CSE419 – Artificial Intelligence and Machine Learning 2018

PhD Furkan Gözükara, Toros University

https://github.com/FurkanGozukara/CSE419_2018

Lecture 6

Feature Selection

Based on Asst. Prof. Dr. David Kauchak (Pomona College) Lecture Slides

Features

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

Where do they come from?

UCI Machine Learning Repository

http://archive.ics.uci.edu/ml/datasets.html

Provided features

Predicting the age of abalone from physical measurements

Name / Data Type / Measurement Unit / Description

Sex / nominal / -- / M, F, and I (infant)

Length / continuous / mm / Longest shell measurement

Diameter / continuous / mm / perpendicular to length

Height / continuous / mm / with meat in shell

Whole weight / continuous / grams / whole abalone

Shucked weight / continuous / grams / weight of meat

Viscera weight / continuous / grams / gut weight (after bleeding)

Shell weight / continuous / grams / after being dried

Rings / integer / -- / +1.5 gives the age in years

Provided features

Predicting breast cancer recurrence

1. Class: no-recurrence-events, recurrence-events
2. age: 10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, 90-99.
3. menopause: lt40, ge40, premeno.
4. tumor-size: 0-4, 5-9, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59.
5. inv-nodes: 0-2, 3-5, 6-8, 9-11, 12-14, 15-17, 18-20, 21-23, 24-26, 27-29, 30-32, 33-35, 36-39.
6. node-caps: yes, no.
7. deg-malig: 1, 2, 3.
8. breast: left, right.
9. breast-quad: left-up, left-low, right-up, right-low, central.
10. irradiated: yes, no.

Provided features

In many physical domains (e.g. biology, medicine, chemistry, engineering, etc.)

- the data has been collected and the *relevant* features identified
- we cannot collect more features from the examples (at least “core” features)

In these domains, we can often just use the provided features

Raw data vs. features

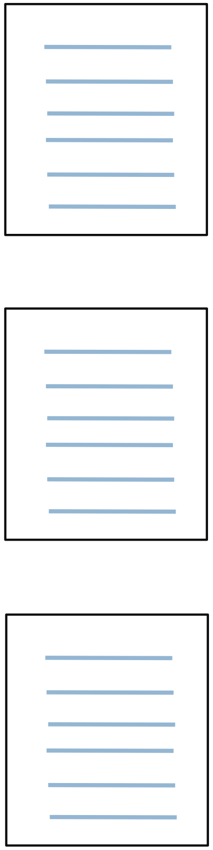
In many other domains, we are provided with the raw data, but must extract/identify features

For example

- image data
- text data
- audio data
- log data
- …

Text: raw data

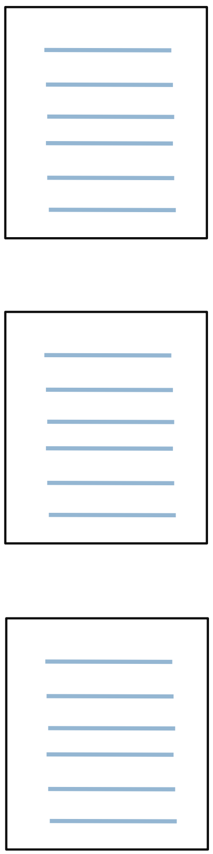
Raw data



Features?

Feature examples

Raw data



Features

Clinton said banana repeatedly last week on tv, “banana, banana, banana”

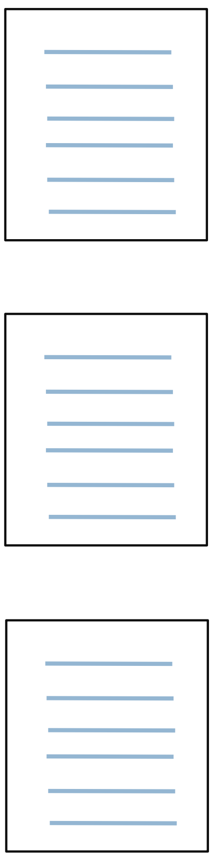
(1, 1, 1, 0, 0, 1, 0, 0, ...)

banana clinton said california across tv wrong capital

Occurrence of words

Feature examples

Raw data



Features

Clinton said banana repeatedly last week on tv, “banana, banana, banana”

(4, 1, 1, 0, 0, 1, 0, 0, ...)

