Dataset preparation II:
Data transformation

Mario Martin

CS-UPC

November 6, 2019
Recap

- We have seen that data for Data mining should be represented as a table.
- We have seen ways to represent non-tabular data into tables: Images, temporal series, documents, transactions, etc.
- We have seen how to inspect your data:
  - Visually: Boxplots, Scatterplots
  - Numerically: Correlation, regression
- And clean your raw data:
  - Find and impute Missing Data
  - Finding and correcting errors
  - Finding and considering outliers

Mario Martin (CS-UPC)  Data Mining - Dataset preparation  November 6, 2019
Today

- Transformation of data:
  - Values in the table
  - Columns of the table
  - Rows of the table
Transformation of data:
  - Values in the table
  - Columns of the table
  - Rows of the table
Value transformation
Value transformation

- It means to change the values in the table
- Several reasons to do that:
  - Data homogenization
  - Re-scaling of data
  - Change of distribution
Value transformation

- It means to change the values in the table
- Several reason to do that:
  - **Data homogenization**
  - Re-scaling of data
  - Change of distribution
Data homogenization: Categorical to Numerical

- We have seen that there are a lot of possible values for a column:
  - Numerical
  - Categorical
- Some Data mining algorithms only work with numerical features
- Different ways to transform Categorical Data into numerical:
  - Label Encoding
  - One-Hot encoding
  - Mean encoding
Ordinal (Label) encoding

- When categories have an implicit order
- Examples:
  - Age: Child, Young, Adult, Old
  - Monthly Income: "0-4k", "4-10k", "10k-20k", ">20k"
**Ordinal (Label) encoding**

- Example of transformation for *Age* attribute:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Child</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Young</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Adult</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Child</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Old</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

- Coding in python and pandas:

  ```python
  Age_dict = {'Child':1, 'Young':2, 'Adult':3, 'Old':4}
  df['Age'] = df.Age.map(Age_dict)
  ```
One-Hot encoding

- When categories have no order related to target and there are not a lot of categories (f.i. color)
One-Hot encoding

- When categories have no order related to target and there are not a lot of categories (f.i. *color*)
- Ordinal encoding is not right because it induces different distances between modalities

<table>
<thead>
<tr>
<th></th>
<th>...</th>
<th>color</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_1$</td>
<td>...</td>
<td>red</td>
<td>...</td>
</tr>
<tr>
<td>$o_2$</td>
<td>...</td>
<td>blue</td>
<td>...</td>
</tr>
<tr>
<td>$o_3$</td>
<td>...</td>
<td>brown</td>
<td>...</td>
</tr>
<tr>
<td>$o_4$</td>
<td>...</td>
<td>red</td>
<td>...</td>
</tr>
<tr>
<td>$o_5$</td>
<td>...</td>
<td>green</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th></th>
<th>...</th>
<th>color</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_1$</td>
<td>...</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>$o_2$</td>
<td>...</td>
<td>2</td>
<td>...</td>
</tr>
<tr>
<td>$o_3$</td>
<td>...</td>
<td>3</td>
<td>...</td>
</tr>
<tr>
<td>$o_4$</td>
<td>...</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>$o_5$</td>
<td>...</td>
<td>4</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
One-Hot encoding

- One-Hot encoding: Create *dummy* variables, one for modality

<table>
<thead>
<tr>
<th></th>
<th>...</th>
<th><strong>color</strong></th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>o₁</td>
<td>...</td>
<td>red</td>
<td>...</td>
</tr>
<tr>
<td>o₂</td>
<td>...</td>
<td>blue</td>
<td>...</td>
</tr>
<tr>
<td>o₃</td>
<td>...</td>
<td>brown</td>
<td>...</td>
</tr>
<tr>
<td>o₄</td>
<td>...</td>
<td>red</td>
<td>...</td>
</tr>
<tr>
<td>o₅</td>
<td>...</td>
<td>green</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>...</th>
<th><strong>red?</strong></th>
<th><strong>blue?</strong></th>
<th><strong>brown?</strong></th>
<th><strong>green?</strong></th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>o₁</td>
<td>...</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>o₂</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>o₃</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>o₄</td>
<td>...</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>o₅</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>...</td>
</tr>
</tbody>
</table>

- When too many modalities you can group some values to reduce number of modalities (f.i. Grouping categorical values when there is a hierarchical structure over them - *kind of illness* example).
- Always for binary categories (only one column 0-1)
- Coding in python and pandas:
  ```python
df2 = pd.get_dummies(df, columns =['color'])
```
Mean encoding

