metrics for Performance Evaluation

- How to evaluate the performance of a model?

Methods for Performance Evaluation

- How to obtain reliable estimates?

Methods for Model Comparison

- How to compare the relative performance among competing models?

Difference in Error of two Hypotheses

h_1 has been tested on a sample S_1 containing n_1 randomly drawn examples

h_2 has been tested on an independent sample S_2 containing n_2 examples drawn from the same distribution

Estimate: $\hat{d} = error(h_1|S_1) - error(h_2|S_2)$

Then with approximately 100(1−α)% probability,

$d = error(h_1|P) - error(h_2|P)$ lies in interval

$$\hat{d} \pm z_{1-\alpha/2}\sqrt{\frac{e_1(1-e_1)}{n_1} + \frac{e_2(1-e_2)}{n_2}},$$

where $e_i = error(h_i|S_i)$

Difference in Error of two Hypotheses

We can test the hypothesis $d = 0$:

Method 1: If $0 \notin CI$, than reject H_0 (there should be a difference between the two algorithms)

Method 2: If $\left| \frac{\hat{d}}{\sqrt{\frac{e_1(1-e_1)}{n_1} + \frac{e_2(1-e_2)}{n_2}}} \right| > z_{1-\alpha/2}$

than reject H_0

Paired t-test

Divide data into subsets T_1,..., T_K with $|T_i| > 30$

On each subset compute

$\delta_i = error(h_1|T_i) - error(h_2|T_i)$

Now compute: $\bar{\delta} = \frac{1}{K}\sum_{k=1}^{K} \delta_k$ and $SE(\bar{\delta}) = \sqrt{\frac{\sum_{k=1}^{K}(\delta_k - \bar{\delta})^2}{K(K-1)}}$

100(1−α)% confidence interval of the true difference

$\bar{\delta} \pm t_{1-\alpha/2,K-1} SE(\bar{\delta})$

where $t_{1-\alpha/2,K-1}$ is the critical value of student distribution with K-1 DOF

Paired t-test

We can test the hypothesis $d = 0$:

Method 1: If $0 \notin CI$, than reject H_0 (there should be a difference between the two algorithms)

Method 2: If $\left|\frac{\bar{\delta}}{SE(\bar{\delta})}\right| > t_{1-\alpha/2,K-1}$

than reject H_0

Difference in Error of two Classifiers

Divide data into subsets T_1,..., T_k with $|T_i| > 30$

Create training sets $S_i = X \setminus T_i$

Train models $h_j = L_j(S_i)$

Continue comparing two models

! Samples are not independent

Use heuristic test: e.g. corrected resampled t-test

$$SE(\bar{\delta}) = \sqrt{\left(\frac{1}{K} + \frac{N_{test}}{N_{train}}\right) \frac{\sum_{k=1}^{K} (\delta_k - \bar{\delta})^2}{(K-1)}}$$

McNemar's test

Compute the following contingency matrix for classifiers L1 and L2

L1 \ L2	correct	incorrect
correct	n_{00}	n_{01}
incorrect	n_{10}	n_{11}

n_{00}: the number of instances correctly classified by both classifiers

n_{01}: the number of instances correctly classified by L1 but not by L2

n_{10}: the number of instances correctly classified by L2 but not by L1

n_{11}: the number of instances misclassified by both classifiers

McNemar's test

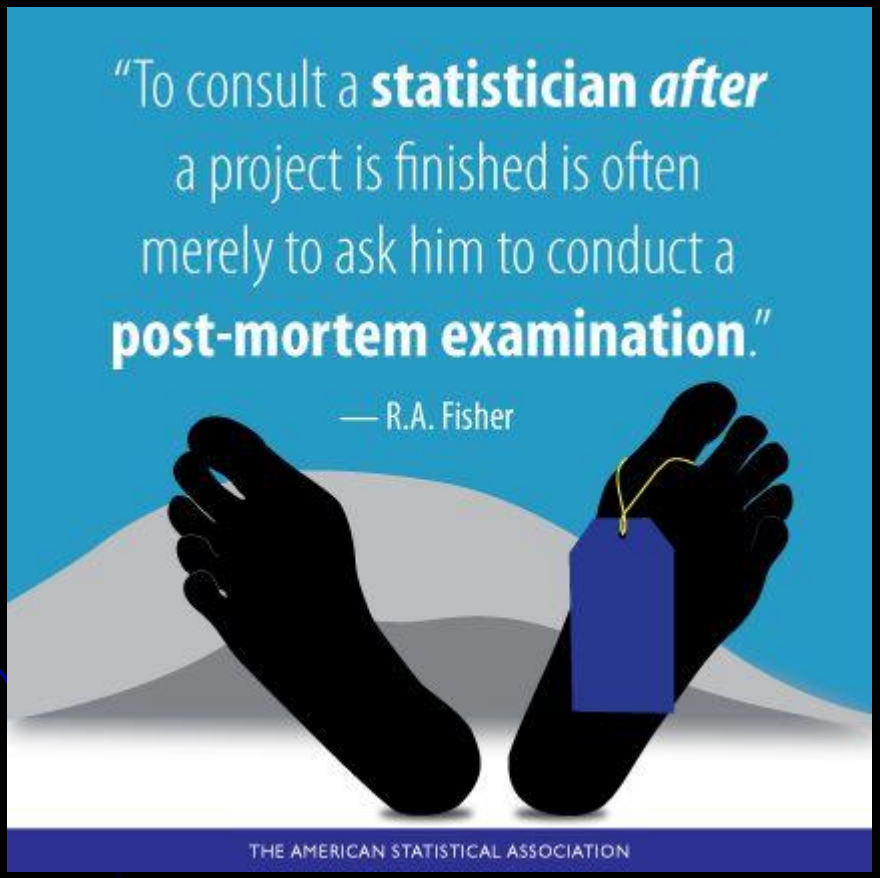
$$M = \frac{(n_{01} - n_{10})^2}{n_{01} + n_{10}}$$