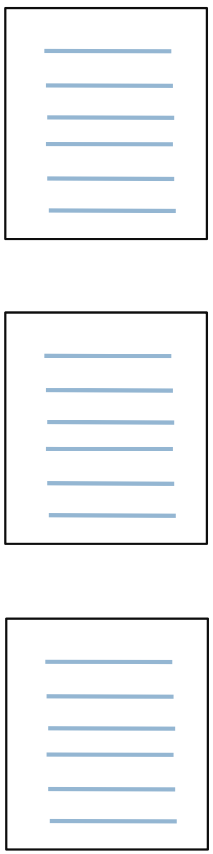
banana clinton said california across tv wrong capital

Frequency of word occurrence

Do we retain all the information in the original document?

Feature examples

Raw data



Features

Clinton said banana repeatedly last week on tv, "banana, banana, banana"

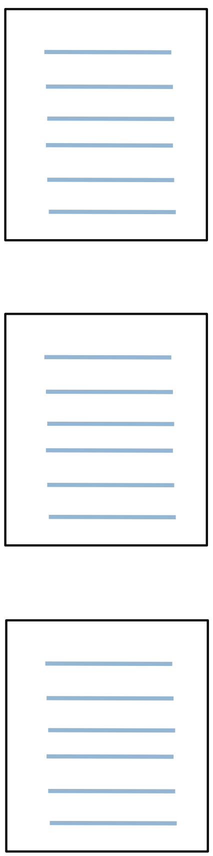
(1, 1, 1, 0, 0, 1, 0, 0, ...)

banana repeatedly
clinton said
said banana
california schools
across the
tv banana
wrong way
capital city

Occurrence of bigrams

Feature examples

Raw data



Features

Clinton said banana repeatedly last week on tv, “banana, banana, banana”

(1, 1, 1, 0, 0, 1, 0, 0, ...)

- banana repeatedly
- clinton said
- said banana
- california schools
- across the
- tv banana
- wrong way
- capital city

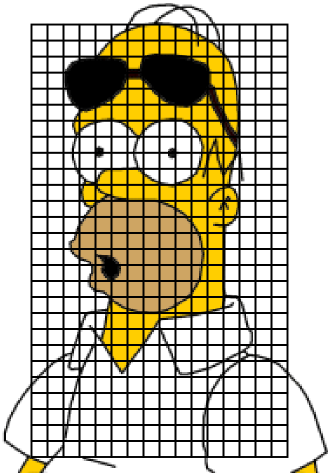
Other features?

Lots of other features

- POS (Part-of-speech): occurrence, counts, sequence
- Constituents
- Whether ‘V1agra’ occurred 15 times
- Whether ‘banana’ occurred more times than ‘apple’
- If the document has a number in it
- …
- Features are very important, but we’re going to focus on the models today

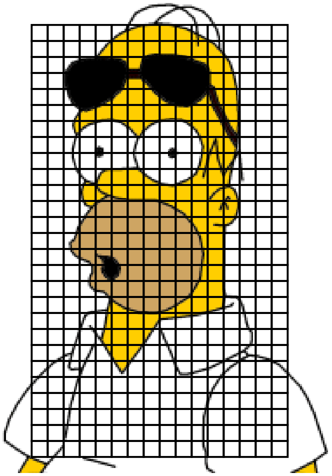
How is an image represented?

How is an image represented?



- images are made up of pixels
- for a color image, each pixel corresponds to an RGB value (i.e. three numbers)

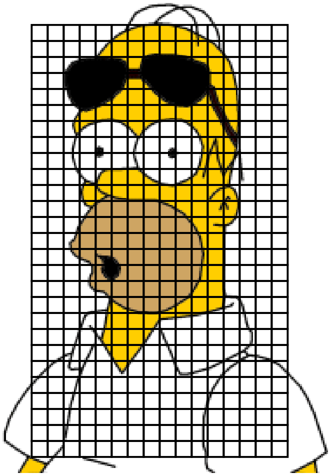
Image features

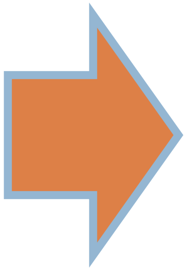


for each pixel: R[0-255]
G[0-255]
B[0-255]

Do we retain all the information in the original document?