- When goal is to predict a label, modalities can be transformed according to joint appearance with labels

<table>
<thead>
<tr>
<th></th>
<th>...</th>
<th>color</th>
<th>...</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>(o_1)</td>
<td>...</td>
<td>red</td>
<td>...</td>
<td>yes</td>
</tr>
<tr>
<td>(o_2)</td>
<td>...</td>
<td>blue</td>
<td>...</td>
<td>yes</td>
</tr>
<tr>
<td>(o_3)</td>
<td>...</td>
<td>brown</td>
<td>...</td>
<td>no</td>
</tr>
<tr>
<td>(o_4)</td>
<td>...</td>
<td>red</td>
<td>...</td>
<td>no</td>
</tr>
<tr>
<td>(o_5)</td>
<td>...</td>
<td>green</td>
<td>...</td>
<td>yes</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\[ \Rightarrow \]

<table>
<thead>
<tr>
<th></th>
<th>...</th>
<th>color</th>
<th>...</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>(o_1)</td>
<td>...</td>
<td>0.5</td>
<td>...</td>
<td>yes</td>
</tr>
<tr>
<td>(o_2)</td>
<td>...</td>
<td>1</td>
<td>...</td>
<td>yes</td>
</tr>
<tr>
<td>(o_3)</td>
<td>...</td>
<td>0</td>
<td>...</td>
<td>no</td>
</tr>
<tr>
<td>(o_4)</td>
<td>...</td>
<td>0.5</td>
<td>...</td>
<td>no</td>
</tr>
<tr>
<td>(o_5)</td>
<td>...</td>
<td>1</td>
<td>...</td>
<td>yes</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Categorical to numeric

- Not the whole story. A lot of other methods depending on the data.

**Figure:** source
Data homogenization: Numerical to Qualitative

- From qualitative to numerical
- Reducing number of values:
  - From numerical to ranges (f.i. age: 0-18, 19-40, 40-65, 67-120)
  - From numerical to qualitative (f.i. age: young, adult, senior, old)
Value transformation

- It means to change the values in the table
- Several reason to do that:
  - Data homogenization
  - **Re-scaling of data**
  - Change of distribution
Transforming values of columns

- All variables should be in the same range to avoid bias in computation of distances.
- Mandatory for some algorithms like $k$-NN, SVM or Neural Networks.
- Can be seen as change of units.
- Common procedures: Normalization or Standardization.
- Caveat: Different assumptions!
Transforming values of columns

- Normalization [0 – 1]:
  \[
  v(i) = \frac{v(i) - \text{min}(v)}{\text{max}(v) - \text{min}(v)}
  \]

```python
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(data)
d2 = scaler.transform(data)
```

- Standardization \(N(0, 1)\):
  \[
  v(i) = \frac{v(i) - \text{mean}(v)}{\text{std}(v)}
  \]

```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(data)
d2 = scaler.transform(data)
```
Value transformation

- It means to change the values in the table
- Several reason to do that:
  - Data homogenization
  - Re-scaling of data
  - Change of distribution
Value transformation

- In some cases distribution of data has heavy tails

![Histogram of Yearly income](image)

- In these cases, we have problems distinguishing between most of data.
- Data appear as *outliers*
- Solutions:
  - Apply Log of column (log-normalization)
  - Replace values by quantiles
Value transformation

- Effect of log-normalization:
Value transformation

- Replace value according to the position in the quantile
- Effect of quantiles separation when using 4 values

You can use a higher degree of quantization if needed
Today

- Transformation of data:
  - Values in the table
  - **Columns of the table**
  - Rows of the table
Work on columns
Subsection 1

Reducing number of columns: feature selection and feature extraction
Removing variables

- Very common in DM
- Why dimensionality reduction?
  1. Reduced Computing Time
  2. Increased predictive/descriptive accuracy
  3. Better representation of the data-mining model
  4. The intrinsic dimension may be small.
- Some data mining techniques may not be effective for high-dimensional data (they suffer the *Curse of dimensionality*: accuracy and efficiency may degrade rapidly as the dimension increases.)
Reducing number of columns

Two ways to reduce the number of columns:

1. **Feature selection**: A process that chooses an optimal subset of features according to an objective function
   - Feature ranking algorithms, and
   - Minimum subset algorithms.