χ^2 with 1 DOF

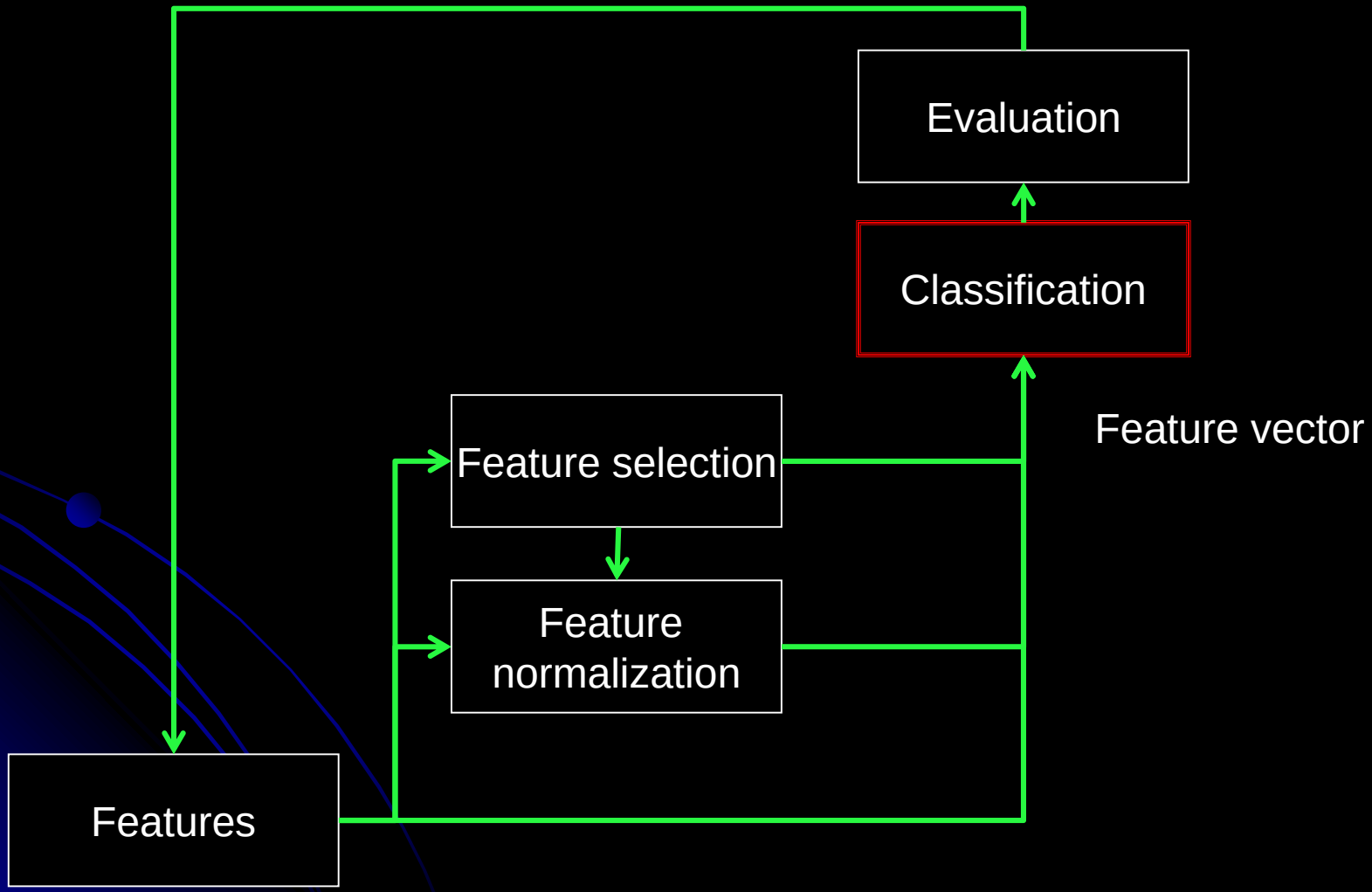
If M>3.84, then with 95% confidence we can reject the H_0 hypothesis that the classifiers have the same error rate

Takehome message

Consult a statistician

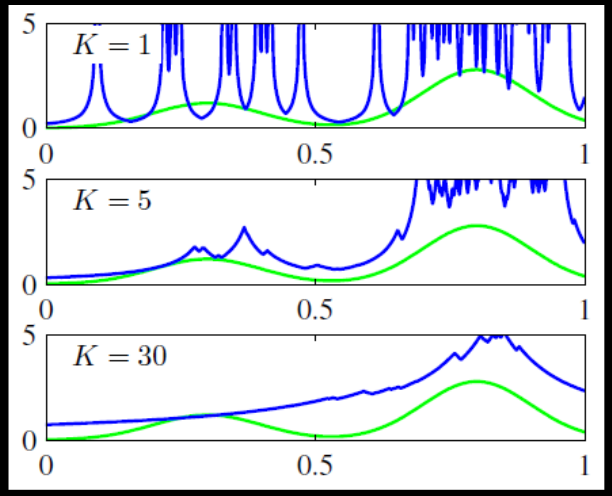


Classification pipeline



Nonparametric PDF estimation

$$\hat{p}(\mathbf{x}) = \frac{K}{NV}$$



Let's fix K

Not a real PDF

In theory, for infinite number of samples, yes

K nearest neighbors

This approach can be used for classification:

draw a sphere centered on **x** containing precisely *K* points irrespective of their class

$$(\mathbf{x}) = \frac{K}{NV}, \;\; p(\mathbf{x}|\omega_i) = \frac{K_i}{N_i V}, \;\; p(\omega_i) = \frac{N_i}{N}$$

$$p(\omega_i|\mathbf{x}) = \frac{\frac{K_i}{N_i V} \cdot \frac{N_i}{N}}{\frac{K}{NV}} = \frac{K_i}{K}$$

