Supervised methods: Introduction and evaluation

Mario Martin

UPC - Computer Science Dept.

Outline

Outline

Outline

Supervised Learning definition

Definition

Given: A dataset D of observations, usually represented as a table where each row is an observation and each column a descriptor.

Goal: Learn to predict one column of the dataset, named *label*, for new data.

Dataset example

Table (5 fields, 151 records) #2

File Edit Generate

Table Annotations

	Sepal_len_cm	Sepal_wid_cm	Petal_len_cm	Petal_wid_cm	Class
1	5.100	3.500	1.400	0.200	Iris-setosa
2	4.900	3.000	1.400	0.200	Iris-setosa
3	4.700	3.200	1.300	0.200	Iris-setosa
4	4.600	3.100	1.500	0.200	Iris-setosa
5	5.000	3.600	1.400	0.200	Iris-setosa
6	5.400	3.900	1.700	0.400	Iris-setosa
7	4.600	3.400	1.400	0.300	Iris-setosa
8	5.000	3.400	1.500	0.200	Iris-setosa
9	4.400	2.900	1.400	0.200	Iris-setosa
10	4.900	3.100	1.500	0.100	Iris-setosa
11	5.400	3.700	1.500	0.200	Iris-setosa
12	4.800	3.400	1.600	0.200	Iris-setosa
13	4.800	3.000	1.400	0.100	Iris-setosa
14	4.300	3.000	1.100	0.100	Iris-setosa
15	5.800	4.000	1.200	0.200	Iris-setosa

Supervised Learning examples

Some examples

- From a dataset of bank costumers, *predict* whether a new costumer will return a loan or not.
- From a dataset of images of digits, learn to *identify* the digit represented by a new image.
- From a dataset of emails, learn to *distinguish* between SPAM and not SPAM.
- From a dataset of Magnetoencephalography recordings of patients, *diagnose* whether a new individual is schizophrenic or not.

Supervised Learning as Classification

- All the previous tasks were examples of *learning* (why?)

Supervised Learning as Classification

- All the previous tasks were examples of *learning* (why?) to *classify* (a lot of tasks can be expressed as classification tasks)

Supervised Learning as Classification

- All the previous tasks were examples of *learning* (why?) to *classify* (a lot of tasks can be expressed as classification tasks)
- Notice that labels are discrete values (in other case we are talking about *regression*)

Supervised Learning as Classification

- All the previous tasks were examples of *learning* (why?) to *classify* (a lot of tasks can be expressed as classification tasks)
- Notice that labels are discrete values (in other case we are talking about *regression*)
- In order to classify an object in a right class, we need to build a *classifier*

Supervised Learning constraints

Usual considerations:

- Dataset contains **positive** and **negative** examples for each labels.
- Examples are randomly sampled (i.i.d.).
- ... Other assumptions later on.

Outline

Generalization

- How can we infer the class of one observation from other cases?

Generalization

- How can we infer the class of one observation from other cases?
- Generalization is the ability to extend knowledge of a dataset to new data

Generalization

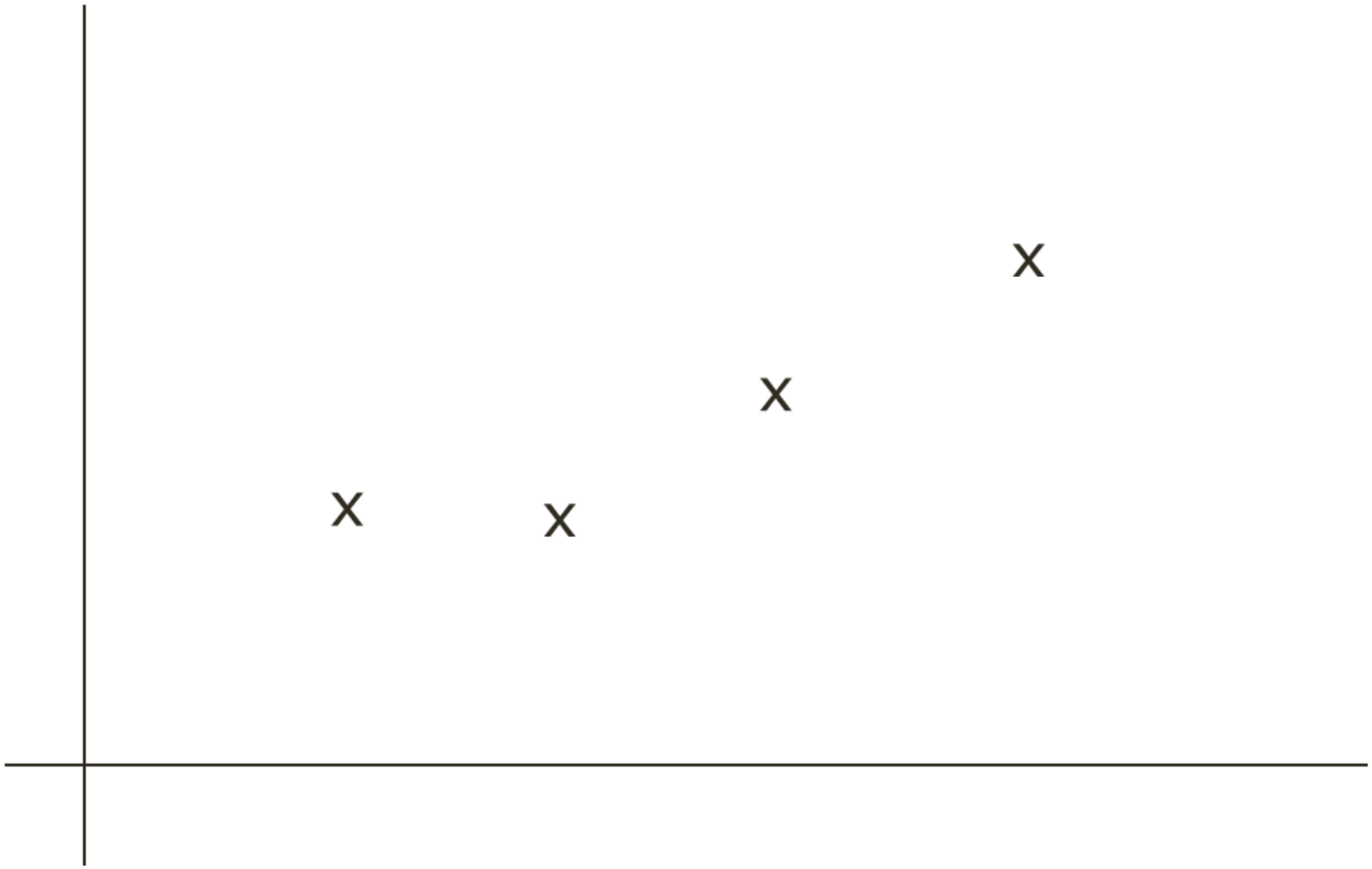
- How can we infer the class of one observation from other cases?
- Generalization is the ability to extend knowledge of a dataset to new data
- But which is the source of the magic of generalization?

