Data preprocessing

Data Systems Group - UT
Radwa El Shawi

Data Preprocessing

- Data Preprocessing: An Overview
  - Data Quality
  - Major Tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- Data Reduction
- Data Transformation and Data Discretization
- Handling imbalanced data
- Summary

Data Transformation

- A function that maps the entire set of values of a given attribute to a new set of replacement values s.t. each old value can be identified with one of the new values
- Methods
  - Smoothing: Remove noise from data
  - Attribute/feature construction
    - New attributes constructed from the given ones
  - Aggregation: Summarization, data cube construction
  - Normalization: Scaled to fall within a smaller, specified range
    - min-max normalization
    - z-score normalization
    - normalization by decimal scaling
  - Discretization - Concept hierarchy generation

Normalization

- **Min-max normalization**: to [new_min$_A$, new_max$_A$]

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

  - Ex. Let income range \$12,000 to \$98,000 normalized to [0.0, 1.0]. Then \$73,600 is mapped to $\frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000}(1.0 - 0) + 0 = 0.716$

- **Z-score normalization** (μ: mean, σ: standard deviation):

$$v' = \frac{v - \mu_A}{\sigma_A}$$

  - Ex. Let μ = 54,000, σ = 16,000. Then $\frac{73{,}600 - 54{,}000}{16{,}000} = 1.225$

- **Normalization by decimal scaling**

$$v' = \frac{v}{10^j}$$ Where j is the smallest integer such that Max(|v'|) < 1

Normalization(Cont.)

- **Normalization by decimal scaling**

$$v' = \frac{v}{10^j}$$

Where j is the smallest integer such that Max(|v'|) < 1

- Ex. Let the recorded values of A range from −986 to 917. The maximum absolute value of A is 986. To normalize by decimal scaling, we therefore divide each value by 1000 (i.e., j = 3) so that −986 normalizes to −0.986 and 917 normalizes to 0.917.

Discretization

- Discretization: Divide the range of a continuous attribute into intervals
  - Interval labels can then be used to replace actual data values
  - Reduce data size by discretization

age	5	6	6	9	…	15	16	16	17	20	…	24	25	41	50	65	…	67
own a car	0	0	0	0	…	0	1	0	1	1	…	0	1	1	1	1	…	1

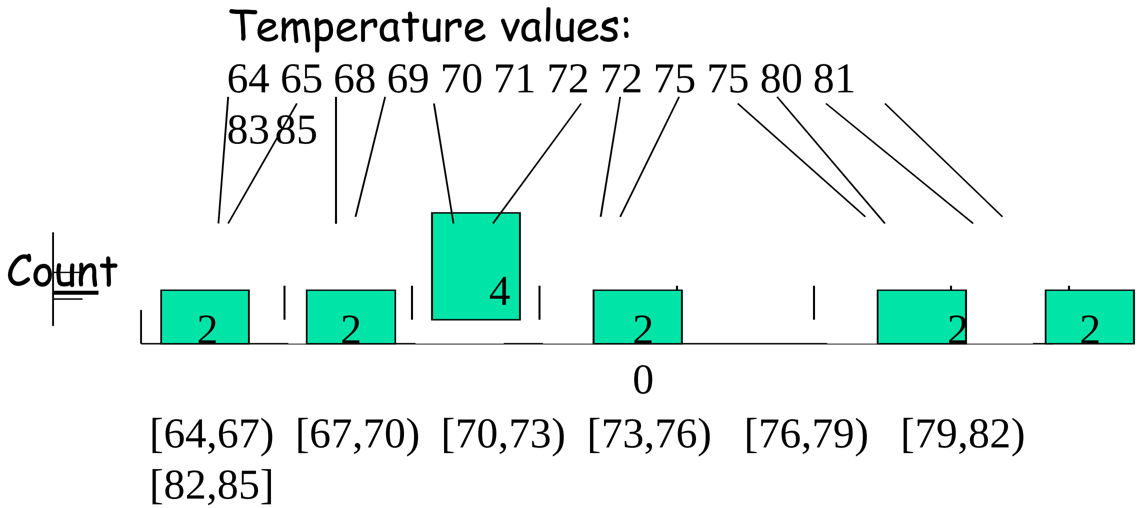
Age [5,15]   Age [16,24]   Age [25,67]

Data Discretization Methods

- Typical methods: All the methods can be applied recursively
  - Binning
    - Top-down split, unsupervised
  - Histogram analysis
    - Top-down split, unsupervised
  - Clustering analysis (unsupervised, top-down split or bottom-up merge)
  - Decision-tree analysis (supervised, top-down split)
  - Correlation (e.g., χ^2) analysis (unsupervised, bottom-up merge)

Simple Discretization: Binning

- Equal-width (distance) partitioning
  - It divides the range into N intervals of equal size (range): uniform grid
  - If A and B are the lowest and highest values of the attribute, the width of intervals will be: W = (B -A)/N.