K nearest neighbors

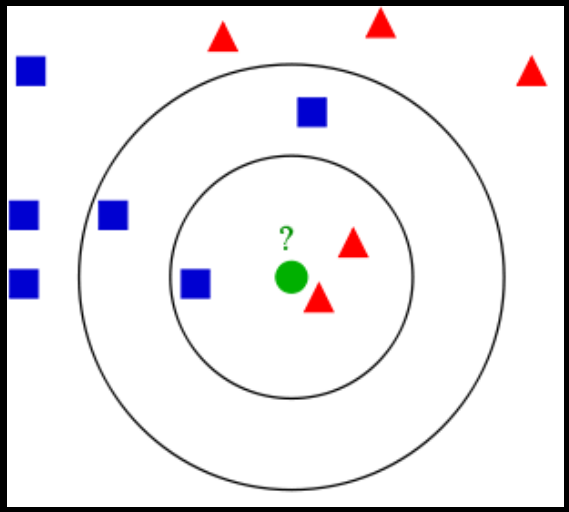
Learning (training)

store the training data

Classification

for unknown **x**, find K nearest neighbors $(\mathbf{x}_1, \cdots \mathbf{x}_K)$, classify according to majority:

$f(\mathbf{x}) = \arg\max_c \sum_{k=1}^{K} \delta(c, y_k)$



K nearest neighbors

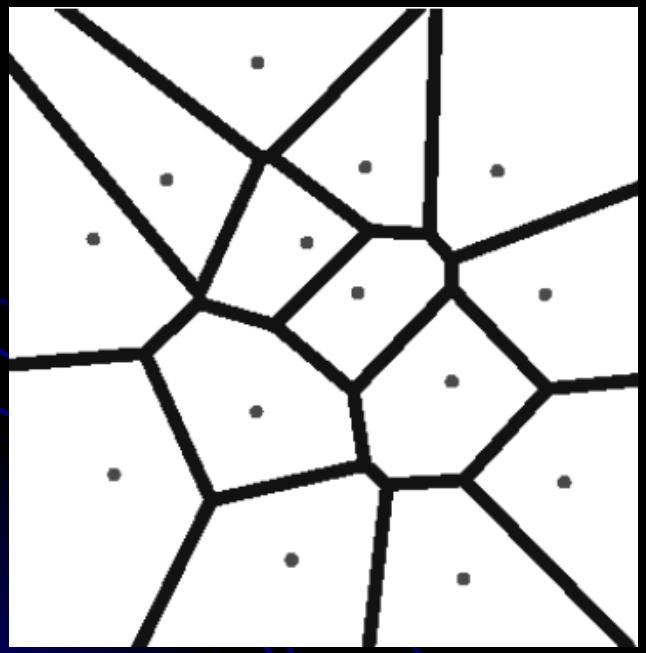
Requires

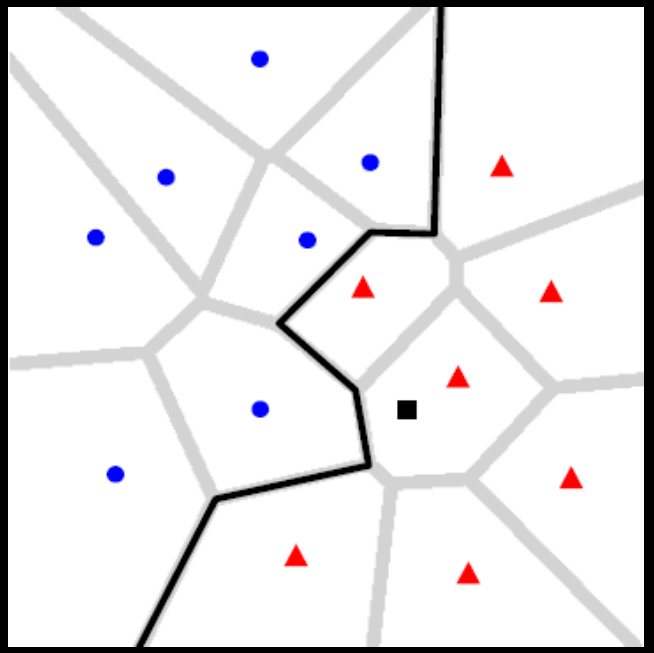
1. The set of stored records
2. Distance metric to compute distance between records
3. The value of K, the number of nearest neighbors to retrieve

1 nearest neighbor

K=1

$$f(\mathbf{x}) = y_i, \qquad \text{where } i = \arg\min_j \|\mathbf{x}_j - \mathbf{x}\|$$