Image features



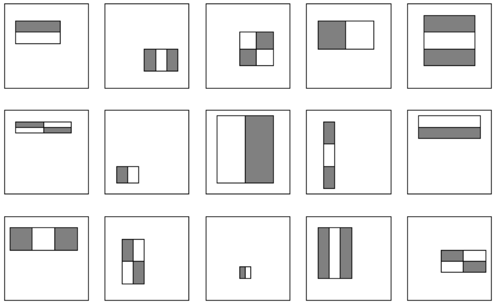


for each pixel: R[0-255]
G[0-255]
B[0-255]

Other features for images?

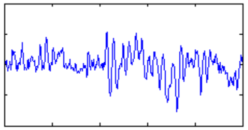
Lots of image features

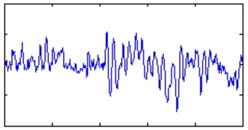
- Use “patches” rather than pixels (sort of like “bigrams” for text)
- Different color representations (i.e. L*A*B*)
- Texture features, i.e. responses to filters



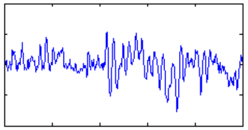
- Shape features
- …

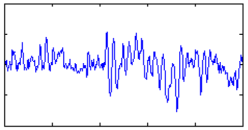
Audio: raw data

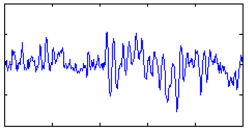




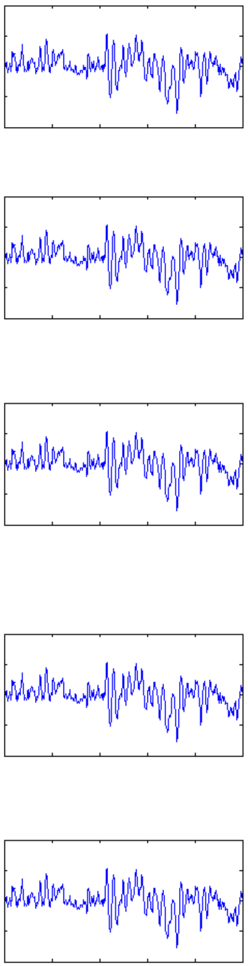
How is audio data stored?

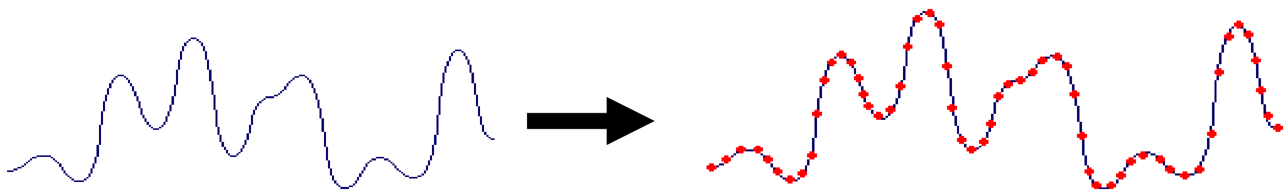






Audio: raw data



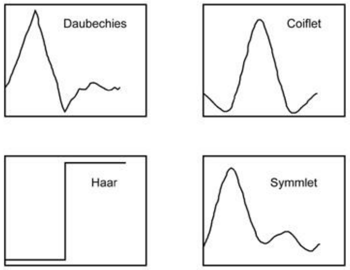


Many different file formats, but some notion of the frequency over time

Audio features?