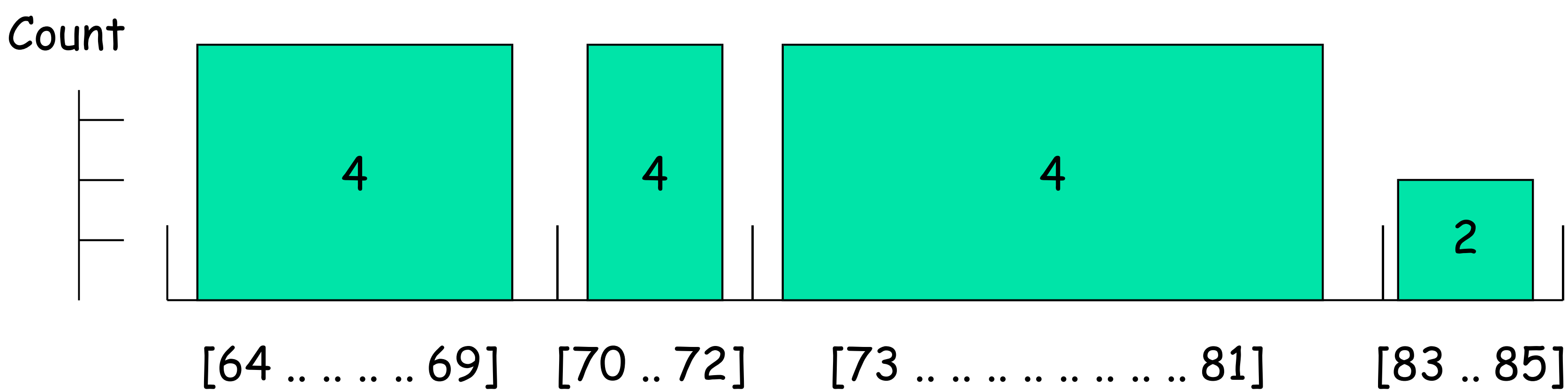


Equal Width, bins Low <= value < High

Simple Discretization: Binning (Cont.)

- Equal-depth (frequency) partitioning
  - Divides the range into *N* intervals, each containing approximately same number of samples

Temperature values:
64 65 68 69 70 71 72 72 75 75 80 81 83 85



Equal Height = 4, except for the last bin

Entropy Based Discretization

Class dependent

1. Sort examples in increasing order
2. Each value forms an interval ('m' intervals)
3. Calculate the entropy measure of this discretization
4. Find the binary split boundary that minimizes the entropy function over all possible boundaries. The split is selected as a binary discretization.

$$E(S,T) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2)$$

5. Apply the process recursively until some stopping criterion is met, e.g.,

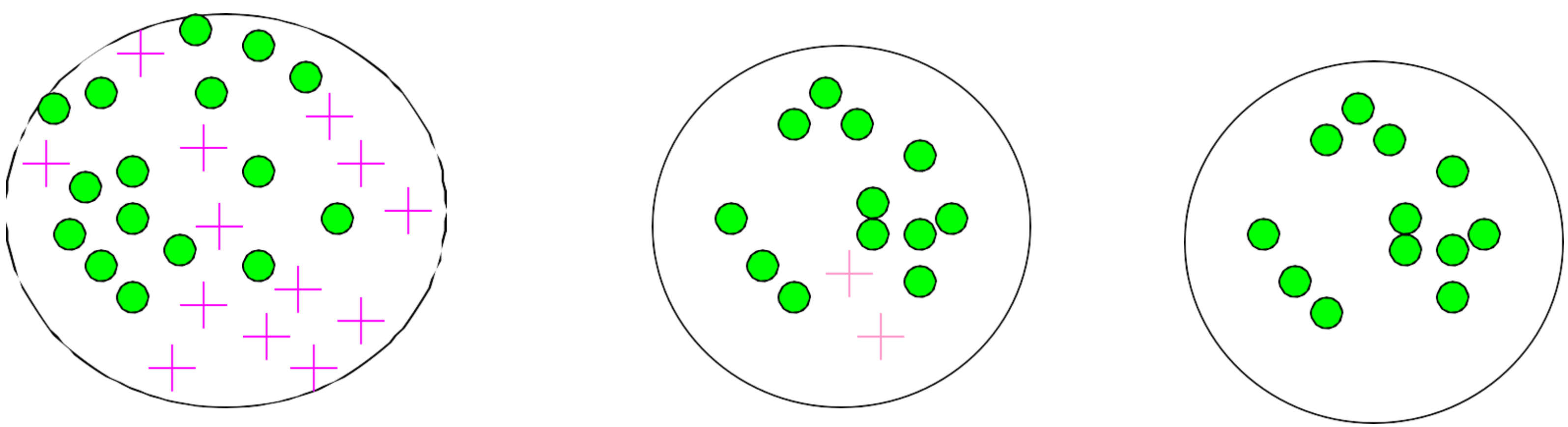
$$Ent(S) - E(T,S) > \delta$$

Entropy/Impurity

- S - training set, $C_1,\ldots,C_N$ classes
- Entropy Ent(S) - measure of the impurity in a group of examples
  - p_c - proportion of C_c in S

$$Ent(S) = -\sum_{c=1}^{N} p_c \cdot \log_2 p_c$$

Very impure group | Less impure | Minimum impurity



high entropy | null entropy

An example of entropy

Test split temp < 71.5

Temp.	Play?
64	Yes
65	No
68	Yes
69	Yes
70	Yes
71	No
72	No
72	Yes
75	Yes
75	Yes
80	No
81	Yes
83	Yes
85	No

(4 yes, 2 no)

(5 yes, 3 no)

	yes	no
< 71.5	4	2
> 71.5	5	3

$$Ent(split\ 71.5) = \frac{6}{14} \cdot \left(\frac{4}{6} \log \frac{4}{6} + \frac{2}{6} \log \frac{2}{6} \right) + \frac{8}{14} \cdot \left(\frac{5}{8} \log \frac{5}{8} + \frac{3}{8} \log \frac{3}{8} \right) = 0.939$$

	yes	no
< 77	7	3
> 77	2	2

$$Ent(split\ 77) = \frac{10}{14} \cdot \left(\frac{7}{10} \log \frac{7}{10} + \frac{3}{10} \log \frac{3}{10} \right) + \frac{4}{14} \cdot \left(\frac{2}{4} \log \frac{2}{4} + \frac{2}{4} \log \frac{2}{4} \right) = \boxed{0.915}$$

An example (cont.)

