Data Mining Issues

Danushka Bollegala

UNIVERSITY OF LIVERPOOL

Missing values

- What if we do not know the value of a particular feature of an instance?
    - eg. the height of the 10-th student is missing although other feature values are known for the 10-th student and all the other students in the dataset
- May be it is important to know this missing value because the height is an essential feature for our classification task such as obesity.

Handling missing values

- Ignore the training instance completely
  - might get away because of redundancy in the dataset. We might have another student with the same height as student no 10.
  - might not get away because student no 10 was the only student who had obesity. By ignoring student no 10, we loose all the information we had about the positive class.

Fill in values by hand

- Re-annotate the data or re-measure the instances with missing feature values
  - Reliable method to overcome the missing value problem
  - Might not be possible in practice because we no longer have access to the subjects.
  - Too slow (manual work)
  - Costly (manual work)
  - There might be lots of data points with missing feature values

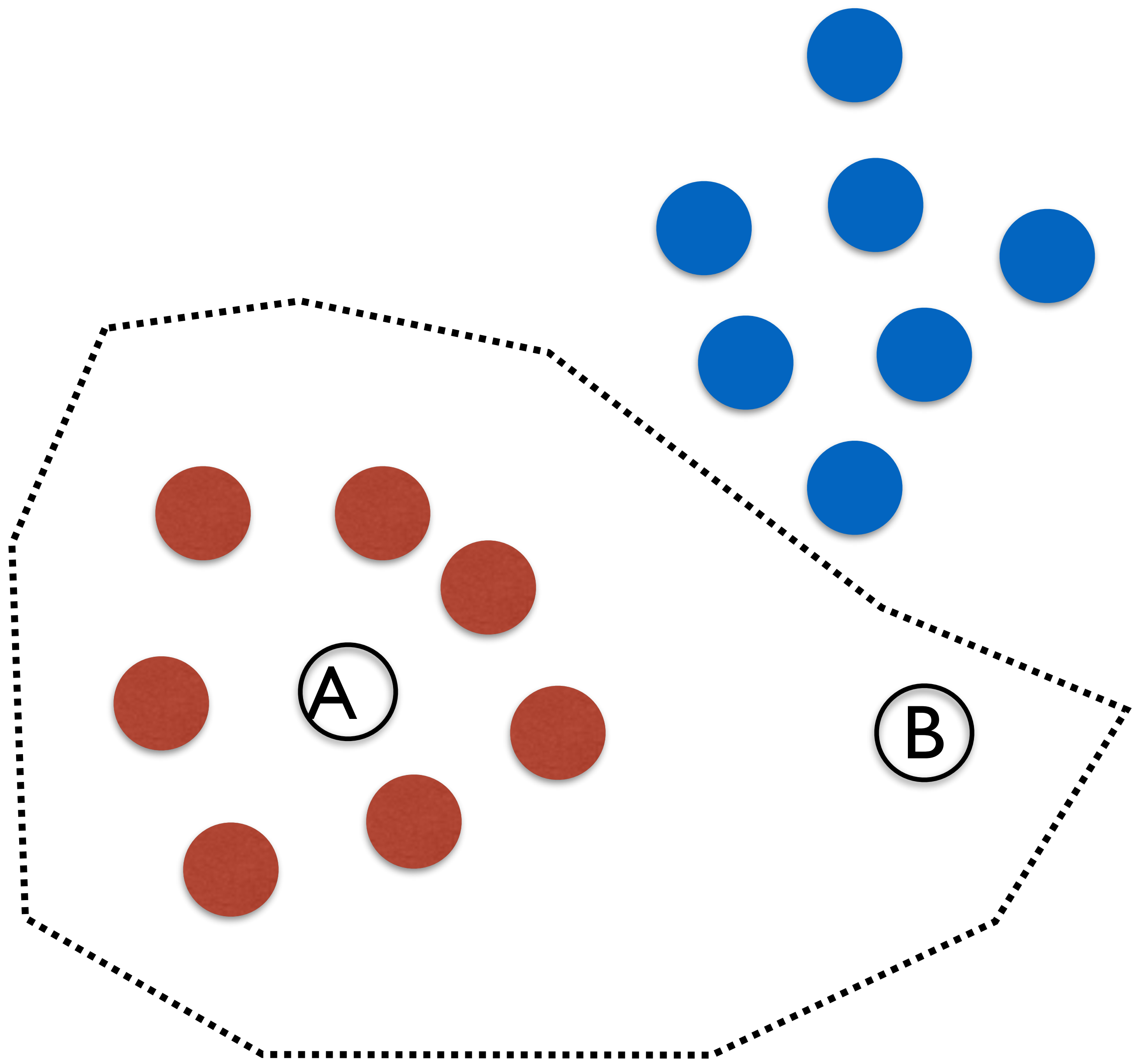
setting a “missingValue” constant

- We consider “missing” as another category for the feature and set some constant indicating that the value is missing such as “missingValue”.
- Not possible for numerical data. Does “0” mean the value was actually measured and turned out to be zero, or was it simply missing (possibly non-zero) in the dataset?
- Does not actually solve anything!