Generalization

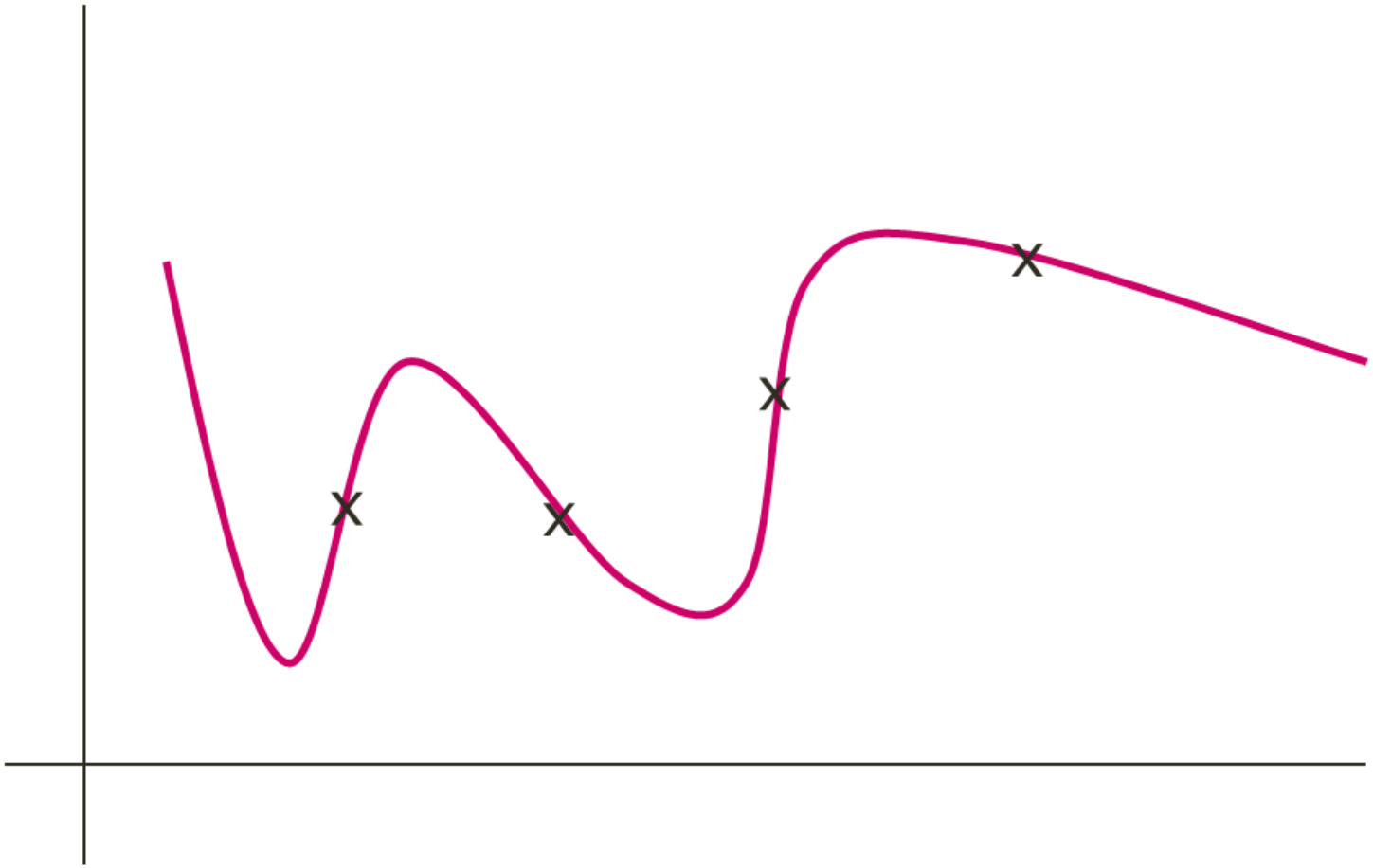
- How can we infer the class of one observation from other cases?
- Generalization is the ability to extend knowledge of a dataset to new data
- But which is the source of the magic of generalization?

