CS 383 – Machine Learning

Dealing w/ Data

Part 1

Slides adapted from material created by E. Alpaydin
Prof. Mordohai, Prof. Greenstadt, Pattern Classification (2nd Ed.),
Pattern Recognition and Machine Learning

Objectives

- Standardizing Features
- Understand the Curse of Dimensionality
- Perform *Feature Selection* based on statistics

Additional Resources

- http://people.cs.pitt.edu/~milos/courses/cs3750-Fall2011/lectures/class16.pdf

Parsing Text Files

- The first thing we'll need to do in this class is get data!
- We'll get data from text files
- But unfortunately they may be in all different formats ☹
  - If we're lucky it may be in .csv format and we can use the `csvread` function
  - But even then there may be mixed data types making it impossible.
- So let's look at how to parse a general text file
  - Plus this will allow us to look at some Matlab!

Parsing Text Files

- Parsing files takes a while
  - And we might need to do this over and over again while we test/build our system.
- So often we want to just parse it once, save its data in an easy to use format, then next time load it.
- Let's get started

```
clear all; %remove all the old variables in the workspace
close all; %closes all open figures

filename = 'toy.data';  %the file to read
datafile = 'toy.mat';
if(exist(datafile,'file'))
    load(datafile);
else
```

Parsing Text File

- Else we need to parse the file.
- First we need to try to open it.

```
fid = fopen(filename);
if(fid<0)
    display('File not found');
    return;
end
```

Parsing Text File

- We opened the file and saw
  - The first line is just header info, so we want to skip it
  - Then the data is in the format
    - “[integer]” [double] [double] “[integer]”
- So we can use regular expressions to help us get this data!

```
fgetl(fid);%remove the first line
x=[];
y=[];
while(~feof(fid))
    line = fgetl(fid);
    C = textscan(line,'"%d" %f %f "%d"');
    x(end+1, :) = [C{2}, C{3}];
    y(end+1,1) = C{4};
end
```

Parsing Text File

- Finally let's close the file and save the data so next time we don't need to re-parse

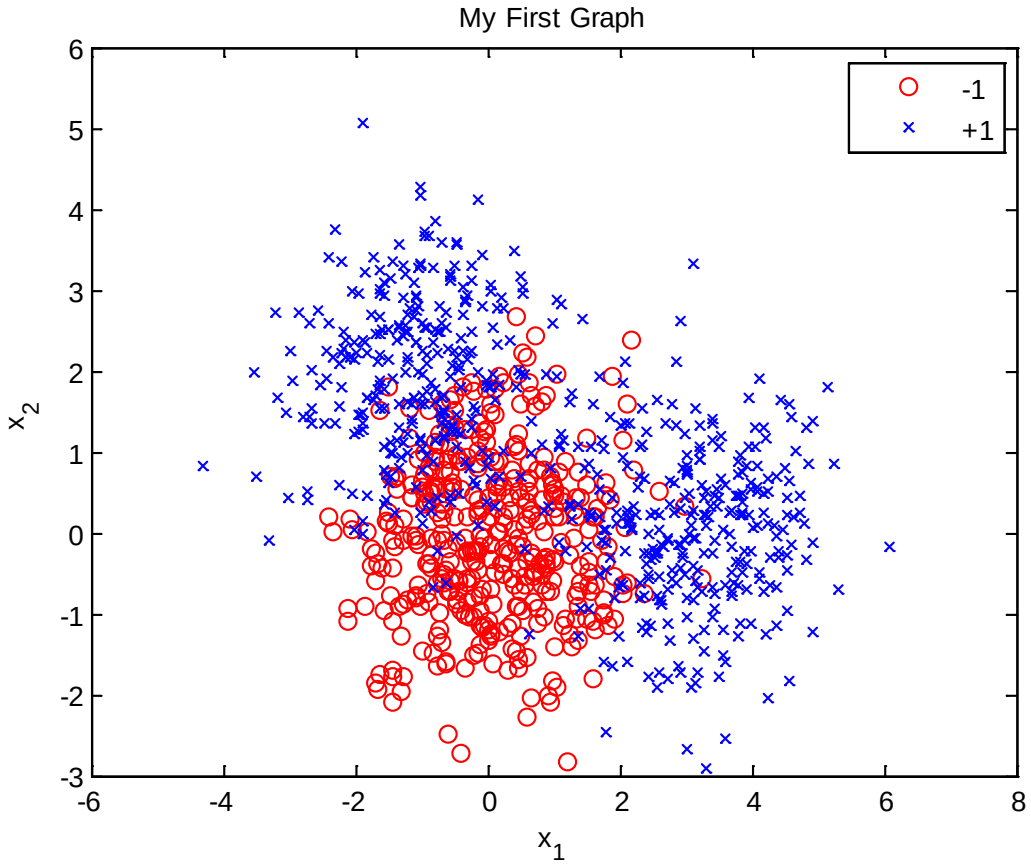
```
    fclose(fid);  %don't forget to close your file!
    save(datafile,'x','y');
end %end of if(exist.... else...
```