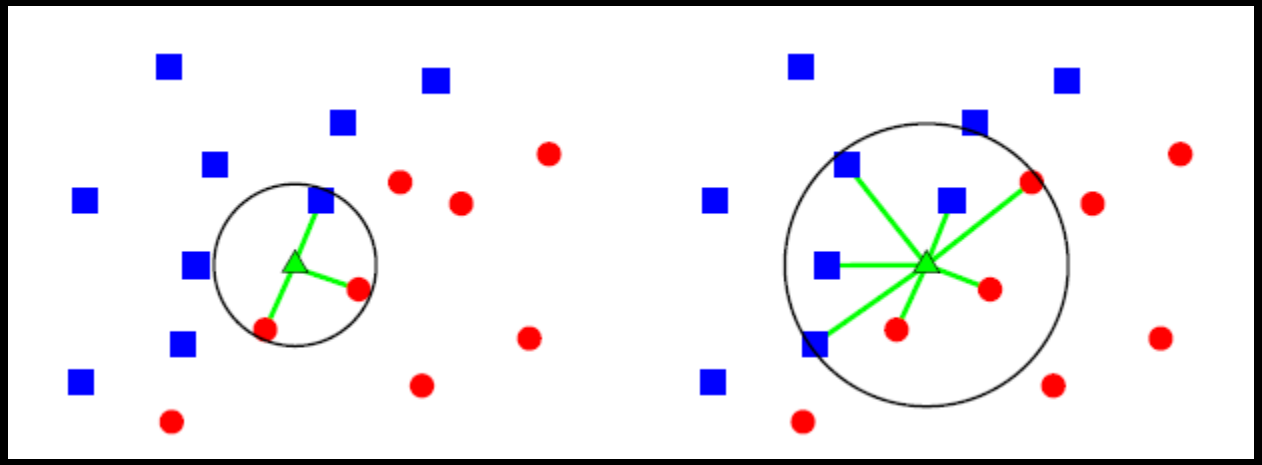




Voronoi diagram

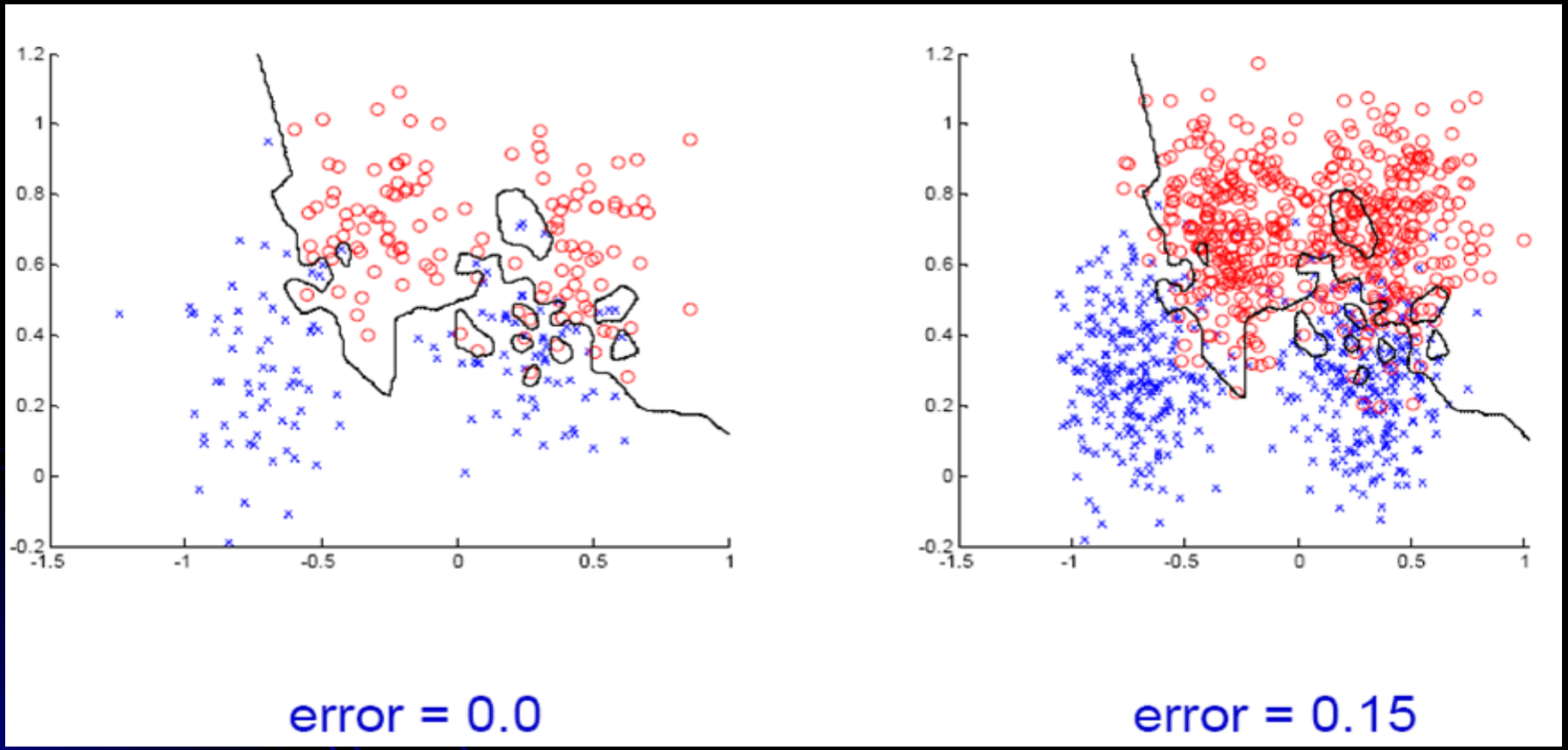
K nearest neighbors

Value of K?