Audio features

- frequencies represented in the data (FFT)
- frequencies over time (STFT)/responses to wave patterns (wavelets)



- beat
- timber
- energy
- zero crossings
- …

Obtaining features

Very often requires some domain knowledge

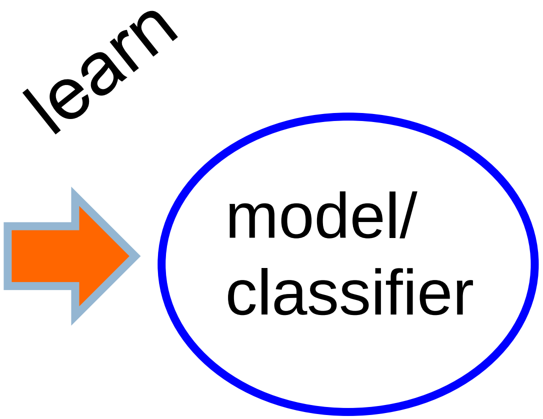
As ML algorithm developers, we often have to trust the “experts” to identify and extract reasonable features

That said, it can be helpful to understand where the features are coming from

Current learning model

training data
(labeled examples)

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES



Pre-process training data

training data
(labeled examples)

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

pre-process data

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

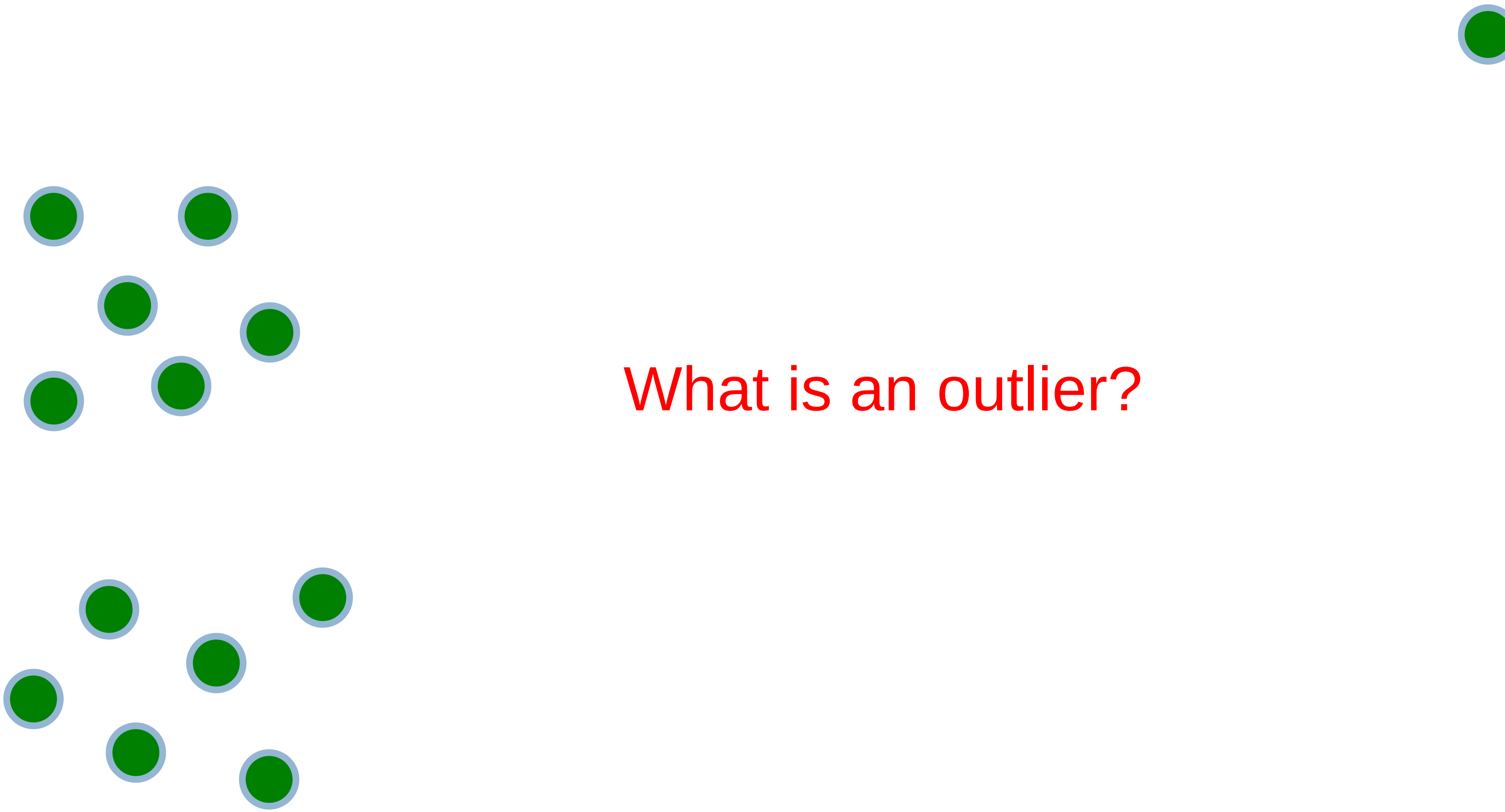
"better" training data

learn

model/