... Simplicity

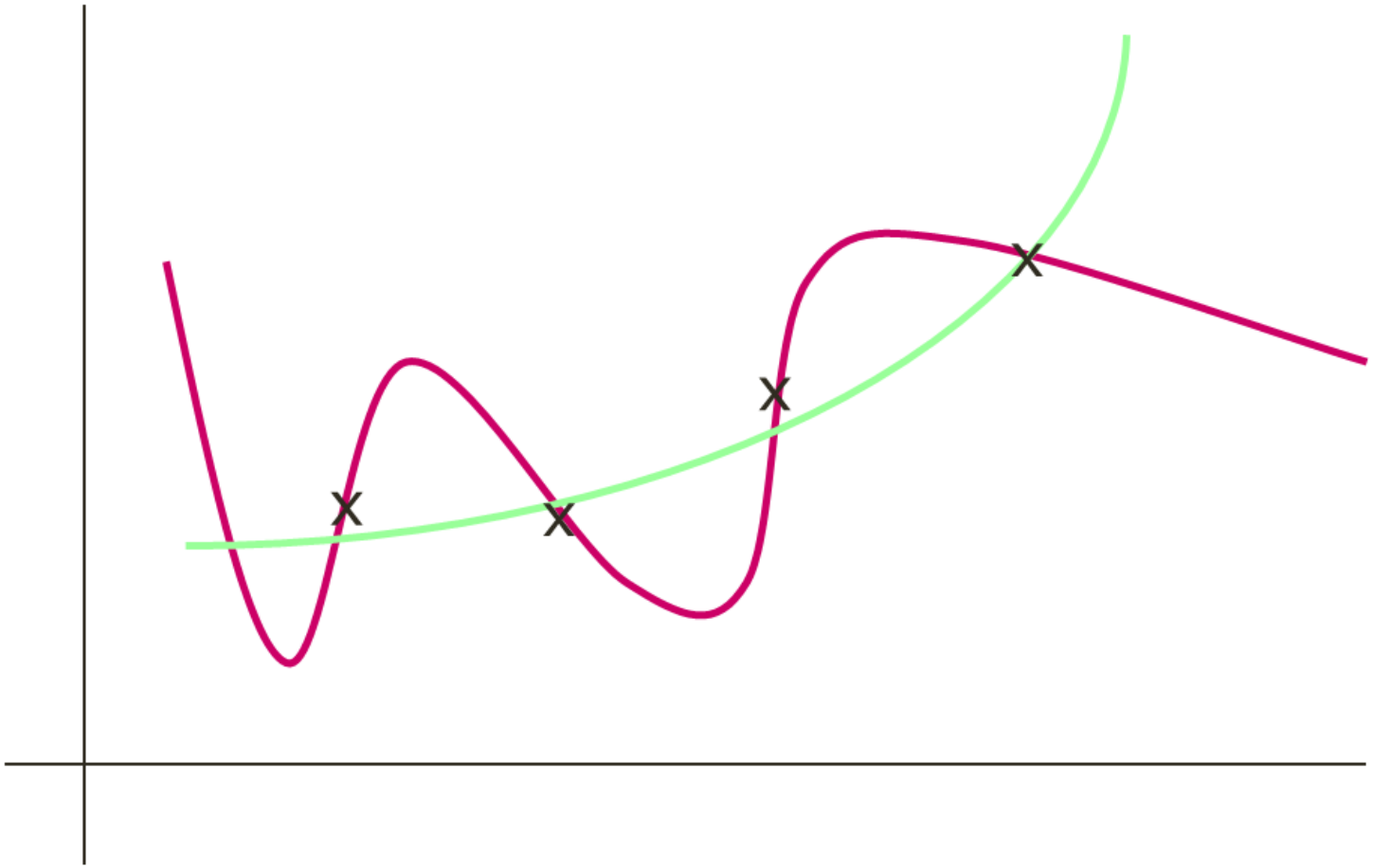
Generalization



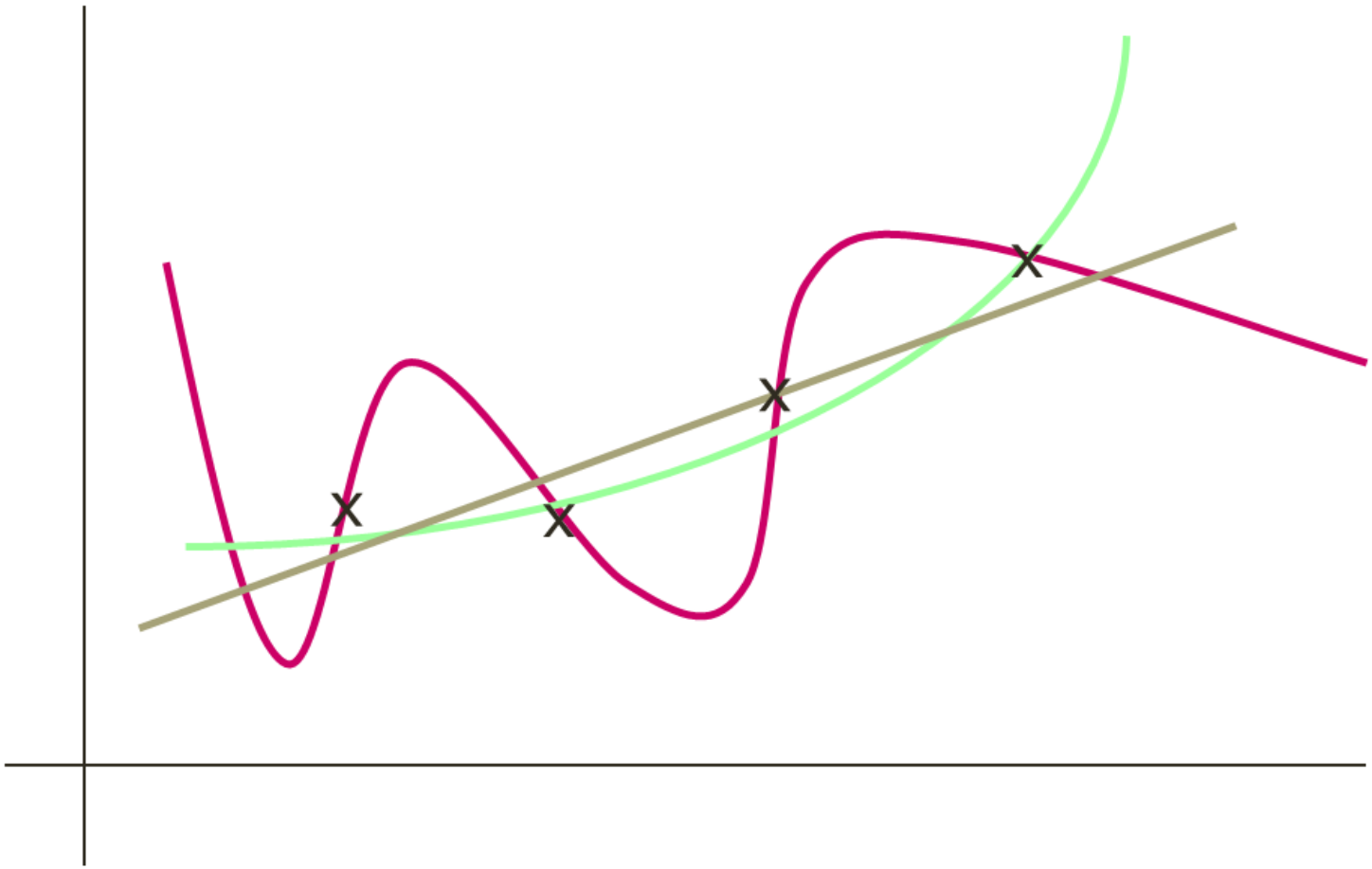
Generalization



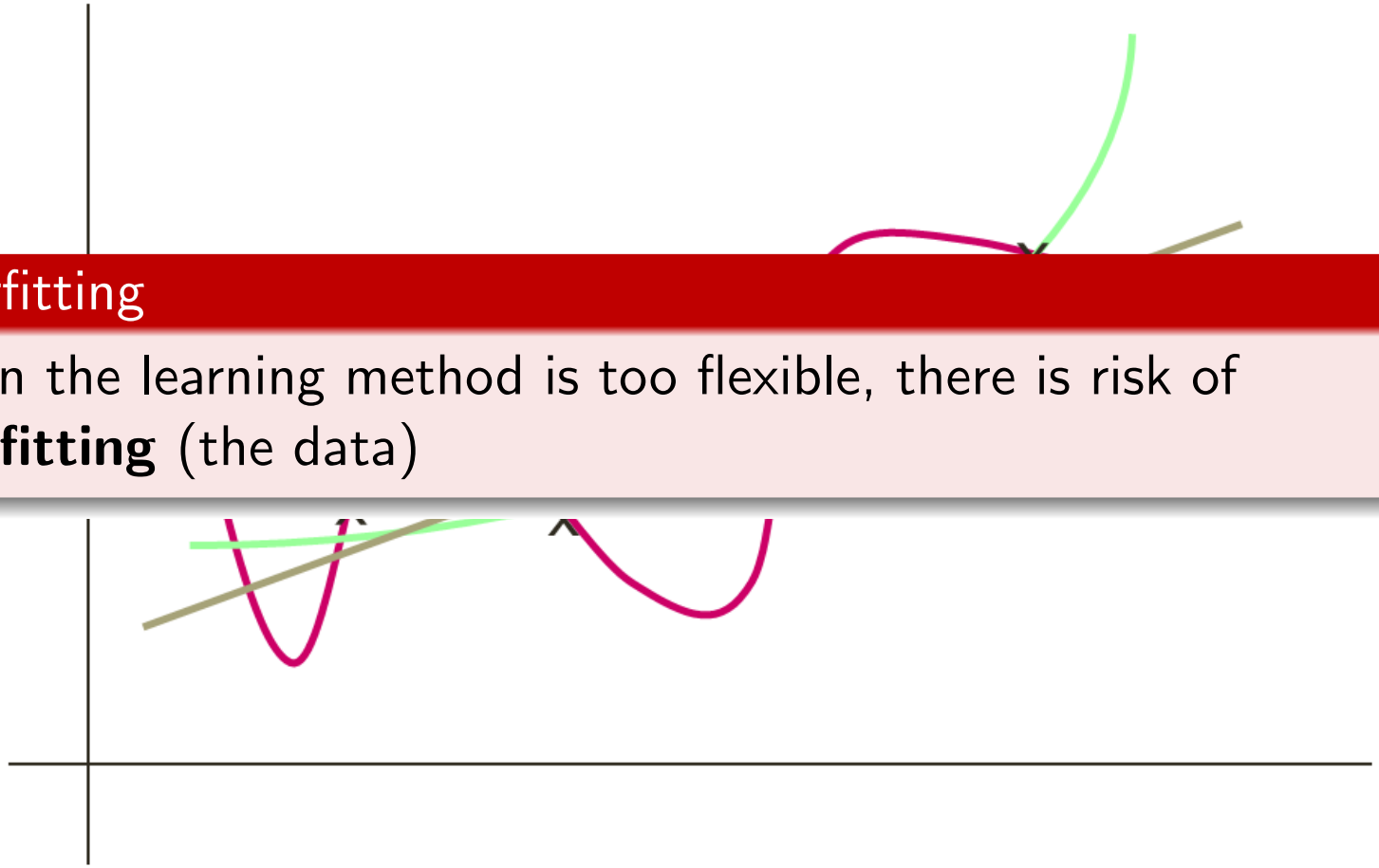
Generalization



Generalization



Generalization



Overfitting

When the learning method is too flexible, there is risk of **overfitting** (the data)