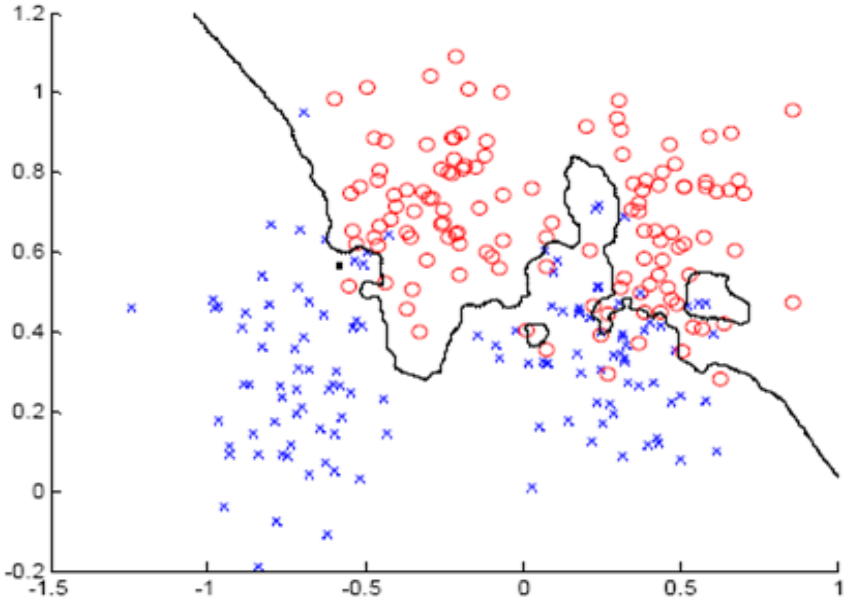


K=3 K=7

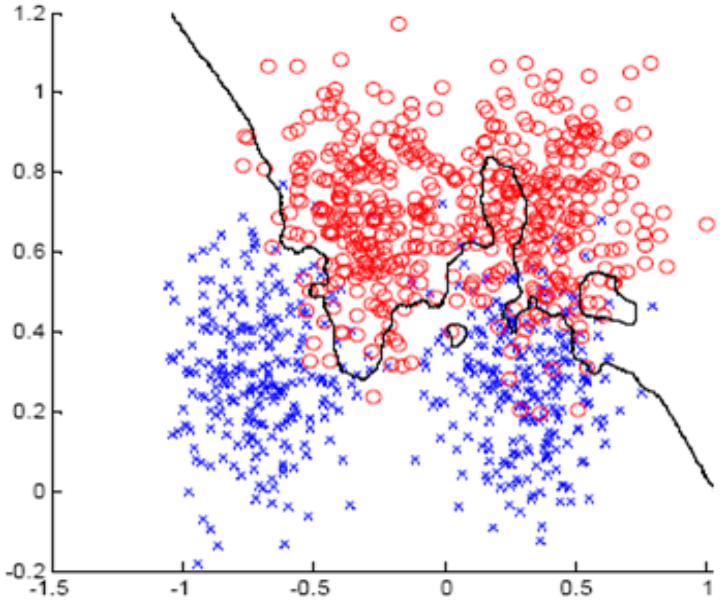
K=1



K=3

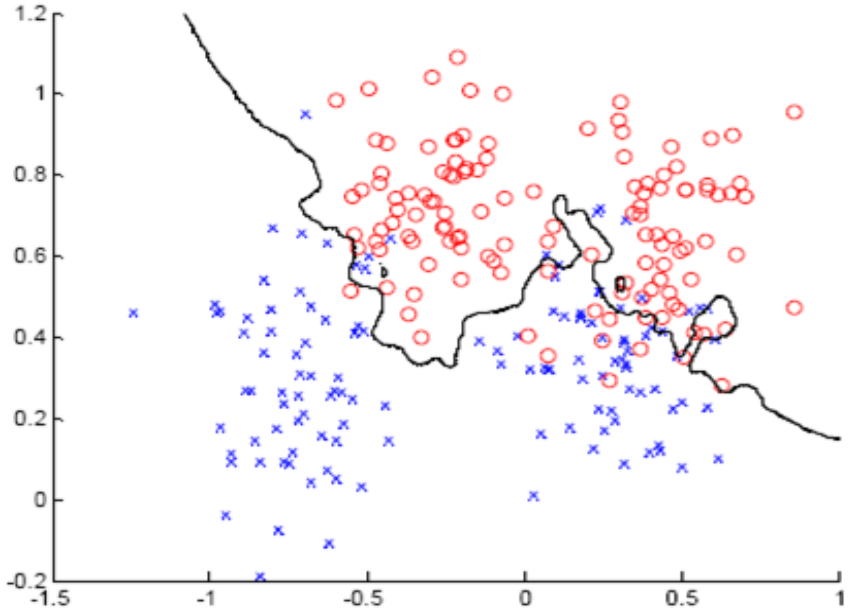


error = 0.0760

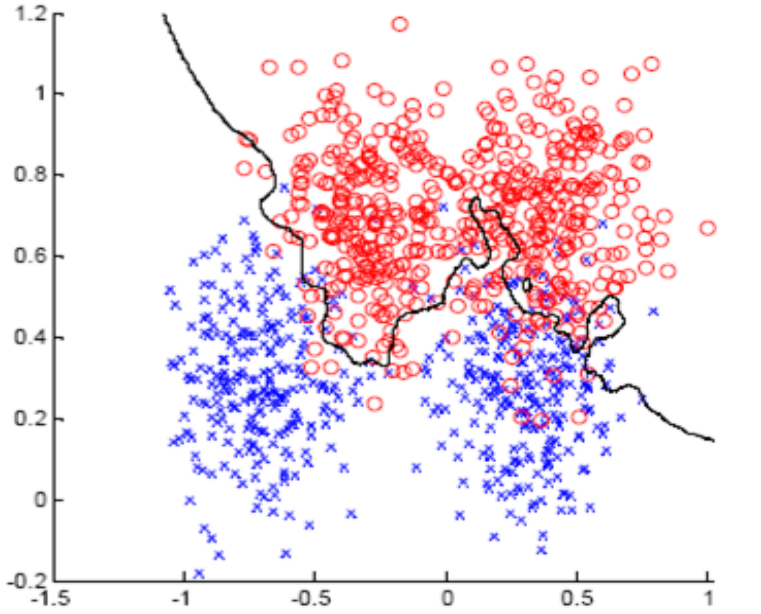


error = 0.1340

K=7

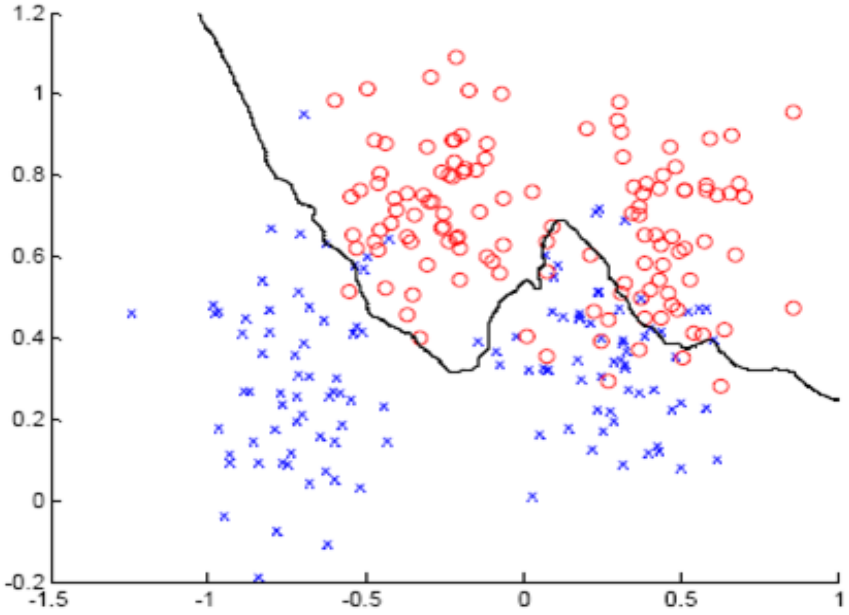


error = 0.1320

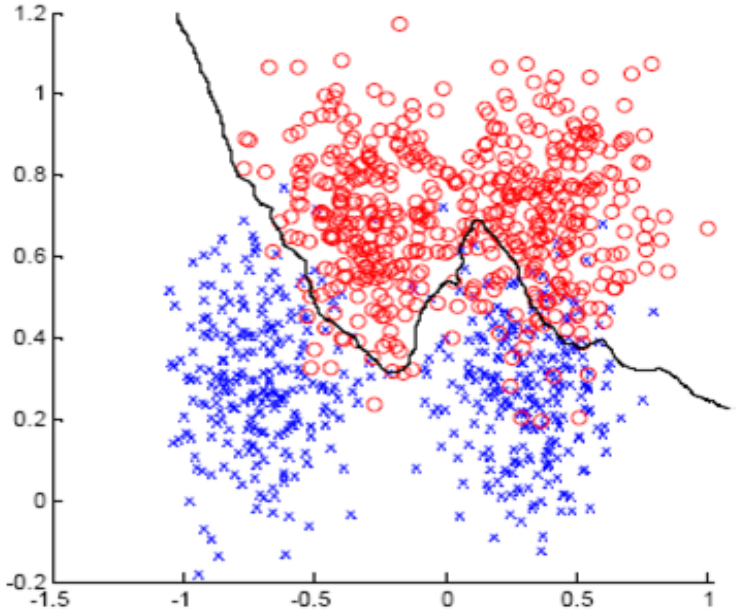


error = 0.1110

K=21



error = 0.1120



error = 0.0920

Optimal K

Increase K:

- less sensitive to noise
- smoother boundary

Decrease K:

- captures finer structure

Pick K not too large, but not too small (depends on data)

Use validation set

Distance metric

Minkowsky:

$$D(\boldsymbol{x},\boldsymbol{y}) = \left(\sum_{i=1}^{m} |x_i - y_i|^r \right)^{1/r}$$

Euclidean:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Manhattan / city-block:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} |x_i - y_i|$$

Camberra:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i + y_i|}$$

Chebychev:

$$D(\boldsymbol{x},\boldsymbol{y}) = \max_{i=1}^{m} |x_i - y_i|$$

Quadratic:

$$D(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T Q(\boldsymbol{x} - \boldsymbol{y}) = \sum_{j=1}^{m} \left(\sum_{i=1}^{m} (x_i - y_i) q_{ji} \right) (x_j - y_j)$$

Q is a problem-specific positive definite $m \times m$ weight matrix