classifier

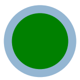
What types of preprocessing might we want to do?

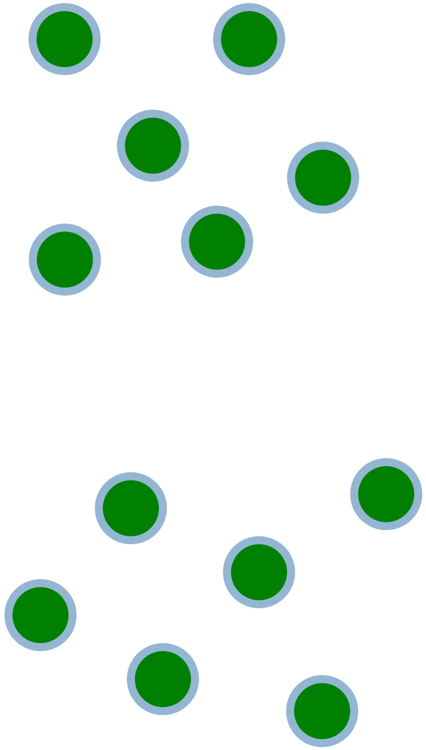
Outlier detection



What is an outlier?

Outlier detection

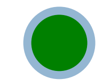


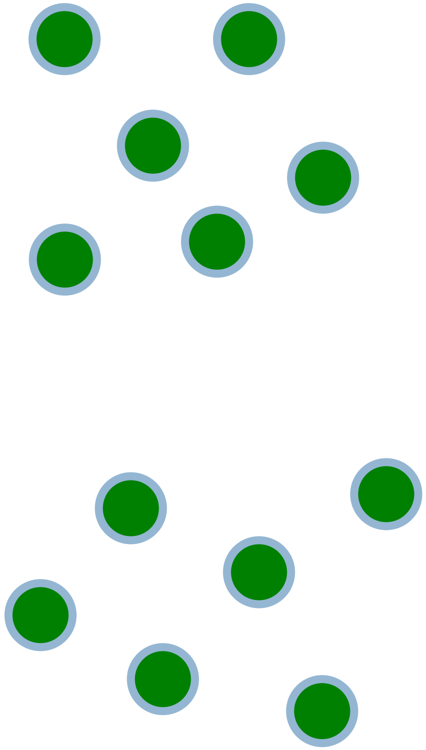


An example that is inconsistent with the other examples

What types of inconsistencies?

Outlier detection

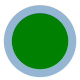


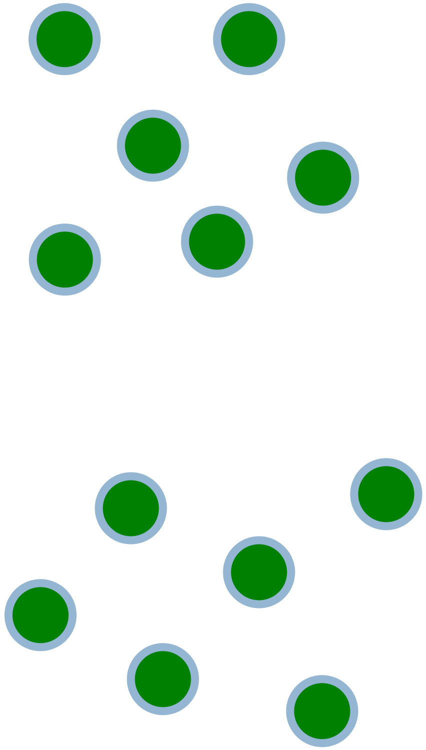


An example that is inconsistent with the other examples

- extreme feature values in one or more dimensions
- examples with the same feature values but different labels

Outlier detection





An example that is inconsistent with the other examples

- extreme feature values in one or more dimensions
- examples with the same feature values but different labels

Fix?