Temp.	Play?
64	Yes
65	No
68	Yes
69	Yes
70	Yes
71	No
72	No
72	Yes
75	Yes
75	Yes
80	No
81	Yes
83	Yes
85	No

6th split

5th split

4th split

3rd split

2nd split

1st split

The method tests all split possibilities and chooses the split with smallest entropy.

In the first iteration a split at 84 is chosen.

The two resulting branches are processed recursively.

The fact that recursion only occurs in the first interval in this example is an artifact. In general both intervals have to be split.

Discretization Correlation Analysis

- Compute the $\chi 2$ value for each pair of adjacent intervals
- Merge the pair of adjacent intervals with the lowest $\chi 2$ value
- Repeat the above steps and until $\chi 2$ values of all adjacent pairs exceeds a threshold

ChiMerge Discretization Example

Sample	F	K
1	1	1
2	3	2
3	7	1
4	8	1
5	9	1
6	11	2
7	23	2
8	37	1
9	39	2
10	45	1
11	46	1
12	59	1

Intervals

{0,2}
{2,5}
{5,7.5}
{7.5,8.5}
{8.5,10}
{10,17}
{17,30}
{30,38}
{38,42}
{42,45.5}
{45.5,52}
{52,60}

- Sort and order the attributes that you want to group (in this example attribute F).
- Start with having every unique value in the attribute be in its own interval.

ChiMerge Discretization Example (Cont.)

Sample	F	K
1	1	1
2	3	2
3	7	1
4	8	1
5	9	1
6	11	2
7	23	2
8	37	1
9	39	2
10	45	1
11	46	1
12	59	1

- Begin calculating the Chi Square test on every interval

Sample	K=1	K=2	
2	0	1	1
3	1	0	1
total	1	1	2

Sample	K=1	K=2	
3	1	0	1
4	1	0	1
total	2	0	2

ChiMerge Discretization Example (Cont.)

Sample	K=1	K=2	
2	0	1	1
3	1	0	1
total	1	1	2

E_{11} = (1/2)*1 = .05
E_{12} = (1/2)*1 = .05
E_{21} = (1/2)*1 = .05
E_{22} = (1/2)*1 = .05

$\mathbf{X^2} = (0\text{-}.5)^2/.5 + (1\text{-}.5)^2/.5 + (1\text{-}.5)^2/.5 + (0\text{-}.5)^2/.5 = \mathbf{2}$

Sample	K=1	K=2	
3	1	0	1
4	1	0	1
total	2	0	2

E_{11} = (1/2)*2 = 1
E_{12} = (0/2)*2 = 0
E_{21} = (1/2)*2 = 1
E_{22} = (0/2)*2 = 0

$\mathbf{X^2} = (1\text{-}1)^2/1+(0\text{-}0)^2/0+ (1\text{-}1)^2/1+(0\text{-}0)^2/0 = \mathbf{0}$

Threshold .1 with df=1 from Chi square distribution chart merge if $\mathbf{X^2} < 2.7024$

ChiMerge Discretization Example

Sample	F	K
1	1	1
2	3	2
3	7	1
4	8	1
5	9	1
6	11	2
7	23	2
8	37	1
9	39	2
10	45	1
11	46	1
12	59	1

Intervals

{0,2}
{2,5}
{5,7.5}
{7.5,8.5}
{8.5,10}
{10,17}
{17,30}
{30,38}
{38,42}
{42,45.5}
{45.5,52}
{52,60}

Chi^2

2
2
0
0
2
0
2
2
2
0
0

- Calculate all the Chi Square value for all intervals
- Merge the intervals with the smallest Chi values

ChiMerge Discretization Example (Cont.)

Sample	F	K
1	1	1
2	3	2
3	7	1
4	8	1
5	9	1
6	11	2
7	23	2
8	37	1
9	39	2
10	45	1
11	46	1
12	59	1

Intervals

{0,2}

{2,5}

{5,10}

{10,30}

{30,38}

{38,42}

{42,60}

Chi^2

2

4

5

3

2

4

- Repeat

ChiMerge Discretization Example (Cont.)

Sample	F	K
1	1	1
2	3	2
3	7	1
4	8	1
5	9	1
6	11	2
7	23	2
8	37	1
9	39	2
10	45	1
11	46	1
12	59	1

Intervals Chi^2

{0,5}

1.875

{5,10}

5

{10,30}

1.33

{30,42}

1.875

{42,60}

- Again

ChiMerge Discretization Example (Cont.)

Sample	F	K
1	1	1
2	3	2
3	7	1
4	8	1
5	9	1
6	11	2
7	23	2
8	37	1
9	39	2
10	45	1
11	46	1
12	59	1

Intervals	Chi^2
{0,5}	
	1.875
{5,10}	
	3.93
{10,42}	
	3.93
{42,60}	

- Until

ChiMerge Discretization Example

Sample	F	K
1	1	1
2	3	2
3	7	1
4	8	1
5	9	1
6	11	2
7	23	2
8	37	1
9	39	2
10	45	1
11	46	1
12	59	1

Intervals	Chi^2
{0,10}	
	2.72
{10,42}	
	3.93
{42,60}	

- There are no more intervals that can satisfy the Chi Square test.