Mahalanobis:

$$D(\boldsymbol{x},\boldsymbol{y}) = [\det V]^{1/m} (\boldsymbol{x} - \boldsymbol{y})^{\mathrm{T}} V^{-1} (\boldsymbol{x} - \boldsymbol{y})$$

V is the covariance matrix of $A_1..A_m$, and A_j is the vector of values for attribute j occuring in the training set instances $1..n$.

Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{\sum_{i=1}^{m} (x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_{i=1}^{m} (x_i - \overline{x_i})^2 \sum_{i=1}^{m} (y_i - \overline{y_i})^2}}$$

$\overline{x_i} = \overline{y_i}$ and is the average value for attribute i occuring in the training set.

Chi-square:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{1}{sum_i} \left(\frac{x_i}{size_{\boldsymbol{x}}} - \frac{y_i}{size_{\boldsymbol{y}}} \right)^2$$

sum_i is the sum of all values for attribute i occuring in the training set, and $size_{\boldsymbol{x}}$ is the sum of all values in the vector $\boldsymbol{x}$.

Kendall's Rank Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = 1 - \frac{2}{n(n-1)} \sum_{i=1}^{m} \sum_{j=1}^{i-1} \mathrm{sign}(x_i - x_j)\,\mathrm{sign}(y_i - y_j)$$

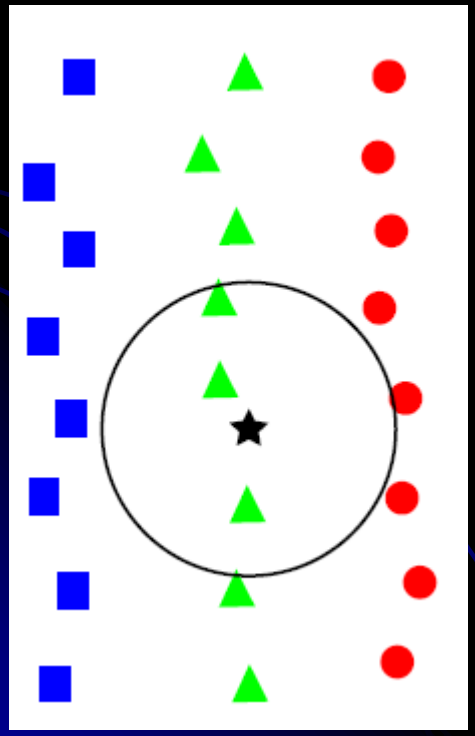
sign(x)=-1, 0 or 1 if $x < 0$, $x = 0$, or $x > 0$, respectively.