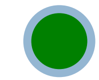
Removing conflicting examples

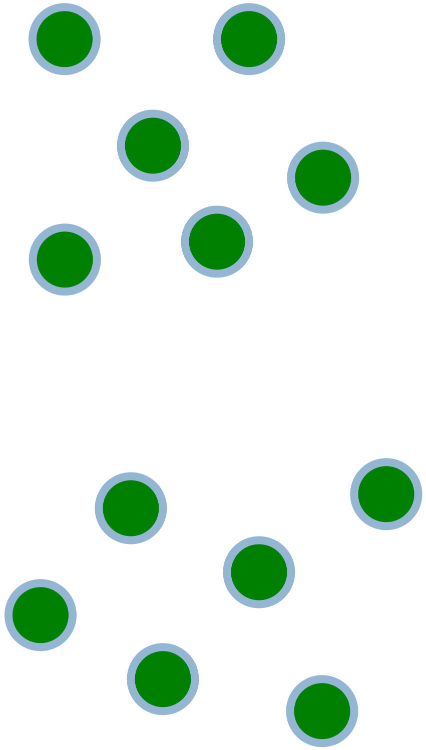
Identify examples that have the same features, but differing values

- For some learning algorithms, this can cause issues (for example, not converging)
- In general, unsatisfying from a learning perspective

Can be a bit expensive computationally (examining all pairs), though faster approaches are available

Outlier detection





An example that is inconsistent with the other examples

- extreme feature values in one or more dimensions
- examples with the same feature values but different labels

How do we identify these?

Removing extreme outliers

Throw out examples that have extreme values in one dimension

Throw out examples that are very far away from any other example

Train a probabilistic model on the data and throw out “very unlikely” examples

This is an entire field of study by itself! Often called outlier or anomaly detection.

Quick statistics recap

What are the mean, standard deviation, and variance of data?

Quick statistics recap

mean: average value, often written as μ

variance: a measure of how much variation there is in the data. Calculated as:

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

standard deviation: square root of the variance (written as σ)

How can these help us with outliers?

Variance Example

Variance Example: Try this!

- 3, 4, 5, 5, 6, 7, 9 (stress ratings); M= 5. 57
- What is the variance?
- $\Sigma(X - M)^2 / N$ is Variance
- Hint: Step 1: 3 - 5.57 (-2.57) and square it (6.6047), 4 - 5.57 (1.57) and square it (2.4645), 5-5.57 and square it, etc.
- Step 2: Add each together 6.6047 + 2.4645...
- Divide by 7, the Total Number of Scores...

