Prepping Data

Week	THURS. Week 1
Assignment Due	
Assignment Done	☐
Due Date	
Notes Done	☑

Presentation

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/3ad26548-f056-4fd7-aeb9-d06b801a345d/L2-Data.pdf

Class Notes

Data can come in many forms, but most algorithms require an ***observed data matrix***, X

- Each *row* of X pertains to an ***observation***.
- Each *column* pertains to info for the observations → known as a ***feature***
  - Can be categorized in one of three types:
    - **Continuous valued**
    - **Categorical-Nominal**
    - **Categorical-Ordinal**
- Depending on the algorithm being used, we may need to:
  - Convert categorical features into a set of binary features → **One-hot encoding**
  - “Standardize” features so that they have the same range → **Z-scoring**

        - A feature that has a wider range of values may have a larger influence on the overall distance of the feature values.

Standardizing Data

Ex: Imagine two features for a person (slide 6)

- Height
- Weight

Each of these features are on different scales:

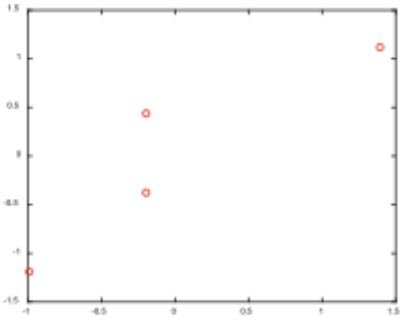
- Height (in.): Maybe 12 → 80
- Weight (lbs.): Maybe 5 → 400

If the data is used as-is, then one feature may have more influence than the other depending on the algorithm.

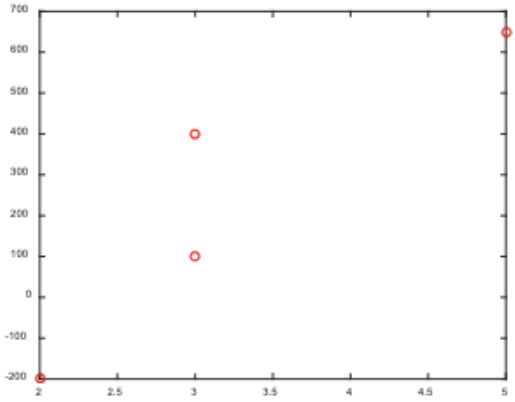
Standardized data has zero mean and unit deviation.

- **Zero mean** - observations for an entire column ends up with a mean of 0
- **Unit deviation** - each column has a standard deviation of 1
- We would do this standardization by treating each feature independently and:
    - Center it (subtract the mean from all samples, each row)
        - The mean is taken from the entire column (feature).
        - From here, the columns will now have a mean of 0.
    - Make them all have the same span (divide by standard deviation, each row)
        - Standard deviation would now be 1.
- This is referred to as standardizing the data or **zscoring** our data.
    - Now in comparing the data, there wouldn't be unequal influence by certain factors having a wider range and/or larger magnitude.

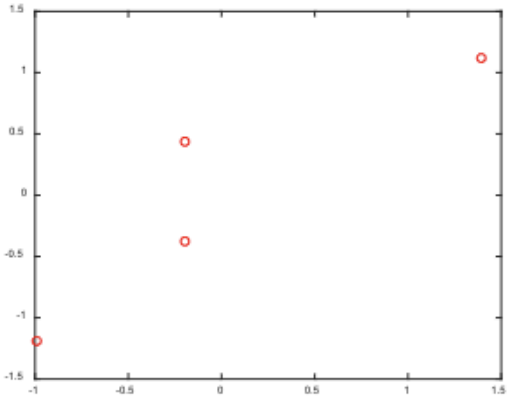
```
m = mean(X);   %get mean of each feature
s = std(X);    %get std of each feature
%m = 3.2500  237.5000
%s = 1.2583  368.2730

X = X - repmat(m,size(X,1),1);
X = X./repmat(s,size(X,1),1);
figure(2);
plot(X(:,1),X(:,2),'or');
```



Original



Standardized



- Shape that the distribution creates is the same, but the magnitude in which each observation is measured by is lessened to avoid some observations having large influence over the entire analysis.

Note on Standard Deviation

SD can be formulated in two diff ways:

- Difference is in averaging → averaging by n or by n-1
- If mean is ***sample mean*** from data itself, there is dependency on standard deviation and mean, leading to a degree of freedom being lost,
  - From here, we should be using the equation that subtracts 1 from the N value.

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)^2}$$

- If the mean is the ***population mean*** over all possible populations, then we can use the first version of the equation where averaging is done with N.
    - `numpy` uses this version by default, which is not convenient since we often do not know the population mean.
        - When using `np.std`, ensure to pass in the correct parameter for degrees of freedom to ensure the correct standard deviation is being calculated.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

In a **target value** matrix, we have one target value for each observation that we have for supervised data.

- We typically DO NOT manipulate our target data, unless its categorical and we need to enumerate it (one-hot encode).

Data Dimensionality

The variable D is typically selected as such since it refers to the dimensionality of our data:

- We could have a LOT of information about our data → Ex: millions of pixels for a photo, thousands of words in a written work
    - Based on this, if data sample X_i has D values associated with it, then we can write this as $X_i = (X_{i,j})_{j=1}^{D}$
- Having more observations is typically associated with better performance → more representation of different observations
    - It is not always the case, however.
- In having more observations, there are issues of:
    - **Computational cost**
        - Especially as we have more features in both time and space efficiency

    - **Statistical cost**
        - We can train a system to be precise, but this specificity can be negative in being trained on certain data that may not generalize to data that it is never worked with before
    - **Visualization**
        - How can we look at the data to understand its structure?
- Ex: Imagine we only have one dimension.
    - Ideally, we'd have a sample for every single possible location in this space.
        - If we have a discrete space, and the range is [0,9], then we'd only need 10 samples to have complete coverage.
    - Then, if we had two features, to get the same coverage, we'd need $10^2 = 100$ samples.
    - ***As the number of features increases, the more sparse our coverage of the hypothesis space will be.***
        - In this high dimensional space, we have a finite number of observations we train with and can train a system with what's in the hypothesis space.
            - However, it may do poorly on the hypothesis space that we did not train on.

Based on this problem, we may want to reduce the number of features that we have:

- We DO NOT want to discard features haphazardly → instead want to reduce dimensionality in an intelligent manner

Dimensionality Reduction

Goal: Represent data with fewer variables

- Want to preserve as much structure in the data as possible
    - If there is class information, we can ensure that when we are manipulating features that we are doing so in a way that is helping us and not hurting us.
- Benefits:
    - Need less data to cover the feature space

    - Easier learning - fewer parameters to learn
        - Will allow generalization to be better
    - Easier visualization - hard to visualize more than 3D or 4D
- Approaches to dimensionality reduction:
    - **Feature selection** - will start today and wrap up next Tuesday
        - Involves selecting a subset of original features
        - If class info exists, we want to choose features that'll get us best to our goal.
    - **Feature construction** - may be covered Tuesday and into Thursday
        - Takes existing features and constructs new features from them
        - Idea: Hopefully, majority of relevant info is encoded in the few features that we've created