Concept Hierarchy Generation

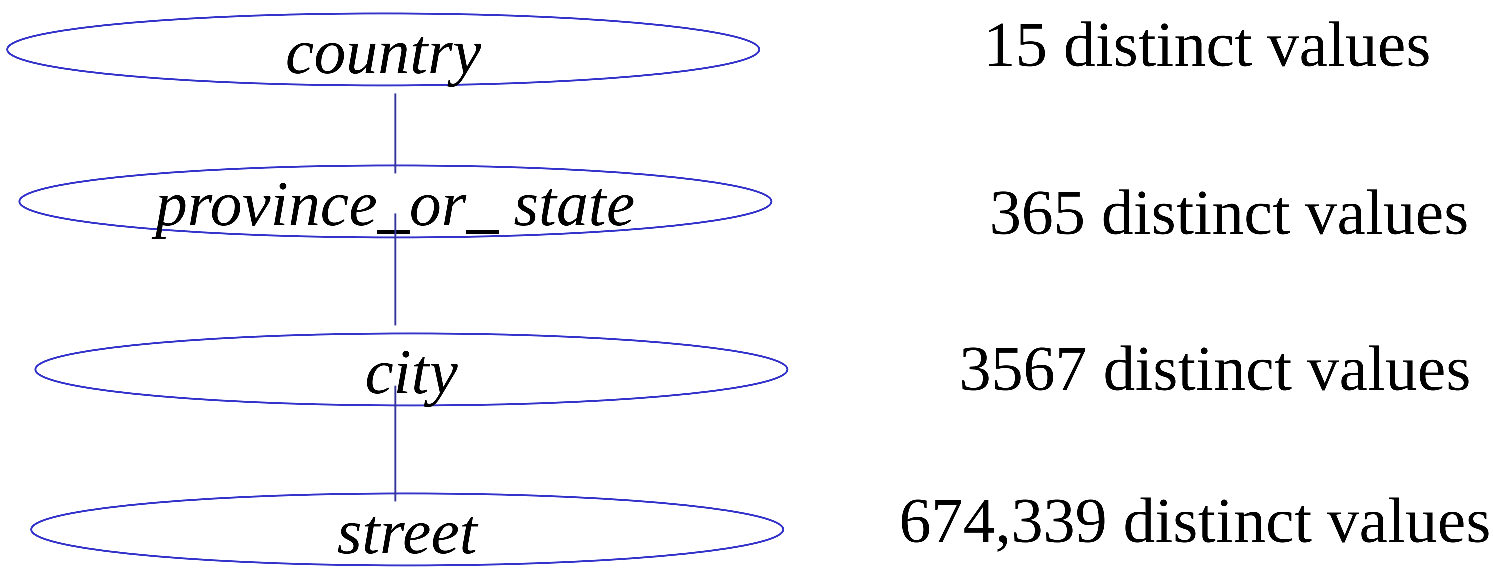
- **Concept hierarchy** organizes concepts (i.e., attribute values) hierarchically and is usually associated with each dimension in a data warehouse
- Concept hierarchies can be explicitly specified by domain experts and/or data warehouse designers
- Concept hierarchies facilitate drilling and rolling in data warehouses to view data in multiple granularity
- Concept hierarchy formation: Recursively reduce the data by collecting and replacing low level concepts (such as numeric values for *age*) by higher level concepts (such as *youth, adult*, or *senior*)
- Concept hierarchy can be automatically formed for both numeric and nominal data. For numeric data, use discretization methods shown.

Concept Hierarchy Generation for Nominal Data

- Specification of ordering of attributes explicitly at the schema level by users or experts
  - *street* < *city* < *state* < *country*
- Specification of only a partial set of attributes
  - E.g., only *street* < *city*, not others
- Automatic generation of hierarchies (or attribute levels) by the analysis of the number of distinct values
  - E.g., for a set of attributes: {*street, city, state, country*}

Automatic Concept Hierarchy Generation

- Some hierarchies can be automatically generated based on the analysis of the number of distinct values per attribute in the data set
  - The attribute with the most distinct values is placed at the lowest level of the hierarchy
  - Exceptions, e.g., weekday, month, quarter, year



Data Preprocessing

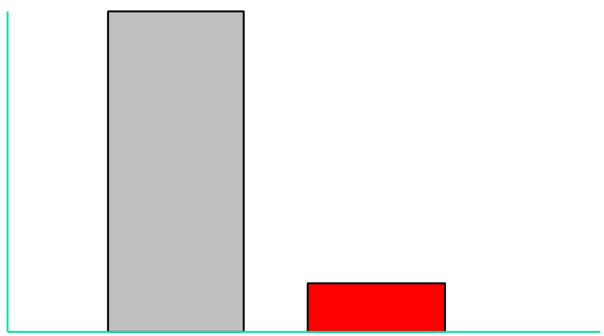
- Data Preprocessing: An Overview
  - Data Quality
  - Major Tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- Data Reduction
- Data Transformation and Data Discretization
- Handling imbalanced data

Handling imbalanced data

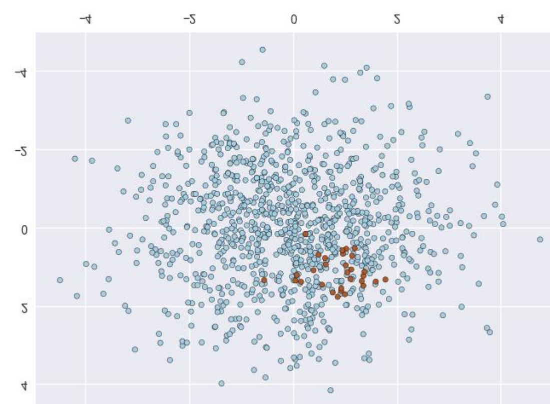
- Random over/under sampling
- SMOTE
- Cluster-based oversampling (CBO)
- One-sided Selection
- Cost sensitive classification

Problem definition

- What is **class imbalanced problem** ?
- It is the problem when the number of examples belonged to a class is significantly greater than those of the others.
- For example:
  - In cancer data, the number of patients who have cancer is much smaller than that who don’t.