2. **Feature extraction**: refers to the mapping of the original high-dimensional data onto a lower-dimensional space.
   - Descriptive setting: minimize the information loss
   - Predictive setting: maximize the class discrimination
Removing variables: Feature selection

**Feature selection:** Filter and feature ranking algorithms.

- Remove columns without enough variability ($std(v) < threshold$)
- Remove variables that are not enough correlated with label ($corr(v, label) < threshold$)
- Remove variables with low *Fisher score* with label ($F(v, label) < threshold$)
- Remove variables without enough Information Gain ($IG(v) < threshold$)
- Remove variables without enough Relief [Paper] ($Relief(v) < threshold$)

All these methods implicitly build a ranking of features using a given measure. After ranking, we use a threshold to select the interesting features.
Removing variables: Ranking or filter methods

Let $X_v^l$ the set of values of feature $v$ with label $l$ and $n_{v,l}$ the number of values of that set.

- **Fisher Score** computes differences in values for different labels:

$$F_v = \frac{|(\mu(X_v^1) - \mu(X_v^0)|}{\sqrt{\frac{\sigma(X_v^1)}{n_{v,1}} + \frac{\sigma(X_v^0)}{n_{v,0}}}}$$

- Example:

<table>
<thead>
<tr>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$l$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>0.2</td>
<td>0.9</td>
<td>1</td>
</tr>
<tr>
<td>0.6</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>0.7</td>
<td>0.7</td>
<td>1</td>
</tr>
<tr>
<td>0.4</td>
<td>0.9</td>
<td>1</td>
</tr>
</tbody>
</table>

$X_{v_1}^0 = \{0.3, 0.6, 0.5\}$

$X_{v_1}^1 = \{0.2, 0.7, 0.4\}$

$X_{v_2}^0 = \{0.7, 0.6, 0.5\}$

$X_{v_2}^1 = \{0.9, 0.7, 0.9\}$
Removing variables: Ranking or filter methods

\[ \sqrt{\frac{\sigma(X_{v1}^1)}{n_{v1,1}} + \frac{\sigma(X_{v1}^0)}{n_{v1,0}}} = \sqrt{\frac{0.0233}{3} + \frac{0.6333}{3}} = 0.4678 \]

\[ \sqrt{\frac{\sigma(X_{v2}^1)}{n_{v2,1}} + \frac{\sigma(X_{v2}^0)}{n_{v2,0}}} = \sqrt{\frac{0.01}{3} + \frac{0.0133}{3}} = 0.0875 \]

- Test with threshold 0.5:

\[ \frac{|\mu(X_{v1}^1) - \mu(X_{v1}^0)|}{\sqrt{\frac{\sigma(X_{v1}^1)}{n_{v1,1}} + \frac{\sigma(X_{v1}^0)}{n_{v1,0}}}} = \frac{|0.4333 - 0.4667|}{0.4678} = 0.0735 < 0.5 \]

\[ \frac{|\mu(X_{v2}^1) - \mu(X_{v2}^0)|}{\sqrt{\frac{\sigma(X_{v2}^1)}{n_{v2,1}} + \frac{\sigma(X_{v2}^0)}{n_{v2,0}}}} = \frac{|0.8333 - 0.6|}{0.0875} = 2.6667 > 0.5 \]
 Removing variables: Ranking or filter methods

- Correlation:

  \[ \rho(v, l) = \frac{\text{cov}(v, l)}{\sigma_v \sigma_l} \]

- Information Gain (Mutual Information): when \( v \) and \( l \) are independent variables

  \[ p(v, l) = p(v)p(l) \]

  \[ I(v; l) = \sum_{x \in \mathcal{D}} p(x_v, x_l) \log \frac{p(x_v, x_l)}{p(x_v)p(x_l)} = H(l) - H(l|v) \]

  where \( H \) is entropy measure
Removing variables: Ranking or filter methods

Relief algorithm:

- **Basic algorithm construct:**
  - Each feature is assigned with cumulative weight computed over a subset of the training data set.
  - Feature with weight over a certain threshold is the selected feature subset.

- **Weight computation:**
  - Let $X$ be a randomly selected sample
  - *near-hit* sample = the closest sample from the same class
  - *near-miss* sample = the closest sample from the different class.
  - Impact on weight for feature $j$ of example $X$ is computed as:
    \[ W_j = -\text{diff}_j(X, \text{near-hit})^2 + \text{diff}_j(X, \text{near-miss})^2 \]
  - Procedure repeated for several random samples
Removing variables: Ranking or filter methods

Kira and Rendell, 1992

Selected sample
**Algorithm 1**: Pseudocode for **Relief** algorithm.