Generalization

Any learning method will show a tendency to *simple explanations* for data:

- Shortest description:
  - Minimum description length
  - Shortest tree size
  - Fewest parameters model
  - Independence of columns
- Regularization about complexity of the classifier:
  - Values of parameters (f.i. limited weight in neural networks)
  - Trade-off on capacity of classifier and error

Outline

Outline

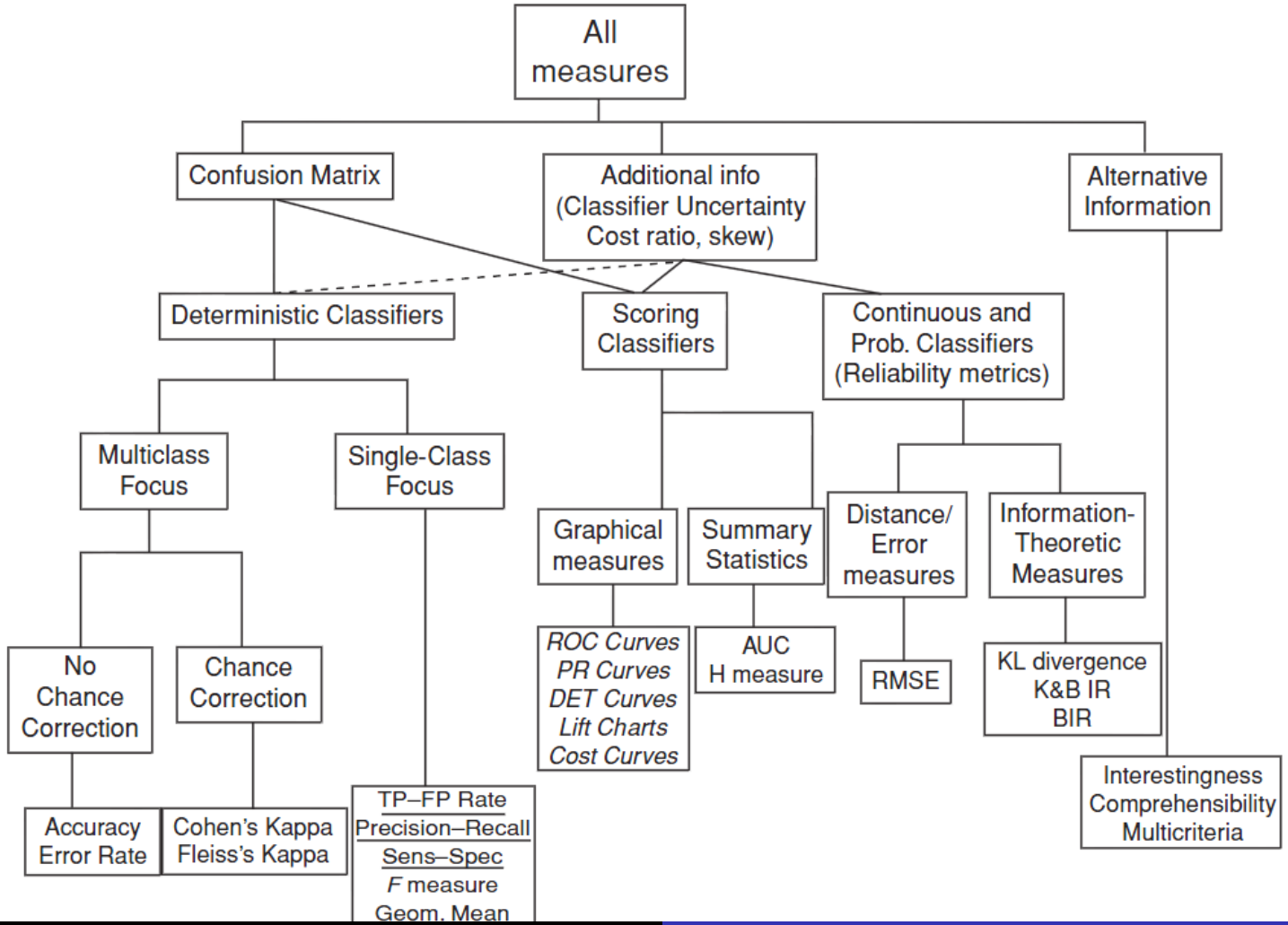
Introduction

- Data mining in supervised mode has a target column.
- Differently than in the case of clustering, now we have an objective way to measure the success of classifiers: Test new cases with the classifier and check predicted labels with reality.

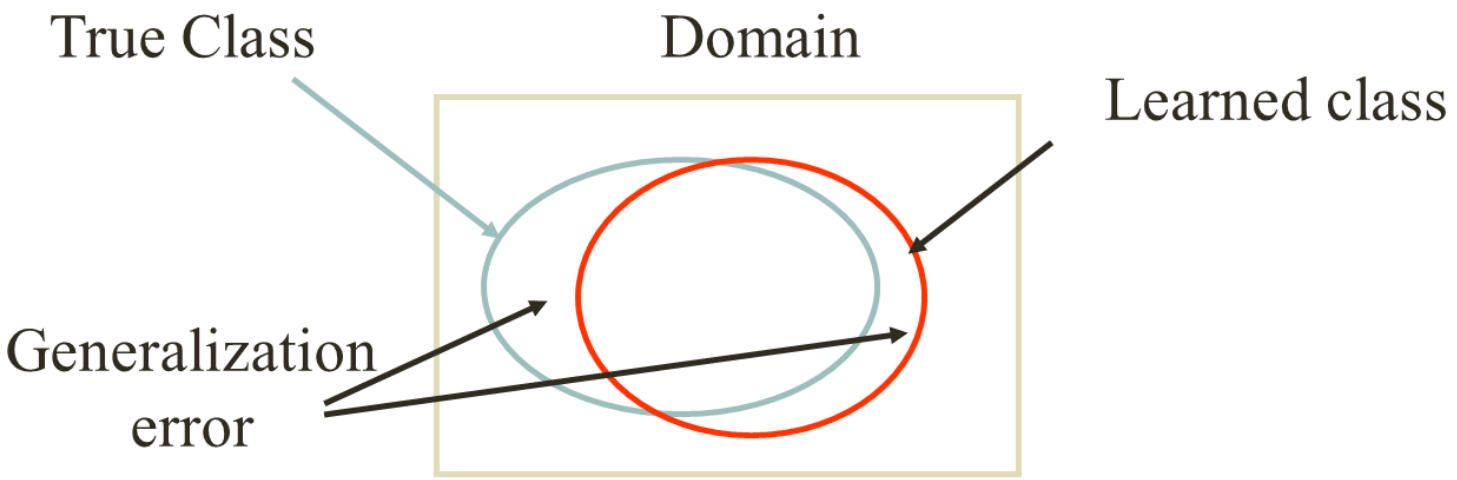
Introduction

- Data mining in supervised mode has a target column.
- Differently than in the case of clustering, now we have an objective way to measure the success of classifiers: Test new cases with the classifier and check predicted labels with reality.
- But not so easy. There are a lot of different measures to qualify success of a classifier.