Majority (Negative) class Minority (Positive) class



Handling imbalanced data (cont.)

You trained a model to predict cancer

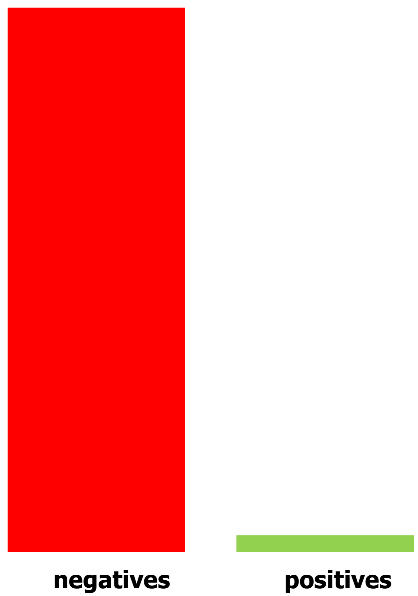
Your model has an accuracy of 99.9%

Handling imbalanced data (cont.)

By looking at the confusion matrix you realize that the model does not detect any of the positive examples.

Handling imbalanced data (cont.)

After plotting your class distribution you see that you have thousands of negative examples but just a few number of positives.



Handling imbalanced data (cont.)

Classifiers try to reduce the overall error so they can be biased towards the majority class.

Negatives = 998

Positives = 2

By always predicting a negative class the accuracy will be 99.8% !!

What can you do?

- Collect more data (difficult in many domains)
- Delete data from the majority class (undersampling)
- Create synthetic data (Oversampling)
- Adapt your learning algorithm (cost sensitive classification)

Random over/under sampling

- Random *oversampling*: randomly duplicate data points from the minority class.
- Random undersampling: randomly delete data points from the majority class.
- Limitations
  - Loss of information (in the case of under sampling)
  - Overfitting

One-sided Selection (OSS)