Figure 1. Equations of selected distance functions.
($\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of m attribute values).

Distance metric

Euclidean $\mathrm{d}(\mathbf{x}_i, \mathbf{x}) = \sqrt{\sum_{d=1}^{D} (x_i^{(d)} - x^{(d)})^2}$

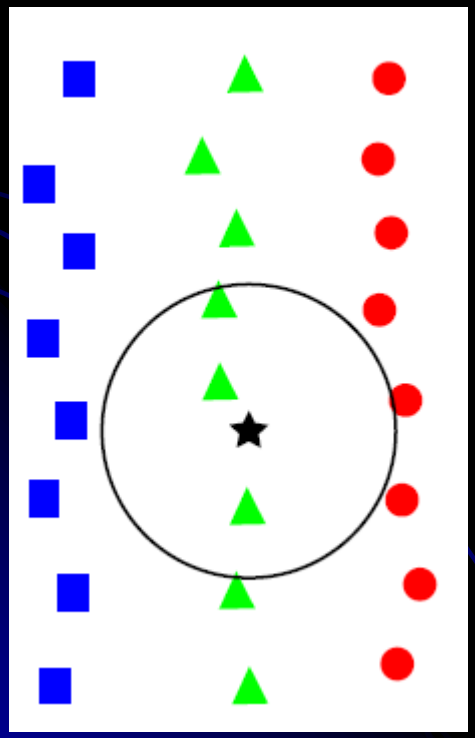
but

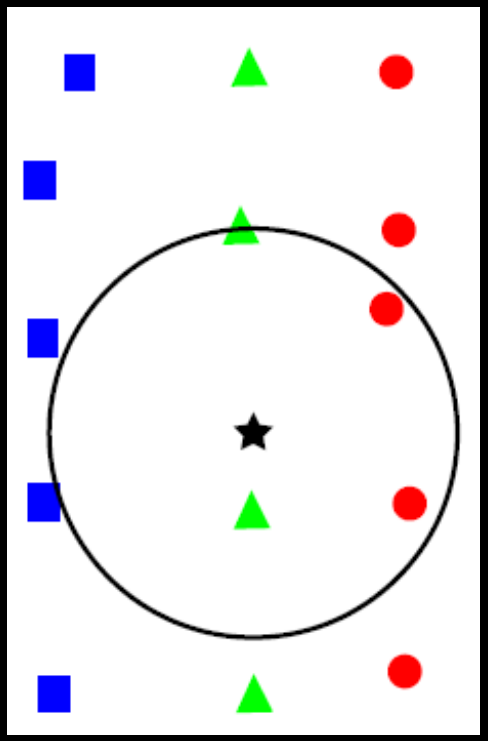


Distance metric

Euclidean $\mathrm{d}(\mathbf{x}_i, \mathbf{x}) = \sqrt{\sum_{d=1}^{D}(x_i^{(d)} - x^{(d)})^2}$

but

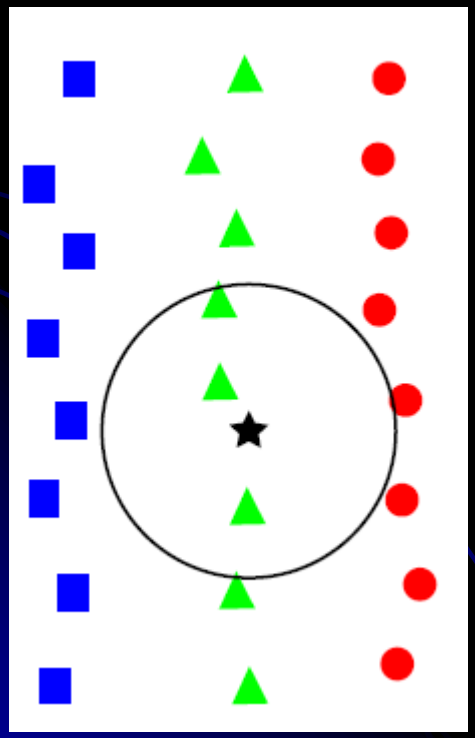


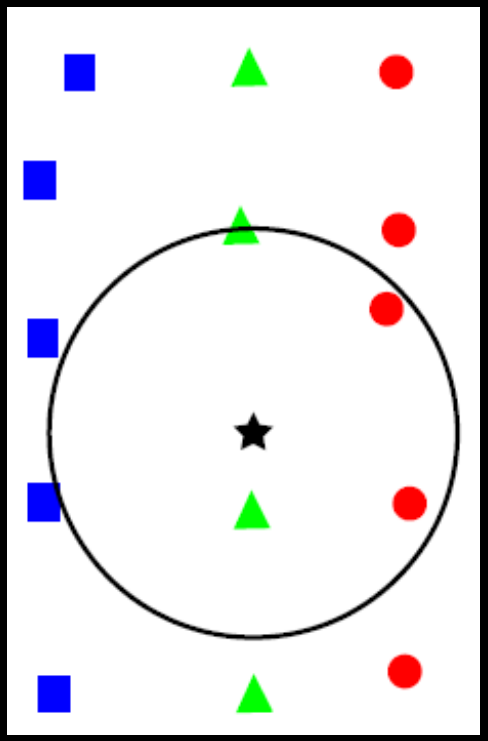


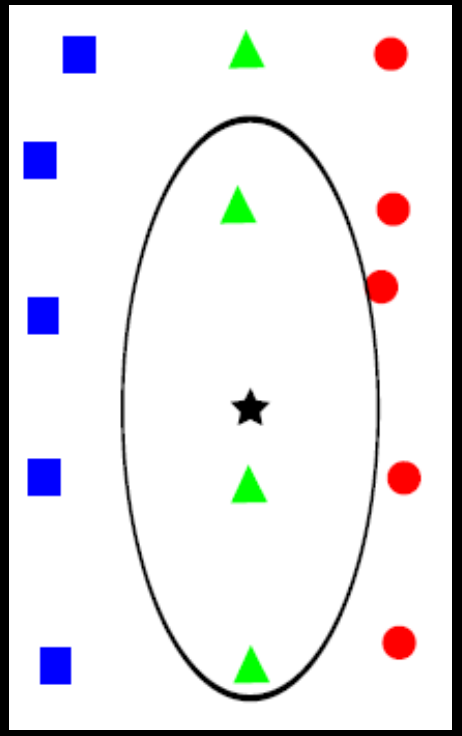
Distance metric

Euclidean $\mathrm{d}(\mathbf{x}_i, \mathbf{x}) = \sqrt{\sum_{d=1}^{D}(x_i^{(d)} - x^{(d)})^2}$