Measures


All measures
Confusion Matrix
Additional info (Classifier Uncertainty Cost ratio, skew)
Alternative Information
Deterministic Classifiers
Scoring Classifiers
Continuous and Prob. Classifiers (Reliability metrics)
Multiclass Focus
Single-Class Focus
Graphical measures
Summary Statistics
Distance/ Error measures
Information-Theoretic Measures
No Chance Correction
Chance Correction
ROC Curves
PR Curves
DET Curves
Lift Charts
Cost Curves
AUC
H measure
RMSE
KL divergence
K&B IR
BIR
Accuracy
Error Rate
Cohen's Kappa
Fleiss's Kappa
TP–FP Rate
Precision–Recall
Sens–Spec
F measure
Geom. Mean
Interestingness
Comprehensibility
Multicriteria

Measures



$$E_{empiric} = \frac{1}{n}\sum_{i}^{n} \left\| \{ i \mid H(x_i) \neq C(x_i) \} \right\|$$

Confussion matrix

Given a classifier, asses the error it produces on a set of instances from the **confusion matrix**:

	Actual Positive	*Actual Negative*
Predicted Positive	True Positive cases (TP)	False Positive cases (FP)
Predicted Negative	False Negative cases(FN)	True Negative cases (TN)

Each entry is the number of instances.

Measures based on confusion matrix

	Actual Positive	*Actual Negative*
Predicted Positive	True Positive cases (TP)	False Positive cases (FP)
Predicted Negative	False Negative cases(FN)	True Negative cases (TN)

Accuracy

Accuracy: Empiric error on the set of cases given

$$Ac = \frac{TP + TN}{TP + TN + FP + FN}$$

Measures based on confusion matrix

Accuracy is not a good measure when:

- Unbalanced datasets
- Different costs for each kind of error

	Act. Pos.	*Act. Neg.*
Pred. Pos.	10	100
Pred. Neg.	40	300

	Act. Pos.	*Act. Neg.*
Pred. Pos.	50	140
Pred. Neg.	0	260

Unbalanced case: Assume we are interested in detecting one kind of documents from a large corpus. Both tables same accuracy but show different behaviors.

Measures based on confusion matrix

Accuracy is not a good measure when:

- Unbalanced datasets
- Different costs for each kind of error

	Act. Pos.	*Act. Neg.*
Pred. Pos.	100	10
Pred. Neg.	100	290

	Act. Pos.	*Act. Neg.*
Pred. Pos.	100	100
Pred. Neg.	10	290

Different costs: Assume cost of misclassifying a positive case is 10 times misclassifying a negative case. Same accuracy but different cost.

Measures based on confusion matrix

Measures taken from document classification (imbalanced case)

- **Recall**: Proportion of positive cases that were detected

$$R = \frac{TP}{TP + FP}$$

- **Precision**: Proportion of positive cases with respect the number of cases labeled positive

$$P = \frac{TP}{TP + FN}$$

- **F-measure**: Harmonic mean of Recall and Precision

$$F = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2PR}{P + R}$$

Outline

Cross-validation

- Matrix confusion is generated form testing the classifier on new data.

Cross-validation

- Matrix confusion is generated form testing the classifier on new data.
- Examples used for learning the classifier cannot be used to generate the matrix confusion (why?)

Cross-validation

- Matrix confusion is generated form testing the classifier on new data.
- Examples used for learning the classifier cannot be used to generate the matrix confusion (why?)
- ... But how do we obtain new data?

Cross-validation

- Matrix confusion is generated form testing the classifier on new data.
- Examples used for learning the classifier cannot be used to generate the matrix confusion (why?)
- ... But how do we obtain new data?

Solution

Do not use all your data for learning.... Keep some examples for testing!

Cross-validation

Cross-validation: Divide your data in two sets - one for learning and one for testing

Cross-validation

Cross-validation: Divide your data in two sets - one for learning and one for testing

- Use more data for training than for testing. Usually 66% data for training - 33% of data testing

Problems with cross-validation

- When data is scarce you "waste" your data for testing.
- More importantly, *results may depend on the split of data into training and testing.*

Problems with cross-validation

- When data is scarce you "waste" your data for testing.
- More importantly, *results may depend on the split of data into training and testing.*
- How can we solve that?

Problems with cross-validation

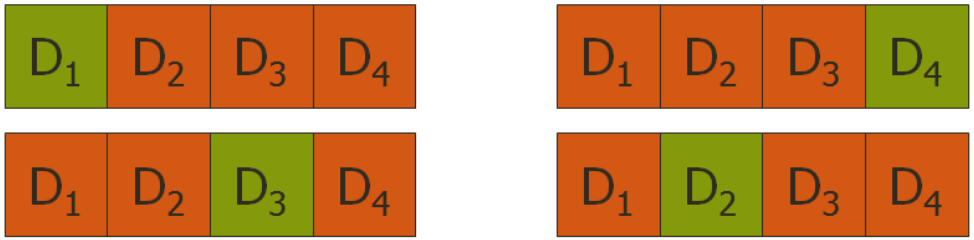
- When data is scarce you "waste" your data for testing.
- More importantly, *results may depend on the split of data into training and testing.*
- How can we solve that?
- ... repeating the split several times and averaging results.
  - k-fold cross-validation
  - leave-one-out
  - Bootstrapping

Outline

k-fold cross-validation

k-fold cross-validation: Divide your dataset into k disjoints subsets of equal length.

Repeat k times the learning process, each time leaving out for testing a different subset.



Average the k accuracy estimators.

k-fold cross-validation

Advantages:

- More robust estimator (less variance)

Problems:

- We still waste 1/k of your data for testing
- We have to repeat the learning process k times

How do we set k? Trade-off time/variance. Usually set to 5 or 10.

Leave-one-out (LOO)

Leave-one-out (LOO): Extreme version of k-fold cross-validation where k is set to n, the number of examples in your dataset.

Repeat n executions of the learning algorithm with $n-1$ examples for learning and 1 for testing (every time a different one).

Average the n estimators.

Obviously it takes time to compute, but takes maximum profit of examples and reduces variance of estimator.

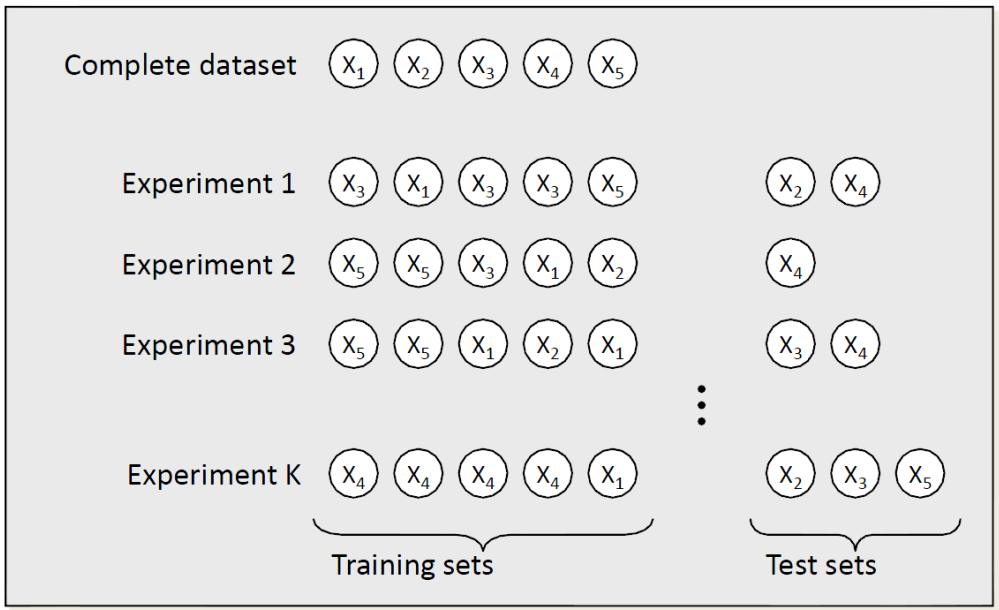
Outline

Bootstrapping

Repeat this process K times (hundreds!):

- Randomly select (*with replacement*) n examples for training
- Examples not selected for training are used for testing (number of examples is likely to change from fold to fold)

True error is estimated as the average error rate on test data



Outline

Confidence interval one classifier

- $E_{\text{empirical}}$ (from now on $\hat{\epsilon}$) follows a binomial distribution (why?)