Parsing Text File

- Finally it would cool to see the plot of this data

```
figure(1);
plot(x(y==-1,1), x(y==-1,2),'or');
hold on;
plot(x(y==1,1), x(y==1,2),'xb');
hold off;
title('My First Graph');
xlabel('x_1');
ylabel('x_2');
legend('-1', '+1');
```

Parsing Text File



Standardizing Data

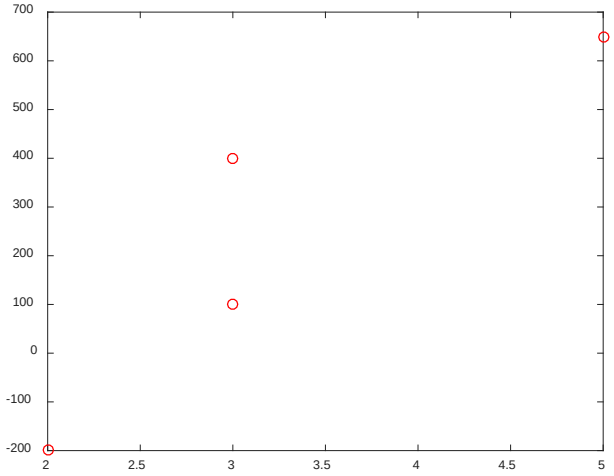
Standardizing Data

- Before we start using data we typically want to *standardize* non-categorical features (this is sometimes referred to as *normalizing* them).
- Imagine two features for a person
  - Height
  - Weight
- These are both on very different scales
  - Height (inches): Maybe 12 → 80
  - Weight (lbs): Maybe 5→400
- If we used the data as-is, then one feature may have more influence than the other

Standardizing Data

- Standardized data has
  - Zero mean
  - Unit deviation
- We treat each feature independently and
  1. Center it (subtract the mean from all samples)
  2. Make them all have the same span (divide by standard deviation).
- Example

```
X = [3, 400;
     2, -200;
     3, 100;
     5, 650];
figure(1);
plot(X(:,1),X(:,2),'or');
```

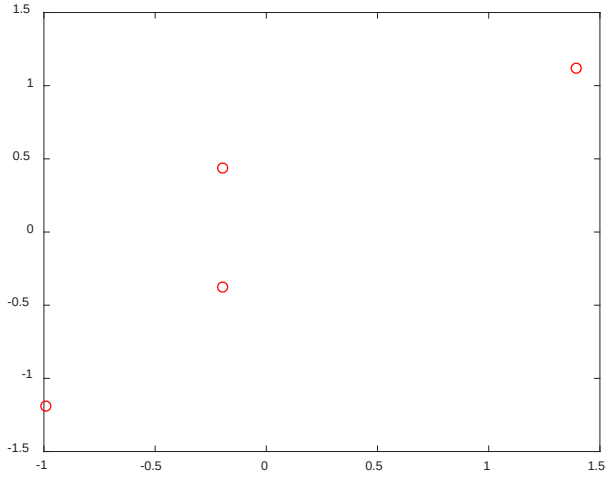


Standardizing Data

- We can compute the mean and standard deviation of each feature easily and then subtract the means from each observation and divide each (centered) observation by the standard deviation of each feature.
- Would it make sense to do this with categorical data?