but







Scale Effects

- Different features may have different measurement scales
  - E.g., patient weight in kg (range [50,200]) vs. blood protein values in ng/dL (range [-3,3])
- Consequences
  - Patient weight will have a much greater influence on the distance between samples
  - May bias the performance of the classifier
- Data standardization

Weighted KNN

Feature weighted

$$\mathrm{d}(\mathbf{x}_i, \mathbf{x}) = \sqrt{\sum_{d=1}^{D} w^{(d)} (x_i^{(d)} - x^{(d)})^2}$$

Weighted by the relevance of the feature (e.g. correlation, mutual information, chi square…)

Weighted KNN

Instance weighted

$$f(\mathbf{x}) = \arg\max_{c} \sum_{k=1}^{K} w_k \delta(c, y_k)$$

- Weighted by the distance from the query (different weights proposed in literature)

Geler, Kurbalija, Radovanović, Ivanović: Comparison of different weighting schemes for the *k*NN classifier on time-series data, https://doi.org/10.1007/s10115-015-0881-0

Mahalanobis distance

$$\mathrm{d}(\mathbf{x}_i, \mathbf{x}) = \sqrt{(\mathbf{x}_i - \mathbf{x})^T \mathbf{A} (\mathbf{x}_i - \mathbf{x})}$$

$\mathbf{A} = \mathbf{I}$ Euclidean

$\mathbf{A} = \mathrm{diag}(\mathbf{s})$ feature weighted

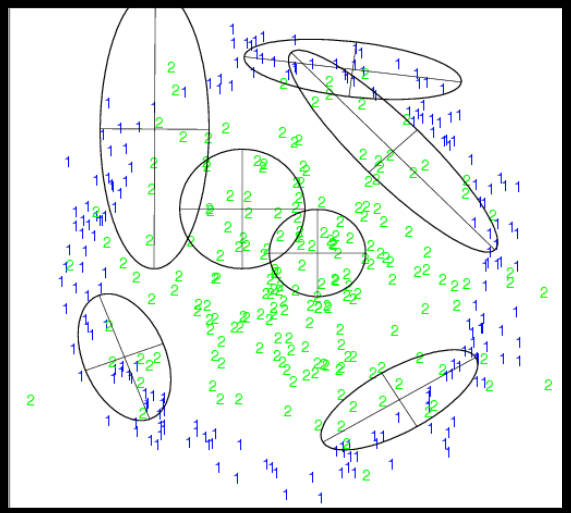
$\mathbf{A} = \boldsymbol{\Sigma}^{-1}$ data driven (covariance matrix)

Discriminant Adaptive NN

Local adaptive Mahalanobis distance

Compute inter- and intraclass scatter in local neighborhood

$$\boldsymbol{\Sigma} = \mathbf{W}^{-\frac{1}{2}}[\mathbf{W}^{-\frac{1}{2}}\mathbf{B}\mathbf{W}^{-\frac{1}{2}} + \epsilon\mathbf{I}]\,\mathbf{W}^{-\frac{1}{2}}$$



$\mathbf{W}$ – within class (intra)

$\mathbf{B}$ – between class (inter)

T. Hastie and R. Tibshirani, "Discriminant adaptive nearest neighbor classification," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 6, pp. 607-616, June 1996.

Categorical data

Most distance measures were designed for linear/real-valued attributes

- Convert categories to numbers (if possible)
- Use 1-hot encoding (Hamming Distance between binary vectors)
- Other similarity metrics

Boriah, Shyam & Chandola, Varun & Kumar, Vipin. (2008). Similarity Measures for Categorical Data: A Comparative Evaluation. SIAM International Conference on Data Mining, SDM 2008

Learning

Learning in this algorithm consists of storing the presented training data

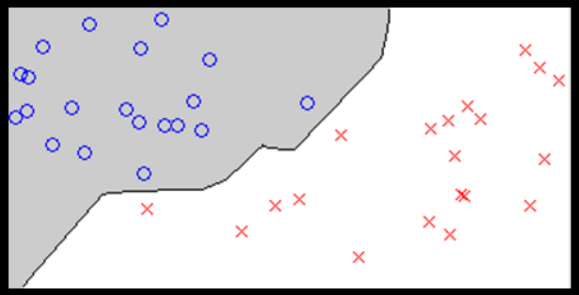
Efficient Data Structures for Retrieval

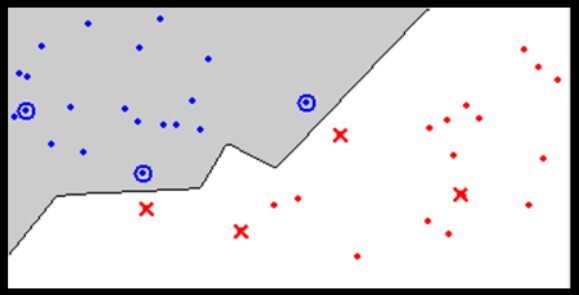
(kd-trees, ball trees)

Selectively Storing Data Points (editing, condensing)

Condensing, editing

Condensing: Retain only the samples that are needed to define the decision boundary





Editing: Remove points that do not agree with the majority of their k nearest neighbours