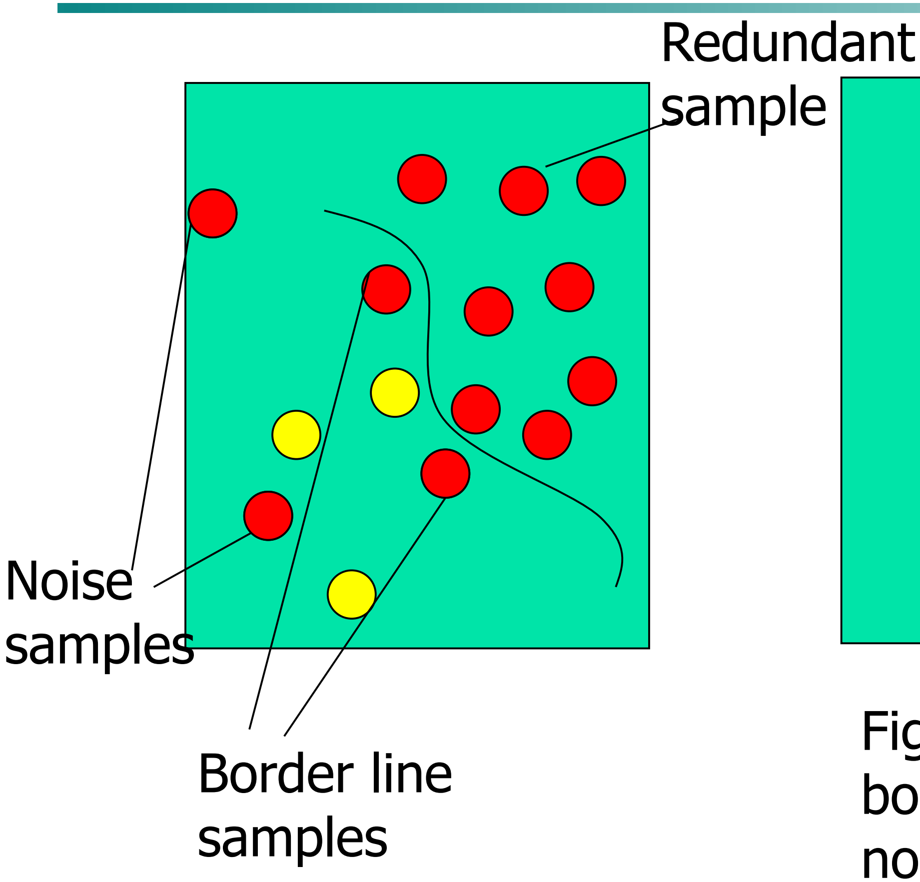


Fig1: Original data

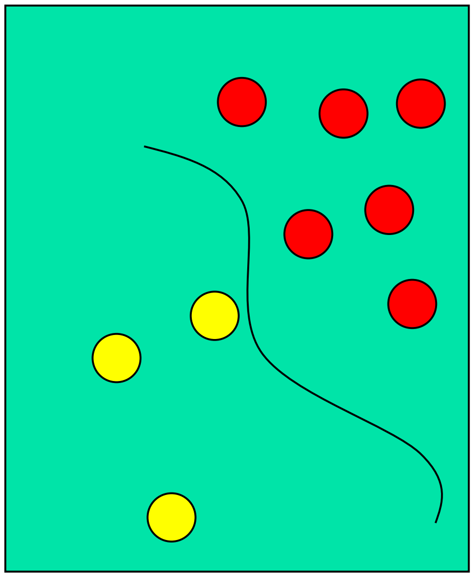


Fig2: removing borderline and nosiy samples from the majority class

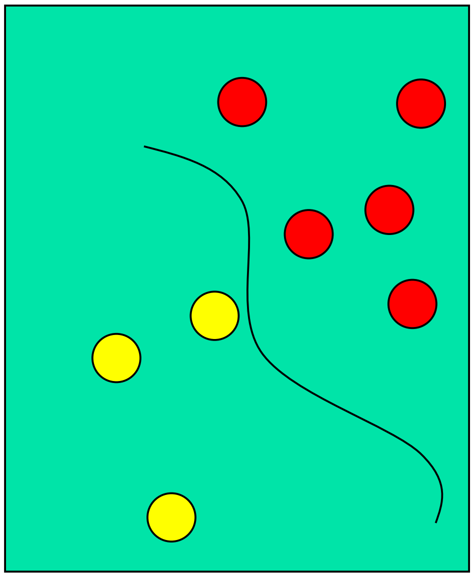


Fig3: removing redundant samples from the majority class

One-sided Selection (OSS)

- It is underdamping technique
  - OSS first chooses one instance x of the majority class at random.
  - Use the instances of the minority class and x as training data
  - Apply the k-Nearest Neighbors (KNN) algorithm with k= 1 to classify the remaining instances of the majority class.
  - The correctly classified instances are then excluded
  - uses a data cleaning technique to remove noisy data from the majority class.

Synthetic Minority Over-sampling Technique (SMOTE)

Oversampling: State-of-the-art algorithm, SMOTE

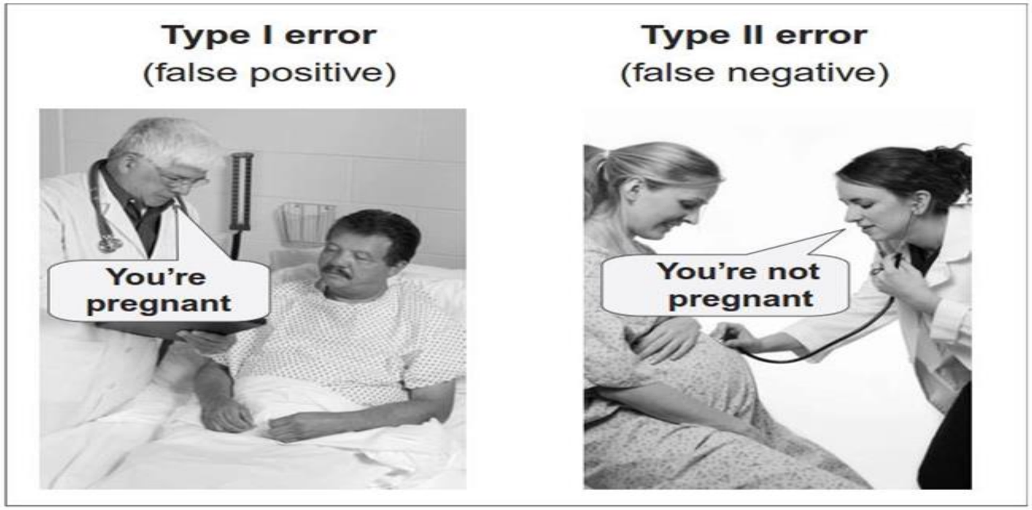
- Synthetic samples are generated in the following way:
  - Take the difference between the feature vector (sample) under consideration and its nearest neighbor.
  - Multiply this difference by a random number between 0 and 1
  - Add it to the feature vector under consideration.

Consider a sample (6,4) and let (4,3) be its nearest neighbor.
(6,4) is the sample for which k-nearest neighbors are being identified
(4,3) is one of its k-nearest neighbors.
Let:
f1_1 = 6 f2_1 = 4 f2_1 - f1_1 = -2
f1_2 = 4 f2_2 = 3 f2_2 - f1_2 = -1
The new samples will be generated as
(f1',f2') = (6,4) + rand(0-1) * (-2,-1)
rand(0-1) generates a random number between 0 and 1.

Cluster-based oversampling (CBO) method

1. For the majority class S_{maj} with m_{maj} clusters
   I. Oversample each cluster $C_{maj:j} \subset S_{maj}, j = 1, \ldots, m_{maj}$ except the largest $C_{maj:\max}$, so that for $\forall j, \left|C_{maj:j}\right| = |C_{maj:\max}|$
   II. Calculate the number of majority class examples after oversampling as N_{CBO}
2. For the minority class S_{min} with m_{min} clusters
   I. Oversample each cluster $C_{min:i} \subset S_{min}, i = 1, \ldots, m_{\min}$ to be of the same size $N_{CBO}/m_{\min}$, so that for $\forall i, |C_{min:i}| = N_{CBO}/m_{\min}$

Cost-sensitive classification



		Predicted class	
		Negative	Positive
Actual class	Negative	0	$Cost[0]$
	Positive	$Cost[1]$	0

$Cost[0]$: Cost of misclassifying negative example as positive
$Cost[1]$: Cost of misclassifying positive example as negative

		Predicted Class	
		No	Yes
Observed Class	No	TN	FP
	Yes	FN	TP

TN	True Negative
FP	False Positive
FN	False Negative
TP	True Positive

Model Performance

Accuracy	= (TN+TP)/(TN+FP+FN+TP)
Precision	= TP/(FP+TP)
Sensitivity	= TP/(TP+FN)
Specificity	= TN/(TN+FP)