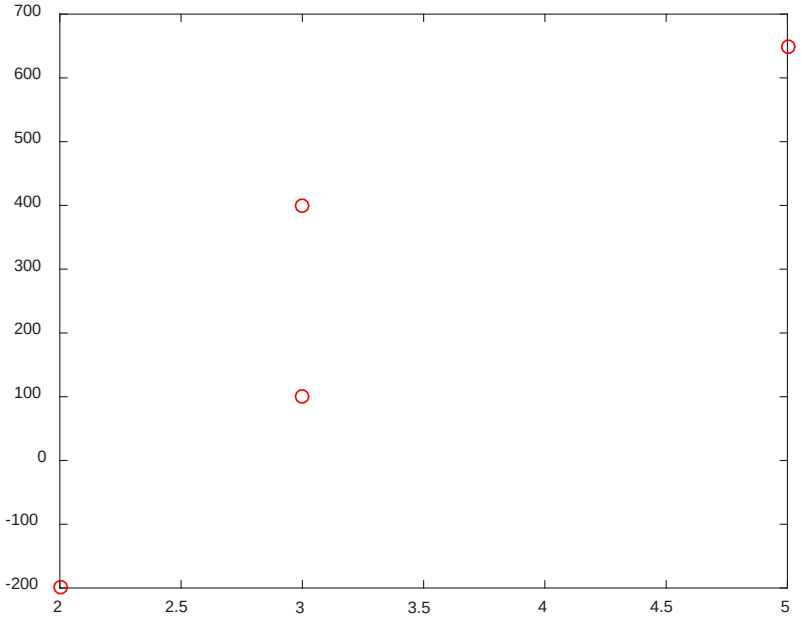
```
m = mean(X);  %get mean of each feature
s = std(X);   %get std of each feature
%m = 3.2500  237.5000
%s = 1.2583  368.2730

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

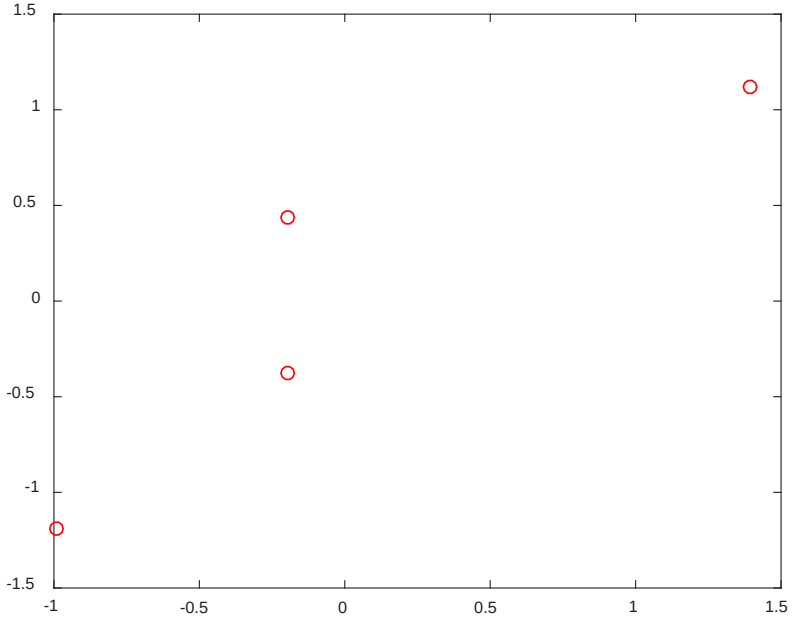


Standardizing Data

Original



Standardized



Data Dimensionality

And the “curse”

Data Dimensionality

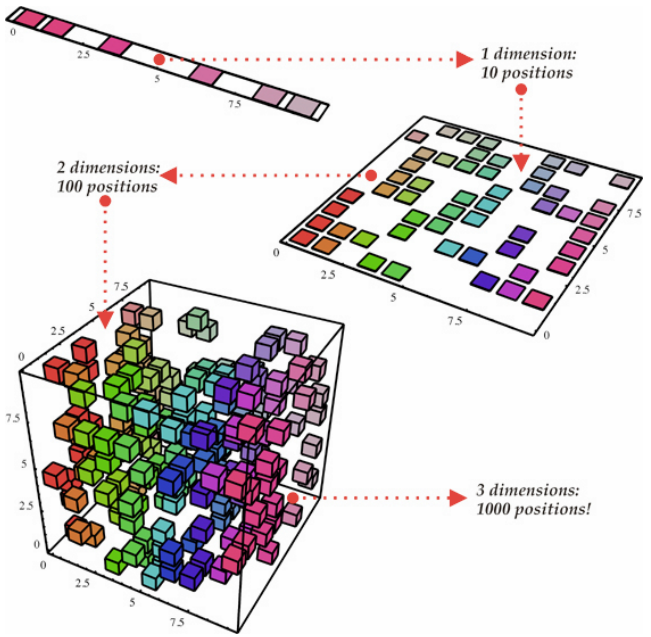
- Typically our data has a lot of information in it
  - An image has millions of pixels
  - A textbook has thousands of words
- We call the amount of information we have for a given data sample, it's *dimension*
- Therefore if data sample X_i has *D* values associated with it (which we call **features**) then we can write this as $X_i = \left(X_{i,j}\right)_{j=1}^{D}$ and call *D* the *dimensionality* of X_i

Data Dimensionality

- Can there be drawbacks of too much information (too high of dimensionality)?
- Computation cost
  - Both time and space efficiency
- Statistical Cost
  - Too specific? Doesn't generalize enough?
  - Too susceptible to noise?
- Visualization
  - How can we look at the data to understand its structure?

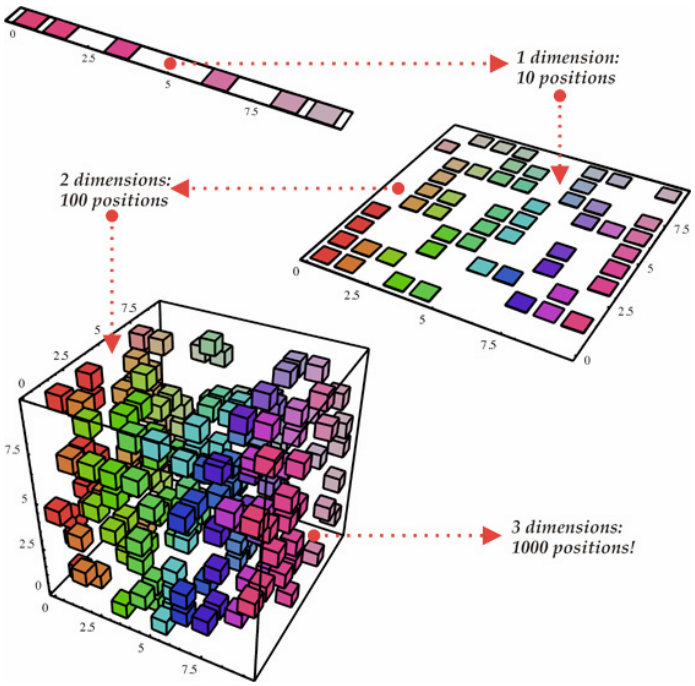
Curse of Dimensionality

- Imagine we only have one dimension
- Ideally we'd have a sample for every single possible location in this space.
- If we have discrete space, and the range is [0, 9], then we'd just need 10 samples to have complete coverage
  - Great. No big deal!



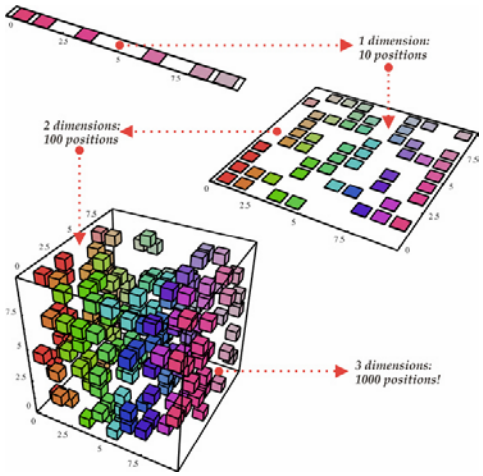
Curse of Dimensionality

- How about if we had two features ($D = 2$)?
- To have the same coverage we'd need 10^2=100 samples



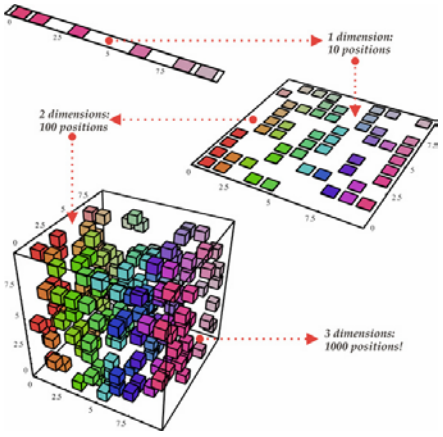
Curse of Dimensionality

- Ok. But in the real world our data ends up having D be really large
  - If we're trying to classify an image, each pixel is a feature, and thus $D = millions$
  - And therefore to have complete coverage we'd need $10^{millions}$ samples
  - Impossible!



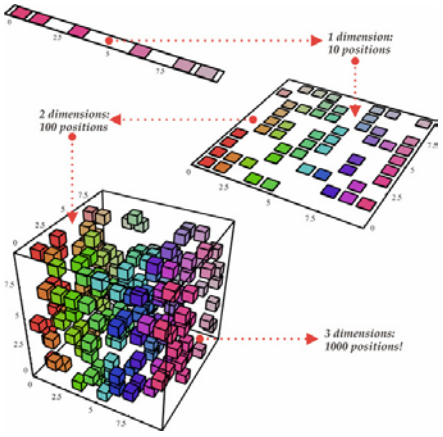
Curse of Dimensionality

- This is the *curse of dimensionality*
