Lecture 5: Normalization, Making Plots in Python

CS 167: Machine Learning

Potential Problem with k-NN

For simplicity, assume we're just looking at the **age** and **sex** columns in the `titanic` data set.

	sex	age
example 1	1	50
example 2	0	48

distance: $\sqrt{(1-0)^2 + (50-48)^2} \approx 2.24$

	sex	age
example 1	1	50
example 3	1	25

distance: $\sqrt{(1-1)^2 + (50-25)^2} = 25$

What's the problem?

Normalization

Normalizing data: rescale attribute values so they're about the same

A simple method: replace each value with proportion relative to the max value.

Example: The oldest person in the titanic data set is 80, so

age	**replaced by**
80	$80/80 = 1$
50	$50/80 = 0.625$
48	$48/80 = 0.6$
25	$25/80 = 0.3125$
4	$4/80 = 0.05$

After Normalizing

	sex	age
example 1	1	0.625
example 2	0	0.6

distance: $\sqrt{(1-0)^2 + (0.625-0.6)^2} \approx 1.0003$

	sex	age
example 1	1	0.625
example 3	1	0.3125

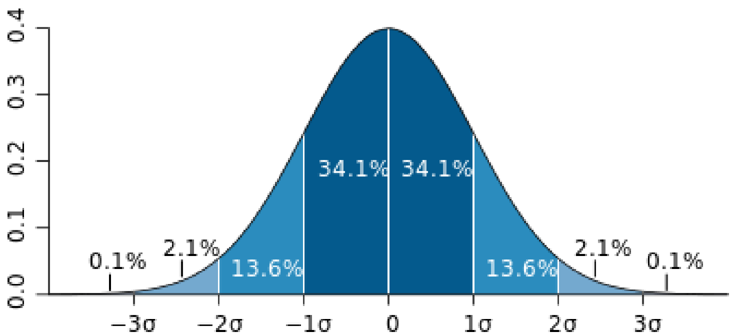
distance: $\sqrt{(1-1)^2 + (0.625-0.3125)^2} = 0.3125$

Now is sex over-emphasized?

Z-Score: Another Normalization Method

Idea: rather than normalize to proportion of max, normalize based on how many **standard deviations** they are away from the **mean**

Standard Deviation: usually represented as σ, a kind of "average" distance from the average value



Computing Standard Deviation

Let μ be the mean, then standard deviation of $x_1, x_2, \ldots, x_N$ is

$$\sigma = \sqrt{\frac{(x_1-\mu)^2 + (x_2-\mu)^2 + \cdots + (x_N-\mu)^2}{N}}$$

```
import math
data_points = [2,4,4,4,5,5,7,9]
total_sum = 0.0
for n in data_points:
    total_sum = total_sum + n
mean = total_sum/len(data_points)
print(mean)

tot_square_diff = 0.0
for n in data_points:
    tot_square_diff = tot_square_diff + (mean-n)**2
stdev = math.sqrt(tot_square_diff/len(data_points))
print(stdev)
```

```
5.0
2.0
```


Computing Z-Score

Then, to normalize, replace each value x_i with it's Z-Score based on the mean (μ) and standard deviation (σ) of its column.

Z-Score:

$$\frac{x_i - \mu}{\sigma}$$

Back to our example...

on the Titanic
sex mean (0: male, 1: female): 0.35
sex standard deviation: 0.48
age mean: 29.7
age standard deviation: 14.5

Z-Score for male: $(0 - 0.35)/0.48 \approx -0.73$
Z-Score for female: $(1 - 0.35)/0.48 \approx 1.35$

Z-Score for age 50: $(50 - 29.7)/14.5 \approx 1.4$
Z-Score for age 48: $(48 - 29.7)/14.5 \approx 1.26$
Z-Score for age 25: $(25 - 29.7)/14.5 \approx -0.32$

Back to our example...

	sex	age
example 1	1.35	1.4
example 2	-0.73	1.26

distance:

$$\sqrt{(1.35 - -0.73)^2 + (1.4 - 1.26)^2}$$
$$\approx 2.08$$

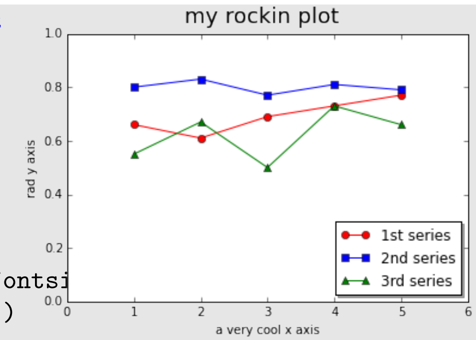
	sex	age
example 1	1.35	1.4
example 3	1.35	-0.32

distance:

$$\sqrt{(1.35 - 1.35)^2 + (1.4 - -0.32)^2}$$
$$= 1.72$$

Does this seem any better?

Plotting your data in Python




```
import matplotlib.pyplot as plt
%matplotlib inline

xvals = [1,2,3,4,5]
series1 = [.66,.61,.69,.73,.77]
series2 = [.8,.83,.77,.81,.79]
series3 = [.55,.67,.5,.73,.66]

plt.suptitle('my rockin plot',fontsi
plt.xlabel('a very cool x axis')
plt.ylabel('rad y axis')
plt.plot(xvals,series1,'ro-',label='1st series')
plt.plot(xvals,series2,'bs-',label='2nd series')
plt.plot(xvals,series3,'g^-',label='3rd series')

plt.legend(loc='lower right', shadow=True)

plt.axis([0, 6, 0, 1])
plt.show()
```


Titanic Exercises

Exercise 14: Normalize the columns used for predictions.

Then... after you get k-NN working

Exercise 15: Create an experiment in which you use several different k, and plot the accuracy on your test set. Include at least two different series of data in which you vary something about the experiment. Ideas:

- k-NN vs. weighted k-NN (make sure to include some really big k)
- normalized vs. non-normalized
- compare different normalization methods
- come up with an auto-generated axis-stretching algorithm and compare with base version