Cost-sensitive learning

- Needs a cost matrix, which encodes the penalty of misclassifying samples.
- In the scenario of imbalanced data-sets, the significance of the recognition of positive instances might be higher:
- Embed a weight w_i into the i-th class, we get the weighted classification accuracy (WCA) as

$$WCA = w_1 \cdot \frac{TP}{TP + FN} + w_2 \cdot \frac{TN}{TN + FP}$$

Cost-sensitive classification with Weka

- Naïve Bayes.

```
=== Summary ===

Correctly Classified Instances         754               75.4    %
Incorrectly Classified Instances       246               24.6    %
Kappa statistic                          0.3813
Mean absolute error                      0.2936
Root mean squared error                  0.4201
Relative absolute error                 69.8801 %
Root relative squared error             91.6718 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.864    0.503    0.800      0.864    0.831      0.385    0.787     0.891     good
                 0.497    0.136    0.611      0.497    0.548      0.385    0.787     0.577     bad
Weighted Avg.    0.754    0.393    0.743      0.754    0.746      0.385    0.787     0.797

=== Confusion Matrix ===

   a   b   <-- classified as
 605  95 |   a = good
 151 149 |   b = bad
```

TP

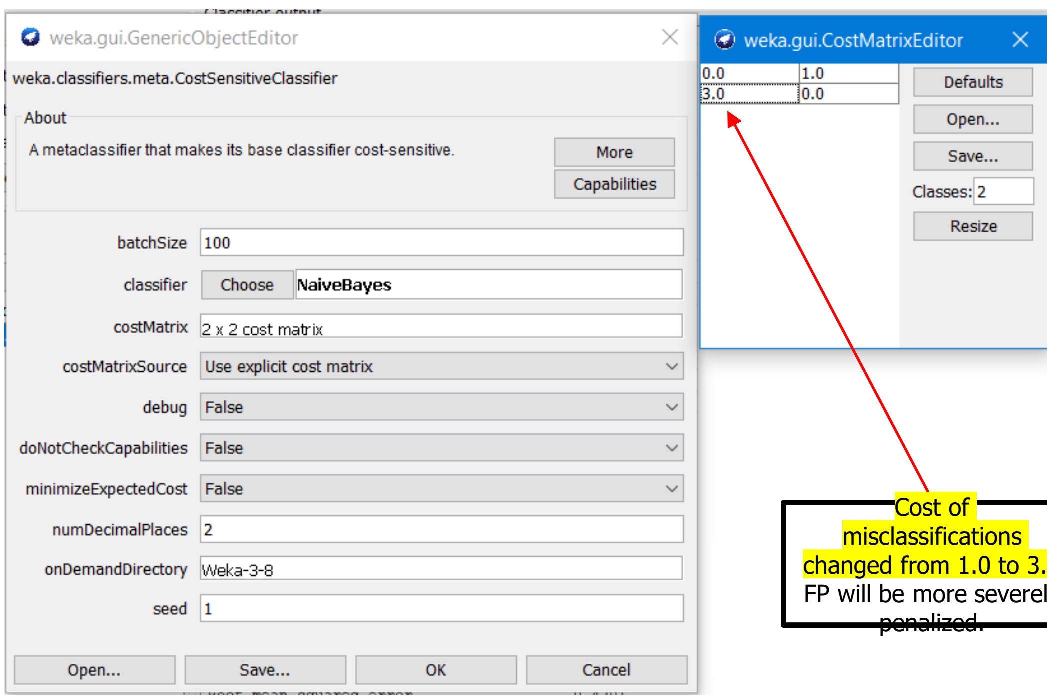
FN

FP

TN

Cost-sensitive classification with Weka

- Cost-sensitive classifier with Naïve Bayes.



Cost of misclassifications changed from 1.0 to 3.0
FP will be more severely penalized.

$$w_2 = 0.75$$
$$w_1 = 0.25$$

Cost-sensitive classification with Weka

- Cost-sensitive classifier with Naïve Bayes.

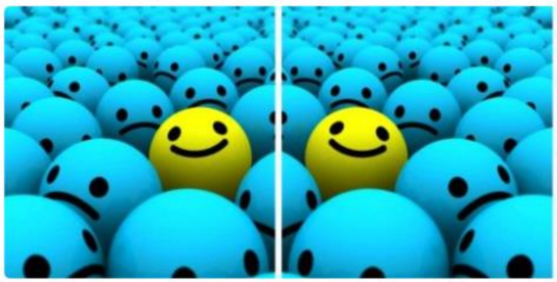
```
=== Summary ===

Correctly Classified Instances         720               72      %
Incorrectly Classified Instances       280               28      %
Kappa statistic                          0.4078
Mean absolute error                      0.3367
Root mean squared error                  0.4448
Relative absolute error                 80.1422 %
Root relative squared error             97.068  %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.706    0.247    0.870      0.706    0.779      0.425    0.787     0.891     good
                 0.753    0.294    0.523      0.753    0.617      0.425    0.787     0.578     bad
Weighted Avg.    0.720    0.261    0.766      0.720    0.731      0.425    0.787     0.797

=== Confusion Matrix ===

   a   b   <-- classified as
 494 206 |   a = good
  74 226 |   b = bad
```

But also TP reduced

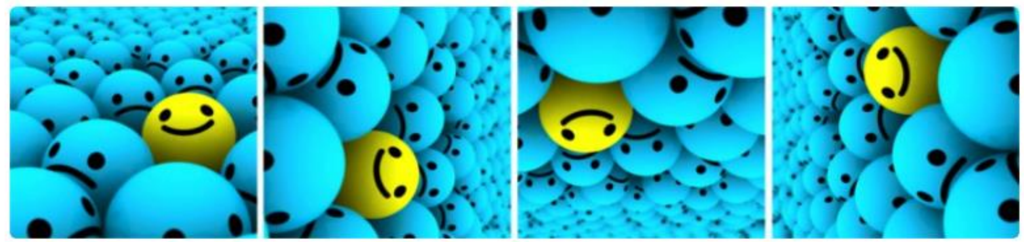
FPs reduced

For images

- Augment training data by applying image transformations: flip, rotate, scale, shift, etc.