  - In higher dimension space we need exponentially more samples for equivalent coverage.
- Therefore if our data is in high dimensional space it will likely sparsely cover that space
  - And instances will be far apart from one another



Curse of Dimensionality

- This might be one motivation for *dimensionality reduction*
  - Going from higher dimension data to lower
- However we want to do this "intelligently"
  - We don't want to lose much important information by doing this!



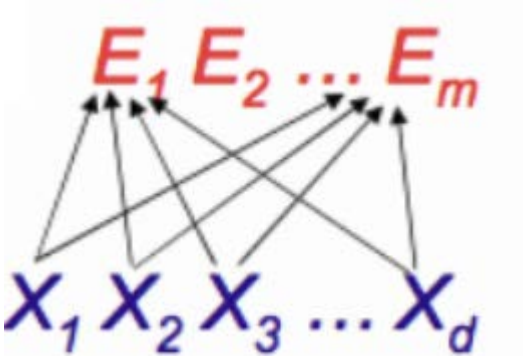
Dimensionality Reduction

Dimensionality Reduction

- Goal: Represent instances with fewer variables
  - Try to preserve as much structure in the data as possible
  - If there is class information
    - Discriminative: increase class separability
- Benefits:
  - Need less data to cover the feature space
  - Easier learning – fewer parameters to learn
  - Easier visualization – hard to visualize more than 3D or 4D

Dimensionality Reduction

- Approaches
  - Feature selection
    - Pick a subset of the original dimensions
    - If there is class information
      - Discriminative: Pick good class predictors
  - Feature construction
    - Construct a new set of features from existing
    - In particular we'll look at feature projection



Feature Selection

Feature Selection

- Give set of features, some features are more important than others
- We want to select some subset of features to be used by learning algorithms
  - Score each feature
  - Select set of features with best score

Greedy Forward Features Selection

- Greedy heuristic:
  - Start from empty set of features $F = \emptyset$
  - For iteration t, for each remaining feature j
    - Run learning algorithm for features $\mathrm{F} \cup j$
  - Select next best feature $\hat{j}$
    - $F = F \cup \{\hat{j}\}$
  - Continue until converge on locally optimal result
- You could also do backwards feature selection
  - Start with all features, remove worst, etc..

Separability Feature Selection

- Running that many learning algorithms seems like a lot!
- It is!