**Input:**
- $N$ number of features
- $m$ number of instances sampled
- $\tau$ adjustable threshold

decrease $w=0$

**for** $i = 1$ to $m$ **do**

- $I \leftarrow$ randomly selected instance of $X$

**for** $j = 1$ to $N$ **do**

- $w(j) \leftarrow w(j) - \frac{\text{diff}_j(X, \text{near-hit})^2}{m} + \frac{\text{diff}_j(X, \text{near-miss})^2}{m}$

**end for**

**end for**

**Output:** features $j$ with $w(j) > \tau$
We have seen how to reduce features irrelevant to the learning task.

- Redundant features are those that show similar values for each instance (row) after normalization.
- You can use some of the proposed measures (e.g., correlation, MI) and thresholds to detect redundant features. When two features are redundant, remove one of them.

However, it is time-consuming because you have to test each pair of features (cost $O(f^2)$).

In general, redundant features do not hurt the performance of models, only impacts in running time and interpretation of models.
Removing variables: Redundant features

- We have seen how to reduce features irrelevant to the learning task.
- We can also reduce the number of features removing redundant features.
- Redundant features are those that show similar values for each instance (row) after normalization.
- You can use some of the proposed measures (f.i. correlation, MI) and thresholds to detect redundant features. When two features are redundant, remove one of them.
Removing variables: Redundant features

- We have seen how to reduce features irrelevant to the learning task.
- We can also reduce the number of features removing redundant features.
- Redundant features are those that show similar values for each instance (row) after normalization.
- You can use some of the proposed measures (f.i. correlation, MI) and thresholds to detect redundant features. When two features are redundant, remove one of them.
- However, it is time consuming because you have to test each pair of features (cost $O(f^2)$).
- In general redundant features do not hurt the performance of models, only impacts in running time and interpretation of models.
Removing variables: Feature selection

- How to select threshold?

(1) Rank features and use top \( n \) in ranking
(2) Plot features against values and choose threshold when values decrease slowly

Rank and cut methods could miss joint effects of variables:

Mario Martin (CS-UPC)
Data Mining - Dataset preparation
November 6, 2019
Removing variables: Feature selection

- How to select threshold? (1) Rank features and use top $n$ in ranking
How to select threshold? (1) Rank features and use top \( n \) in ranking or (2) plot features against values and choose threshold when values decrease slowly.
Removing variables: Feature selection

- How to select threshold? (1) Rank features and use top $n$ in ranking or (2) plot features against values and choose threshold when values decrease slowly
- Rank and cut methods could miss joint effects of variables:

Guyon-Elisseff, JMLR 2004; Springer 2006
Removing variables: Feature selection

Instead of evaluating features isolated, we will evaluate sets of features:

- It can be viewed as a search problem ($2^n$ sets of features to consider)
- An optimal search is not feasible, and simplifications are made to produce acceptable and timely reasonable results:
  - heuristic criteria
  - bottom-up approach
  - top-down approach
Removing variables: Feature selection

Advanced feature selection methods have the following elements:

- A Feature set generator that proposes a set of features to evaluate
- An evaluation method. Two kinds of evaluation methods:
  1. *Filter*: Uses an evaluation metric that is independent of the learning mechanism that will be used (f.i. the ones we have seen in the previous slides)
  2. *Wrapper* method: Uses a measure of the target learning method to evaluate the set of features (f.i. in SVM, the margin of the classifier obtained with that set of features)

- A criteria to stop the search for the set of features
Removing variables: Feature selection

Filter method:
Removing variables: Feature selection

Wrapper method:
Reducing number of columns

Two ways to reduce the number of columns:

1. **Feature selection**: A process that chooses an optimal subset of features according to an objective function
   - Feature ranking algorithms, and
   - Minimum subset algorithms.

2. **Feature extraction**: refers to the mapping of the original high-dimensional data onto a lower-dimensional space.
   - Descriptive setting: minimize the information loss
   - Predictive setting: maximize the class discrimination
Feature extraction: refers to the mapping of the original high-dimensional data onto a lower-dimensional space.

The new dimensions (columns) are a new and should contain the information of the original dataset in a lower number of columns.

Examples:

- **PCA** (descriptive setting): minimize the information loss embedding the dataset in a space of new columns that are build as linear combinations of the original columns. Columns are sorted by inertia, and we keep only the $n$ most informative columns.