Confidence interval one classifier

- $E_{\text{empirical}}$ (from now on $\hat{\epsilon}$) follows a binomial distribution (why?) with mean $\hat{\epsilon}$ and $\sigma = \sqrt{\frac{\hat{\epsilon}\,(1-\hat{\epsilon})}{n}}$
- When $n\,\hat{\epsilon}\,(1-\hat{\epsilon}) \geq 5$ binomial can be approximated by a Normal distribution with same mean and variance. So:

$$\hat{\epsilon} - z_{\alpha/2}\,\sigma \leq \epsilon \leq \hat{\epsilon} + z_{\alpha/2}\,\sigma$$

For a 95% confidence interval $z_{0.025} = 1.96$.

Confidence interval one classifier

- $E_{\text{empirical}}$ (from now on $\hat{\epsilon}$) follows a binomial distribution (why?) with mean $\hat{\epsilon}$ and $\sigma = \sqrt{\frac{\hat{\epsilon}\,(1-\hat{\epsilon})}{n}}$
- When $n\,\hat{\epsilon}\,(1-\hat{\epsilon}) \geq 5$ binomial can be approximated by a Normal distribution with same mean and variance. So:

$$\hat{\epsilon} - z_{\alpha/2}\,\sigma \leq \epsilon \leq \hat{\epsilon} + z_{\alpha/2}\,\sigma$$

For a 95% confidence interval $z_{0.025} = 1.96$.

- When $n\,\hat{\epsilon}\,(1-\hat{\epsilon}) < 5$, apply cumulative distribution function of binomial: prob. of doing maximum k errors is $\Pr\{X \leq k\} = \sum_{i=0}^{k} \binom{n}{i} \hat{\epsilon}^i (1-\hat{\epsilon})^{n-i}$.

Confidence interval one classifier

- $E_{\text{empirical}}$ (from now on $\hat{\epsilon}$) follows a binomial distribution (why?) with mean $\hat{\epsilon}$ and $\sigma = \sqrt{\frac{\hat{\epsilon}\,(1-\hat{\epsilon})}{n}}$
- When $n\,\hat{\epsilon}\,(1-\hat{\epsilon}) \geq 5$ binomial can be approximated by a Normal distribution with same mean and variance. So:

$$\hat{\epsilon} - z_{\alpha/2}\,\sigma \leq \epsilon \leq \hat{\epsilon} + z_{\alpha/2}\,\sigma$$

For a 95% confidence interval $z_{0.025} = 1.96$.

- When $n\,\hat{\epsilon}\,(1-\hat{\epsilon}) < 5$, apply cumulative distribution function of binomial: prob. of doing maximum k errors is $\Pr\{X \leq k\} = \sum_{i=0}^{k} \binom{n}{i} \hat{\epsilon}^i (1-\hat{\epsilon})^{n-i}$.

```
binom.test(k, n, e, "less", conf.level)
```

Confidence interval of averaging k classifiers

- Addition of Binomials can be approximated by Normal. So again:

$$\hat{\epsilon} - z_{\alpha/2}\,\sigma \leq \epsilon \leq \hat{\epsilon} + z_{\alpha/2}\,\sigma$$

But now, $\hat{\epsilon}$ is the average of all cross-validation runs and $\sigma = \sqrt{\frac{\hat{\epsilon}\,(1-\hat{\epsilon})}{n}}$ where n is the numbre of examples used in all cross-validation runs

Comparing two classifiers: McNemar's Test

- Goal: decide which of two classifiers h_0 and h_1 has lower error rate
- Run h_0 and h_1 and build the following table:

n_{00}: number of cases both right	n_{01}: number of cases h_0 right and h_1 fails
n_{10}: number of cases h_0 fails and h_1 right	n_{11}: number of cases both fail

Comparing two classifiers: McNemar's Test

- Null hypothesis: $n_{01} = n_{10}$ (both same error)
- McNemar's Test:

$$M = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}} > \chi^2_{1,\alpha}$$

- M is distributed approximately as χ^2 with 1 degree of freedom. For a 95% confidence test, $\chi^2_{1,0.95} = 3.84$. So if M is larger than 3.84, then with 95% confidence, we can reject the null hypothesis
- Once we know they don't have the same error, which one is better?

Outline

Some observations:

1. Multi-class classification
2. Data preparation
3. General approaches to build classifiers

Some observations:

1. Multi-class classification
2. Data preparation
3. General approaches to build classifiers

Multi-class problem

Some observations:

- In some classifiers labels are assumed to be binary: $+1/$-1
- We name *positive examples* to observations with label $+1$ and *negative examples* to observations with labels -1

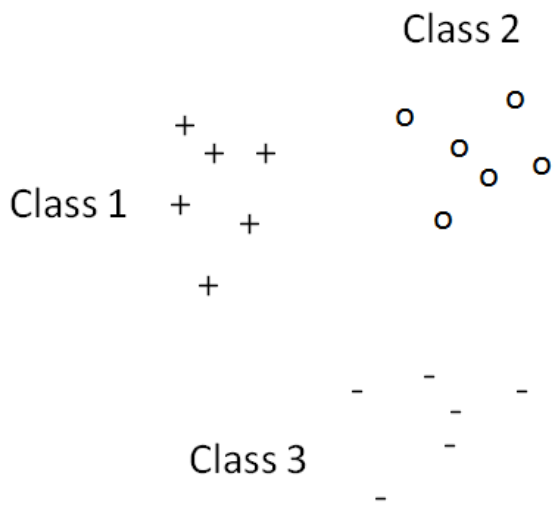
When labels are not binary we always can build a combination of classifiers

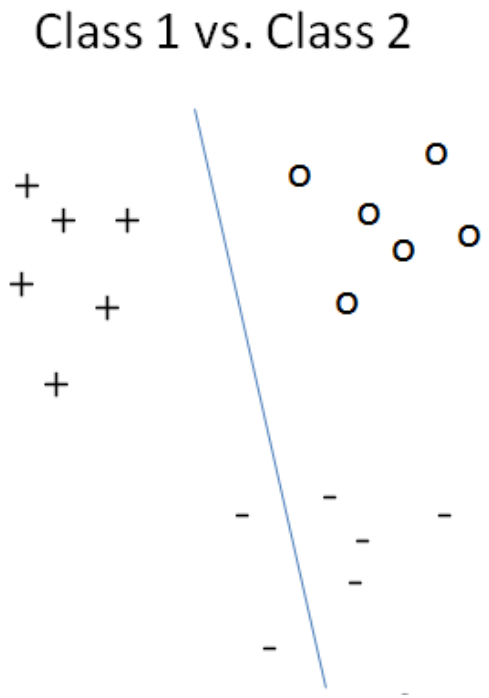
Two approximations:

- One versus One (OVO)
- One versus All (OVA)

One Versus One

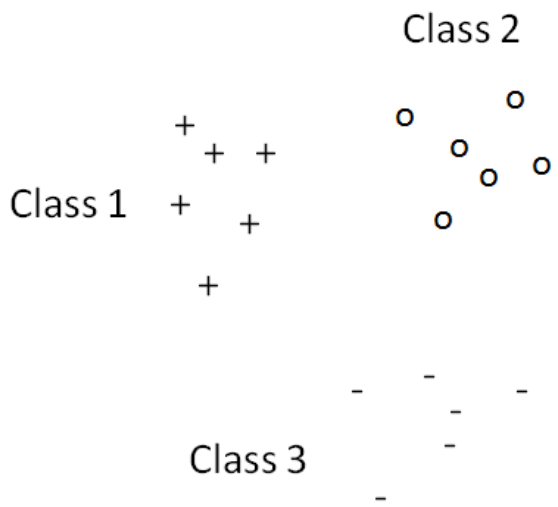
Given k labels, learn to separate class 1 from 2, class 1 from 3, ... class 1 from k, class 2 from 3, class 2 from 4, ... class 2 from k, ... class $k-1$ from class k

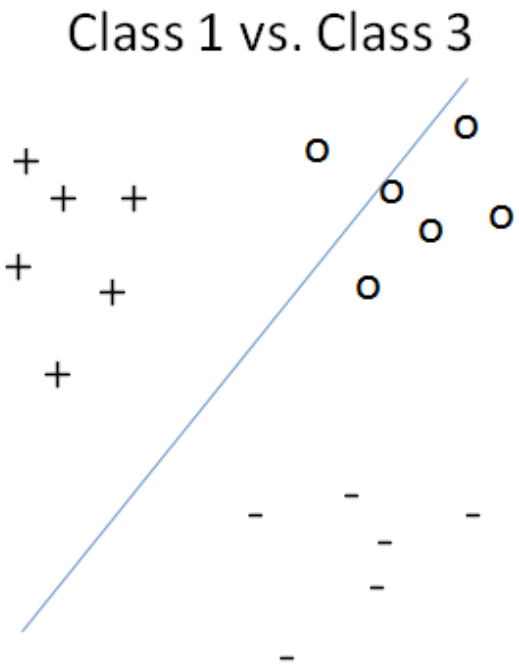




One Versus One

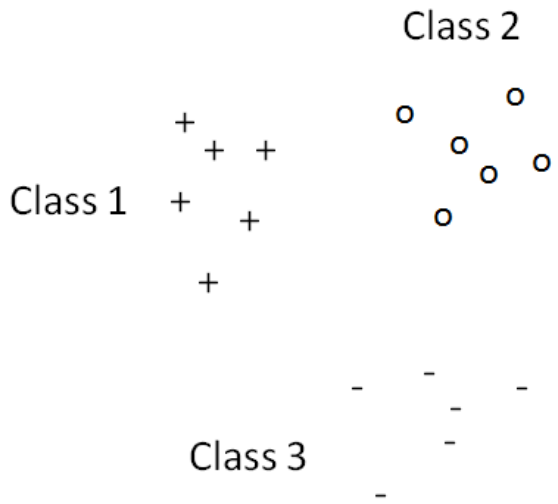
Given k labels, learn to separate class 1 from 2, class 1 from 3, ... class 1 from k, class 2 from 3, class 2 from 4, ... class 2 from k, ... class $k-1$ from class k

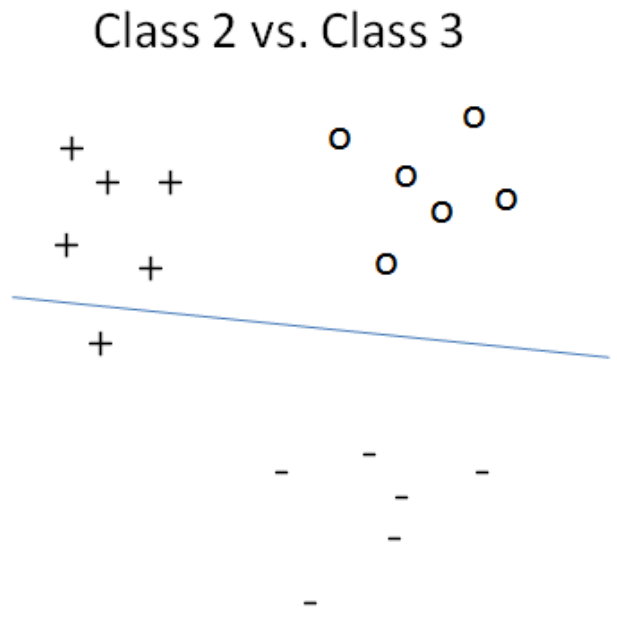




One Versus One

Given k labels, learn to separate class 1 from 2, class 1 from 3, ... class 1 from k, class 2 from 3, class 2 from 4, ... class 2 from k, ... class $k-1$ from class k





One Versus One

After learning $\frac{k(k-1)}{2}$ classifiers, when label for a new observation is required, apply all classifiers and *vote*.

Ties are broken by randomly choosing one label.

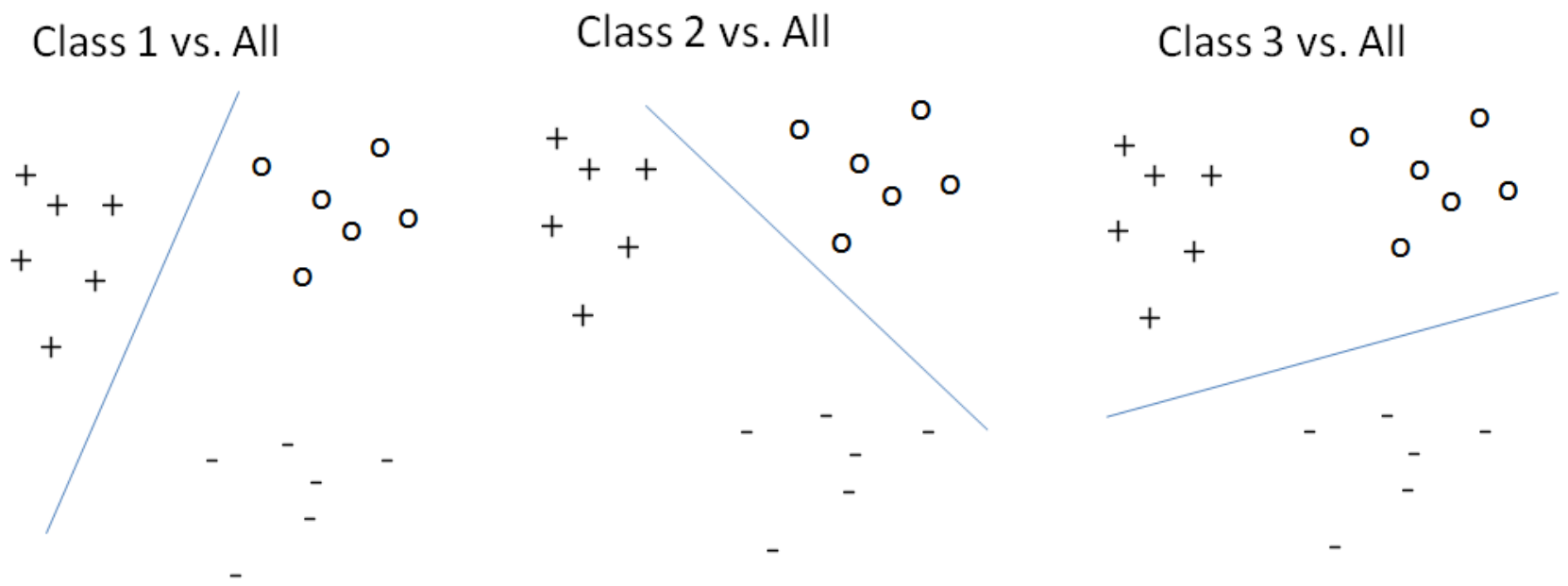
Example:

Classifier	Vote
1 vs. 2	1
1 vs. 3	1
2 vs. 3	3

Final label: 1

One Versus All

Given k labels, learn to separate class 1 from all other classes, class 2 from all other classes .... class $k-1$ from all other classes.