  - If we have 1,000,000 pixels and do a top up approach, then the first time around we must train 1,000,000 system
    - Then the next time we must train 999,999 system
    - Up to N! systems!
- Could we somehow do this without relying on experimentation?
- If our task is classification maybe we can find a way to measure a feature's effect on class separability…

Feature Selection by Information Gain

- Given probability of events $v_1, \dots, v_n$ as $P(v_1), \dots, P(v_n)$ we can compute the ***entropy*** as

$$H\big(P(v_1), \dots, P(v_n)\big) = \sum_{i=1}^{n} (-P(v_i) \log_2 P(v_i))$$

- Entropy measure the randomness of the data
- Example: Tossing a fair coin
  - v_1=heads, v_2=tails,
  - P(v_1) =0.5, P(v_2) =0.5
  - $H\left(\frac{1}{2}, \frac{1}{2}\right) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$

Feature Selection via Information Gain

- Let us assume we have a dataset with binary labels:
$$Y_i \in \{0,1\}$$
- And that all features are discrete.
- Let a chosen feature $j \in \{1, \dots, D\}$ have *k* possible values (thus a discretized feature).
- Then the dataset $X = \{X_i\}_{i=1}^{N}$ can be split into subsets $\{E_1, \dots, E_k\}$ according to each observation's value, $X_{i,j}$
- The intuition is to compute the original entropy of the system and then the average entropy on this split
  - We want to maximize this difference
  - This is our "gain"

Feature Selection via Information Gain

- Let $p = \#\{1\}$,be the number of samples with label one over the entire dataset
- Let $n = \#\{0\}$ be the number of samples with label zero over the entire dataset
- Finally let p_i, n_i be the number samples in subset E_i, with label one and zero, respectively.
- We then define the remainder with respect to *A* as:

$$remainder(A) = \sum_{i=1}^{k} \frac{p_i + n_i}{p + n} H\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

  - This is basically the weighted average entropy

Feature Selection via Information Gain

- We can now compute information Gain (IG), or reduction in entropy, that would occur if we split on this attribute/feature:

$$IG(A) = H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - remainder(A)$$

  - $remainder(A)$ is the remaining uncertainty after splitting on the attribute
  - From it's equation we can consider it the weighted average entropy after splitting based on A
- We should choose the attribute with the largest IG!
  - I.e. the one with the smallest remainder.

Example

- For reference:
  - $H\big(P(v_1), \ldots, P(v_n)\big) = \sum_{i=1}^{n}(-P(v_i)\log_2 P(v_i))$
  - $remainder(A) = \sum_{i=1}^{k} \frac{p_i+n_i}{p+n} H\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$
- Class 1 Samples
  - {(1,1)(1,3),(2,2)}
- Class 0 Samples
  - {(1,2),(3,2),(2,2)}
- Let's figure out which feature provides the highest information gain!

Example

- $H\big(P(v_1), \dots, P(v_n)\big) = \sum_{i=1}^{n}(-P(v_i)\log_2 P(v_i))$
- $remainder(A) = \sum_{i=1}^{k} \frac{p_i+n_i}{p+n} H\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$
- Positive Samples
  - {(1,1)(1,3),(2,2)}
- Negative Samples
  - {(1,2),(3,2),(2,2)}
- Feature 1
  - p_1=2, n_1=1
  - p_2=1, n_2=1
  - p_3=0, n_3=1
  - Remainder(1)=
    (2+1)/(3+3)*(-2/3*$\log_2$(2/3)+-1/3*$\log_2$(1/3))
    + (1+1)/(3+3)*(-1/2*log(1/2)+-1/2log(1/2)
    + (0+1)/(3+3)*(0log(0)-1log(1)) = 0.7925
  - IG(1)=(-3/6log(3/6)-3/6log(3/6))-0.7925 = 0.2075

Example

- Positive Samples
  - {(1,1)(1,3),(2,2)}
- Negative Samples
  - {(1,2),(3,2),(2,2)}
- Feature 2
  - p_1=1, n_1=0
  - p_2=1, n_2=3
  - p_3=1, n_3=0
  - Remainder(2)=
    (1+0)/(3+3)*(-1/1*$\log_2$(1/1)+-0*$\log_2$(0))
    + (1+3)/(3+3)*(-1/4*log(1/4)+-3/4log(3/4)
    + (1+0)/(3+3)*(-1/1log(1/2)-0/1log(0/1)) = 0.5409
  - IG(2)=(-3/6log(3/6)-3/6log(3/6))-0.5409 = 0.4591
- Recall IG(1)=0.2075
- So we should prioritized feature 2!

IG for real-valued features

- The examples we showed work for categorical and/or enumerated features
- This could be expanded to work on real-valued features
- While perhaps not the most sophisticated approach, you could break up the range of possible values K intervals and assign values to enumerated "bins"
- I'll let you think about or research more intelligent approaches…