Replace with the mean

- Compute the mean of the available feature values for the entire training dataset and replace the missing values by this “sample” mean.
- This might be a good option if the data points with missing values are representative samples in the dataset.
- But if those data points are outliers this method is inaccurate

Outliers



It might be OK to say that point A is also red (the mean colour of the points around A) but it might not be accurate to say B is also red. B is an outlier belonging to neither red nor blue classes

Predicting missing values

- We can train a new classifier to first predict the missing values in data instances and then train a second classifier to predict the target class using all (original + missing values predicted) the data points.
- How is this possible?

Predicting missing values

s_1=(height=170, weight=60)

s_2=(height=169, weight=60.1)

s_3=(height=171, weight=59.9)

s_4=(height=150, weight=40)

s_5=(height=155, weight=50)

s_6=(height=160, weight=55)

s_7=(height=168, weight=?)

Guess the weight of s_7?

Accept missing values

- Just leave the data points with missing values as they are, and let the algorithm (eg. classifier) deal with the missing values in an appropriate manner
- The classifier might first try to come up with a rule to classify data without using features that have missing values. If it can do so with high accuracy, then we are fine. Nothing to worry about missing values.

Noisy values

- By “noisy data” we mean random errors scattered in the data
  - eg: due to inaccurate recording, data corruption
- Why is it a problem?
  - Overfitting:
    - If we assume the noisy values are correct values then we will learn classifiers to predict the noise as well. This could not be possible because it is “random” noise OR we might end up learning a classifier that learns “too much” from the train data and does not generalize to test data
      - This phenomenon is called “over-fitting” (fitting too much to the train data such that the model does not work well outside the give train dataset)
- Some noise will be very obvious:
  - data has incorrect type (string in numeric attribute)
  - data is very dissimilar to all other entries (10 in a feature otherwise in the range [0,1]
- Some incorrect values might not be obvious
  - eg. typing 0.52 instead of 0.25

Solutions to noisy values

- Manual inspection and removal
- Use clustering on the data to find instances or features that lie outside the main clusters (outlier detection) and remove them
- Use linear regression to determine the function, then remove those that lie from the predicted value
- Ignore all values that occur below a certain frequency threshold
    - effective for detecting misspellings in text
- If noisy data points can be identified and removed, we can apply missing value techniques to fill the missing features.

Data normalization/canonicalization

- The same name can refer to different entities (namesake disambiguation problem)
  - Jim Clark (netscape ceo vs. F1 champion)
  - I had a *mac* (Apple Mac vs. McDonald burger)
- The same entity can be referred to by different names
  - Will Smith (fresh prince)
- How to normalize/resolve the different names to the proper entity?
  - word sense disambiguation problem

Redundant values

- What if we have multiple copies of the same data point?
  - eg. crawled data from mirror site
- Some machine learning algorithms can get adversely affected by duplicate data
  - Let us assume that seeing the word "advert" in an email is a good indicator that it is spam. We would like to compute the probability P(spam|advert)
    - P(spam|advert) = x / y
    - x = no. of times we saw "advert" in spam emails
    - y = no of times we saw "advert" in an email
    - if we doubly count spam emails with the word "advert" then P(spam|advert) = 2x / (x + y)

Quiz

- Show that if we doubly count the spam emails that contain the word “advert” in our previous example, the probability P(spam|advert) becomes over-estimated.

Over-fitting vs. Under-fitting

- If a model M, trained on train data D(train) performs well on D(train), but poorly on a separate test dataset D(test), then it is likely that M is over-fitting to D(train)
- Typically you will see 90-99% accuracy on D(train) and 40-60% accuracy on D(test) in the case of binary classification on balanced (equal no. of positive and negative) datasets
- This is because M has more than required parameters that it can "fit" to D(train) (too much flexibility), and it fits all of those on D(train), generalizing poorly to D(test)
- Under-fitting is on the other hand is the situation where you get poor performance on D(train) because your model is not sufficiently "fitted" to the train data.

Solutions to under-fitting

- Learning has not converged
  - Let the training proceed for more iterations
- Your feature space is too small/inadequate
  - Implement more/better features
- Your train data is bad/noisy/missing values
  - Cleanse/re-annotate train data
- Your algorithm is not training well
  - select a different training algorithm

Solutions to Over-fitting

- Reduce the flexibility of your model
  - Regularization (we will discuss in detail when we learn logistic regression)
- Early stopping
  - Premature termination of training to prevent parameter overfitting
- Dropout
  - Randomly tune only half of your parameters on any given training instance
  - Known to be effective when training deep neural networks
  - We will discuss dropout in detail in our lecture on deep learning.