- **LDA** (predictive setting): does like in the case of PCA an embedding in a dimensional space of columns that are build as linear combinations of the original columns, but in this case the goal is to maximize the class discrimination instead of minimal information loss.
Subsection 2

Enriching the data set: Joining tables
Enriching the data set: Joining tables

- It is common to have information stored in databases.
- Databases, for instance relational methods, store information in separate tables.
- So you can complete (enrich) your dataset with corresponding data from another table.
Enriching the data set: Joining tables

- It is common to have information stored in data bases.
- Databases, for instance relational methods, store information in separate tables.
- So you can complete (enrich) your dataset with corresponding data from another table.
- For instance think in a dataset about pulmonary diseases. It has several columns plus the identifier of the patient.
- In other dataset you have information about each patient. For instance residence.
- Joining both tables you now can relate features of the first dataset with residence, for instance.
Subsection 3

Enriching the data set: Building new variables
Building new variables: Ratios and Proportions

Sometimes your data is expressed in columns that it is difficult to manipulate with your data mining algorithm. So you add define new columns from the old ones.

For instance think in a dataset about taxis, with columns like departure point, destination point, duration of trip.

You can add to the data set new columns: length of trip, speed of taxi.

So you can complete (enrich) your dataset with columns that ease the finding of a good model.
Some algorithms only work with lineal dependencies of variables.

Generate a new feature set consisting of all “polynomial” combinations of the features with degree less than or equal to the specified degree.

For example, from three initial features

\[ \{ X_1, X_2, X_3 \} \]

add set of new variables

\[ \{ X_1X_2, X_1X_3, X_2X_3, X_1X_2X_3 \} \]

with a degree of 3.
Today

- Transformation of data:
  - Values in the table
  - Columns of the table
  - **Rows of the table**
Data processing: rows
Subsection 1

Reducing the number of rows
Reducing the number of rows

- Most DM algorithms have time complexity cost $O(n^2)$ (f.i. kNN) or even $O(n^3)$ (f.i. SVM), where $n$ is the number of rows in the dataset.
- Some algorithms need to store matrices of distances between elements, so we need $O(n^2)$ space.
- So, in some datasets we could have too many rows to build a model in reasonable time
- ... but maybe not all rows are necessary to build a model.
- So we could try to reduce the number of elements using a *sampling* procedure.
Reducing the number of rows

Some questions appear when considering sampling:
- How many examples should I sample?
- Which kind of sampling should I use?
- Am I sure that the sampling was successful?
Sampling procedure

Which kind of sampling should I use? Different kinds of sampling

- **Systematic sampling**: get every $k$-th sample in the dataset. For instance if I want 50% sampling, get one and the next one no.

- **Problem!**: when there are regularities in data set (data set sorted, for instance)
Sampling procedure

Which kind of sampling should I use? Different kinds of sampling

- **Systematic sampling**: get every k-th sample in the dataset. For instance if I want 50% sampling, get one and the next one no.

- **Problem!**: when there are regularities in data set (data set sorted, for instance)

- **Random sampling**: Two different choices
  - random sampling without replacement,
  - random sampling with replacement.

- **Problem!**: Is my sampling representative of the complete dataset?
Sampling procedure

Which kind of sampling should I use? Different kinds of sampling

- *Stratified sampling*: Split data set into non-overlapping subsets (strata) and combine strata results.
- Used in general to ensure same proportion of each class of examples in the sample.

- **Problem!**: Ok. Class values are representative, but What happens with other columns?
Sampling procedure

Which kind of sampling should I use? Different kinds of sampling

- **Multi-Stratified sampling**: Do a Stratified sampling, and after that, verify that for each column we have a similar statistical structure to the original dataset (histogram of modalities in qualitative features, mean for numerical values). If the sampled set does not pass the test, throw it and do another sampling until one that passes the test is found.
Number of examples

How many examples should I sample?

- Error should be not different than using complete dataset.
- One solution could be **incremental sampling**: In consists in starting with a small sample, measure the accuracy with the learning algorithm and increase the size of the sampling until no improvement in accuracy is found.
How many examples should I sample?

- Data should be representative of the original dataset.
- **Inverse sampling**: Used when features with rare frequencies of values (skewed distributions): Consists in the dynamic increase of examples sampled until some condition about feature are satisfied (f.i. all values appear at least $t$ times)
With sampling procedures we usually waste examples. One approach that could help in polynomial cost algorithms is to use ensemble techniques.