(a) Flip



(b) Rotation

(c) Scale

(d) Crop

(d) Translation

Tools

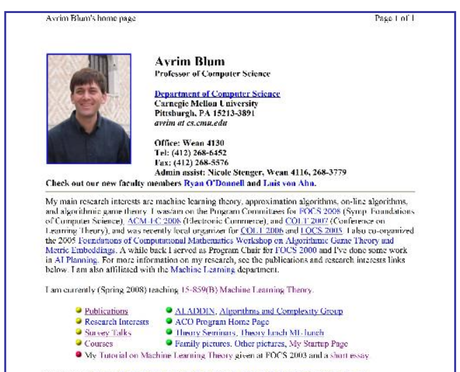
- Python imbalanced-learn library: https://github.com/scikit-learn-contrib/imbalanced-learn
- Weka also has oversampling methods and a cost sensitive meta classifier: https://weka.wikispaces.com/CostSensitiveClassifier

Another approach: Co-training

- Again, learning with a small labeled set and a large unlabeled set.
- The attributes describing each example or instance can be partitioned into two subsets. Each of them is sufficient for learning the target function.

E.g., classifying webpages: can use words on page or words on links pointing to the page.

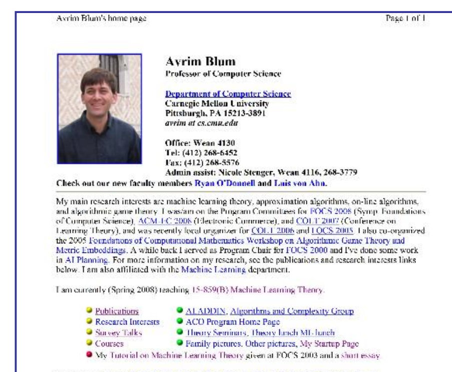
Prof. Avrim Blum My Advisor



x - Link info & Text info

Prof. Avrim Blum My Advisor

x_1- Link info



x_2- Text info

Co-training Algorithm (variant 1)

Given:
- Labeled data ***L***
- Unlabeled data ***U***

Loop
- Train g1 (hyperlink classifier) using ***L***
- Train g2 (page classifier) using ***L***
- Sample N1 points from U, let g1 label p1 positives and n1 negatives
- Sample N2 points from U, let g2 label p2 positives and n2 negatives
- Add self-labeled (N1+N2) examples to ***L***

Co-training Algorithm (variant 2)

Given:

- Labeled data ***L***
- Unlabeled data ***U***

Loop

- Train g1 (hyperlink classifier) using ***L***
- Train g2 (page classifier) using ***L***
- Sample N1 points from U, let g1 label p1 positives and n1 negatives
- Sample N2 points from U, let g2 label p2 positives and n2 negatives
- Add self-labeled n1 < N1 (examples where g1 is more confident) and n2 < N2 (examples where g2 is more confident) to ***L***

Summary

- **Data quality**: accuracy, completeness, consistency, timeliness, believability, interpretability
- **Data cleaning**: e.g. missing/noisy values, outliers
- **Data integration** from multiple sources:
  - Entity identification problem
  - Remove redundancies
  - Detect inconsistencies
- **Data reduction**
  - Dimensionality reduction
  - Numerosity reduction
- **Data transformation and data discretization**
  - Normalization
  - Concept hierarchy generation
- **Techniques to handle imbalanced data**

References

- D. P. Ballou and G. K. Tayi. Enhancing data quality in data warehouse environments. Comm. of ACM, 42:73-78, 1999
- A. Bruce, D. Donoho, and H.-Y. Gao. Wavelet analysis. *IEEE Spectrum*, Oct 1996
- T. Dasu and T. Johnson. Exploratory Data Mining and Data Cleaning. John Wiley, 2003
- J. Devore and R. Peck. *Statistics: The Exploration and Analysis of Data*. Duxbury Press, 1997.
- H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C.-A. Saita. Declarative data cleaning: Language, model, and algorithms. *VLDB'01*
- M. Hua and J. Pei. Cleaning disguised missing data: A heuristic approach. *KDD'07*
- H. V. Jagadish, et al., Special Issue on Data Reduction Techniques. Bulletin of the Technical Committee on Data Engineering, 20(4), Dec. 1997
- H. Liu and H. Motoda (eds.). *Feature Extraction, Construction, and Selection: A Data Mining Perspective*. Kluwer Academic, 1998
- J. E. Olson. *Data Quality: The Accuracy Dimension*. Morgan Kaufmann, 2003
- D. Pyle. Data Preparation for Data Mining. Morgan Kaufmann, 1999
- V. Raman and J. Hellerstein. Potters Wheel: An Interactive Framework for Data Cleaning and Transformation, VLDB’2001
- T. Redman. *Data Quality: The Field Guide*. Digital Press (Elsevier), 2001
- R. Wang, V. Storey, and C. Firth. A framework for analysis of data quality research. IEEE Trans. Knowledge and Data Engineering, 7:623-640, 1995