One Versus All

After learning k classifiers, when label for a new observation is required, apply all classifiers and *vote*.

Ties are broken by randomly choosing one label.

Example:

Classifier	Vote
1 vs. All	1
2 vs. All	All
3 vs. All	All

Final label: 1

Some comments:

1. Multi-class classification
2. Data preparation
3. General approaches to build classifiers

Dataset preparation

From now on, we assume we have a relatively clean dataset. That means:

- **No missing values**.
- Relevant features included in dataset (with not an overwhelming amount of irrelevant columns).
- Some algorithms only work with numerical data or with categorical data.
- Usually, we work with normalized or standardized data.

Dataset preparation

From now on, we assume we have a relatively clean dataset. That means:

- **No missing values** ✓.
- Relevant features included in dataset (with not an overwhelming amount of irrelevant columns).
- Some algorithms only work with numerical data or with categorical data.
- Usually, we work with normalized or standardized data.

Dataset preparation

From now on, we assume we have a relatively clean dataset. That means:

- No missing values.
- **Relevant features included in dataset** (with not an overwhelming amount of irrelevant columns).
- Some algorithms only work with numerical data or with categorical data.
- Usually, we work with normalized or standardized data.

Relevant features

- Most algorithms work with a moderate amount of irrelevant columns. However, when the number of irrelevant columns is too much large, the algorithm can be fooled.
- Some methods remove irrelevant features (this is know as feature selection). For instance, sort features by some correlation measure with labels (Correlation, Information Gain, etc.) and keep the top ones.
- We'll see that in the next set of slides

Dataset preparation

From now on, we assume we have a relatively clean dataset. That means:

- No missing values.
- Relevant features included in dataset (with not an overwhelming amount of irrelevant columns).
- **Some algorithms only work with numerical data or with categorical data**.
- Usually, we work with normalized or standardized data.

When numerical data is needed

- Build a new column for each category.

Object	Color
Obj1	blue
Obj2	green
Obj3	green
Obj4	red

When numerical data is needed

- Build a new column for each category.

Object	Color
Obj1	blue
Obj2	green
Obj3	green
Obj4	red

$\Rightarrow$

Object	blue	green	red
Obj1	1	0	0
Obj2	0	1	0
Obj3	0	1	0
Obj4	0	0	1

When categorical data is needed

Discretization of continuous data. Several approaches:

- Rounding (Decimation)
- Finding best cut places
  - Minimize:

$$ER = \sum_{n=1}^{k} \left(\sum_{i \in bin_k} |v_{i,n} - v_{\mathrm{moda},n}| \right)$$

    where k is the number of bins (categories). For instance $\{1, 1, 2, 2, 2, 5, 6, 8, 8, 9\}$ into three bins: **A**$=\{1, 1, 2, 2, 2\}$, **B**$=\{5, 6\}$, **C**$=\{8, 8, 9\}$
  - Target oriented methods: Look for the statistically best splitting of data according to label distribution, f.i., ChiMerge method

Dataset preparation

From now on, we assume we have a relatively clean dataset. That means:

- No missing values.
- Relevant features included in dataset (with not an overwhelming amount of irrelevant columns).
- Some algorithms only work with numerical data or with categorical data.
- **Usually, we work with normalized or standardized data**.

Normalization and standardization

In order to give the same weight to all features we normalize the data to make them independent of units.

- Normalization 0,1:

$$v_i^n(k) = \frac{v_i(k) - min(v_i)}{max(v_i) - min(v_i)}$$

Normalization and standardization

In order to give the same weight to all features we normalize the data to make them independent of units.

- Normalization 0,1:

$$v_i^n(k) = \frac{v_i(k) - min(v_i)}{max(v_i) - min(v_i)}$$

- Standardization to mean 0, sigma 1

$$v_i^n(k) = \frac{v_i(k) - \mu(v_i)}{\sigma(v_i)}$$

Some comments:

1. Multi-class classification
2. Data preparation
3. General approaches to build classifiers

Supervised Learning as Classification

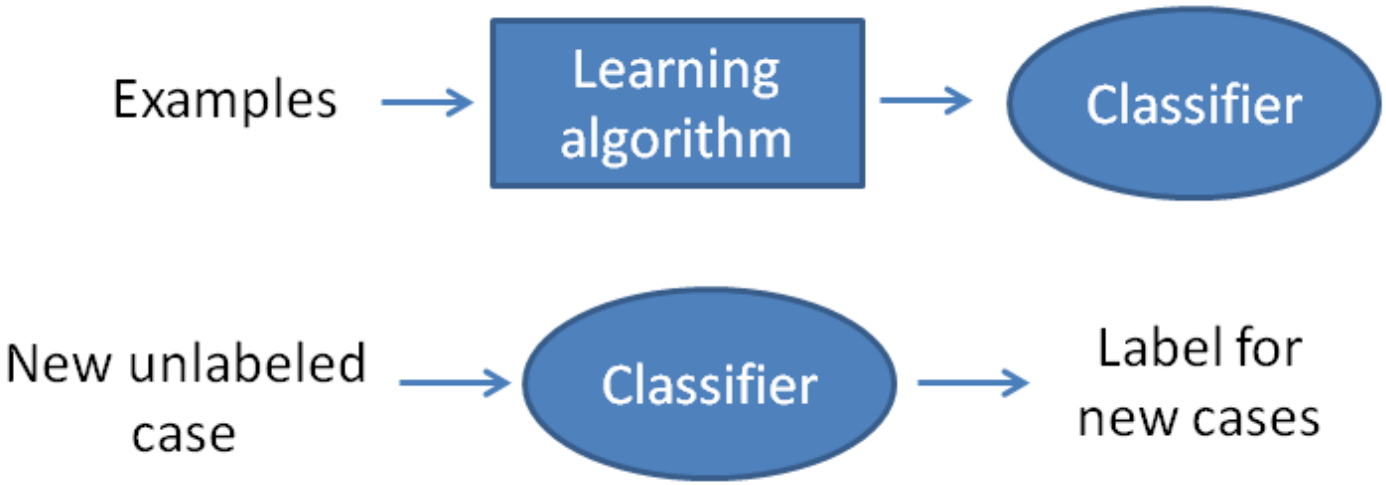
How to determine the right label for a given data? Use of learning algorithms (or data mining algorithms)

Two different approaches:

- Discriminative methods
- Generative methods

Discriminative

Build a mechanism that allows to discriminate (separate) cases of different classes without any assumption about data



Generative

Assume a parametric distribution of data, learn the parameters for the distribution (model the classes) and, when needed, find the most likely class for new observations.