Assume a SVM with training cost $O(n^3)$ and $n = 10^6$ examples:

$$O((10^6)^3) = 10^{18}$$
Number of examples

- With sampling procedures we usually waste examples. One approach that could help in polynomial cost algorithms is to use ensemble techniques.
- Assume a SVM with training cost $O(n^3)$ and $n = 10^6$ examples:

$$O((10^6)^3) = 10^{18}$$

- Consider the following procedure:
  - Divide the dataset in 10 parts
  - Build an SVMs on each part (we have 10 SVMs)
  - Combine by *majority vote* the output of each classifier in testing
With sampling procedures we usually waste examples. One approach that could help in polynomial cost algorithms is to use ensemble techniques.

Assume a SVM with training cost $O(n^3)$ and $n = 10^6$ examples:

$$O((10^6)^3) = 10^{18}$$

Consider the following procedure:

- Divide the dataset in 10 parts
- Build an SVMs on each part (we have 10 SVMs)
- Combine by *majority vote* the output of each classifier in testing
- Now training cost is:

$$\sum_{i=1}^{10} O((10^5)^3) = 10^{16}$$
Subsection 2

Increasing the number of rows
Number of examples

- In some cases we don’t have enough data
  - Small datasets
  - Unbalanced datasets: We have several examples but one of the classes has very few examples
- We need more data, but how to generate data?
- Two strategies: resampling and data generation
Sampling procedures

Cases Reduction: Random Sampling

Raw Data

SRSWOR
(simple random sample without replacement)

SRSWR
(simple random sample with replacement)
More commonly we deal with Unbalanced datasets.

With severely unbalanced datasets it is extremely difficult to obtain good results because the smaller class is a lot underrepresented.

Data Mining algorithm biased towards majority class (usually not interesting)
Unbalanced datasets

- More commonly we deal with Unbalanced datasets.
- With severely unbalanced datasets it is extremely difficult to obtain good results because the smaller class is a lot underrepresented.
- Data Mining algorithm biased towards majority class (usually not interesting)
- *Accuracy* measure is misleading in these cases: in this case you should report results of *precision*, *recall* and *F-measure* on the smaller class.
Unbalanced datasets

- More commonly we deal with Unbalanced datasets.
- With severely unbalanced datasets it is extremely difficult to obtain good results because the smaller class is a lot underrepresented.
- Data Mining algorithm biased towards majority class (usually not interesting)
- *Accuracy* measure is misleading in these cases: in this case you should report results of *precision*, *recall* and *F-measure* on the smaller class.
- Try to use a cost function when building the classifier to penalize different kinds of errors in the confusion matrix.
With unbalanced datasets you can apply one of the following techniques based on resampling:

- **Undersampling**: Force a more similar representation of the smaller class by removing examples of the more populated class.

- **Oversampling**: Force a more similar representation of the smaller class by repeating in the dataset several times the examples of the less common class.

- **Generate new examples of the smaller class.** For instance (SMOTE), choose randomly two examples of the smaller class and compute the average between them. This average is included as an example of the smaller class. Repeat until you have enough examples of the smaller class.

Remember always to test on untouched validation dataset.
Unbalanced datasets

- With unbalanced datasets you can apply one of the following techniques based on resampling:
  - *Undersampling*: Force a more similar representation of the smaller class by removing examples of the more populated class
  - *Oversampling*: Force a more similar representation of the smaller class by repeating in the dataset several times the examples of the less common class.

  *Generate new examples of the smaller class. For instance (SMOTE), choose randomly two examples of the smaller class and compute the average between them. This average is included as an example of the smaller class. Repeat until you have enough examples of the smaller class.*

  Remember always to test on untouched validation dataset.
With unbalanced datasets you can apply one of the following techniques based on resampling:

- **Undersampling**: Force a more similar representation of the smaller class by removing examples of the more populated class.

- **Oversampling**: Force a more similar representation of the smaller class by repeating in the dataset several times the examples of the less common class.

- **Generate new examples** of the smaller class. For instance (SMOTE), choose randomly two examples of the smaller class and compute the average between them. This average is included as a example of the smaller class. Repeat until you have enough examples of the smaller class.

Remember always to test on untouched validation dataset.
Unbalanced datasets

- With unbalanced datasets you can apply one of the following techniques based on resampling:
  - **Undersampling**: Force a more similar representation of the smaller class by removing examples of the more populated class.
  - **Oversampling**: Force a more similar representation of the smaller class by repeating in the dataset several times the examples of the less common class.
  - **Generate new examples** of the smaller class. For instance (SMOTE), choose randomly two examples of the smaller class and compute the average between them. This average is included as a example of the smaller class. Repeat until you have enough examples of the smaller class.

- Remember always to test on untouched validation dataset.