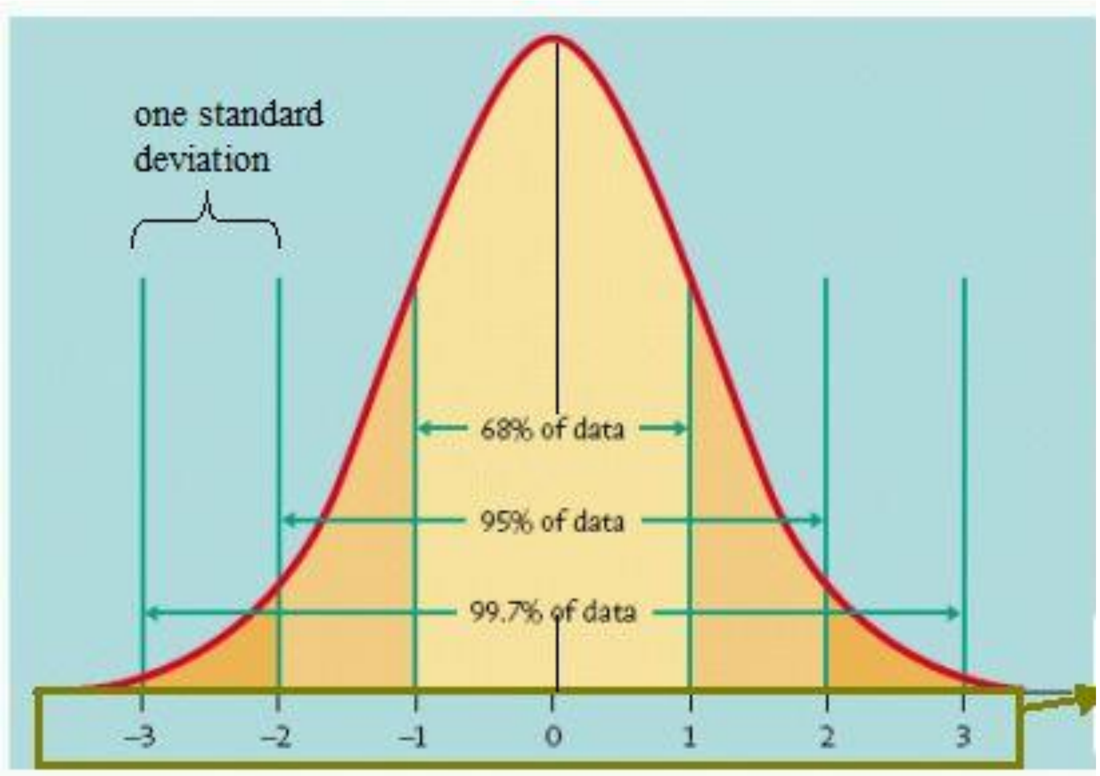
Example Calculated– Did you get this?

- 3-5.57= -2.57 squared = 6.6047; 4-5.57= -1.57 squared= 2.4645; 5 - 5. 57 = -. 57 squared = .3249; 5 - 5. 57 = -. 57 squared = .3249; 6- 5.57 = .43 squared = .1849; 7- 5.57 = 1.43 squared= 2.0449; 9-5.57= 3.43 squared =11.7649
- Added up (6.6047 + 2.4645 + .3249 + .3249 + .1849 + 2.0449 + 11.7649) = 23.7137
- Divide by 7 scores = 3.39= variance

Standard Deviation

- Take the Variance and Take its Square Root (e.g., Variance= 3.39; SD = 1.84)
- It is the average amount that scores differ from the mean
- Gives a good idea of SPREAD! (a small SD means little spread while a large SD means more spread)
- *Most articles report the Mean and Standard Deviation*

Outlier detection

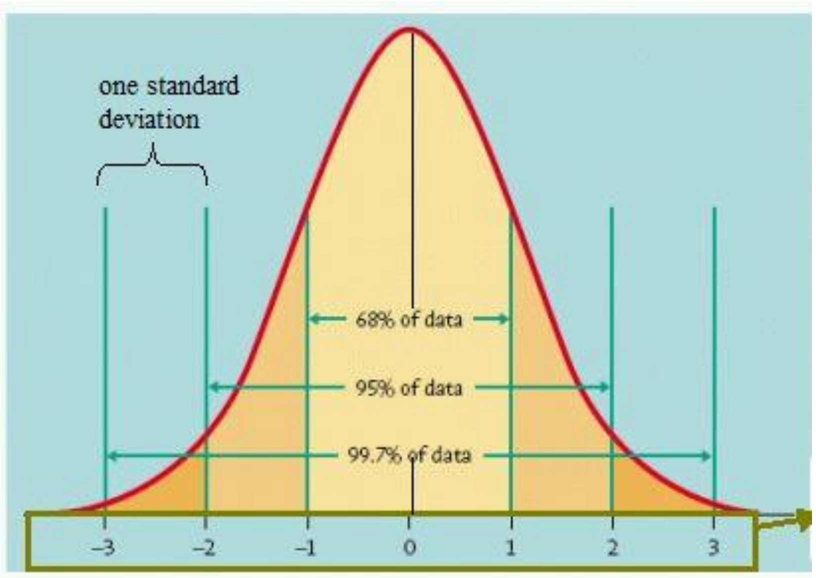


If we know the data is distributed normally (i.e. via a normal/gaussian distribution)

Outliers in a single dimension

Examples in a single dimension that have values greater than
$|k\sigma|$ can be discarded (for k >>3)

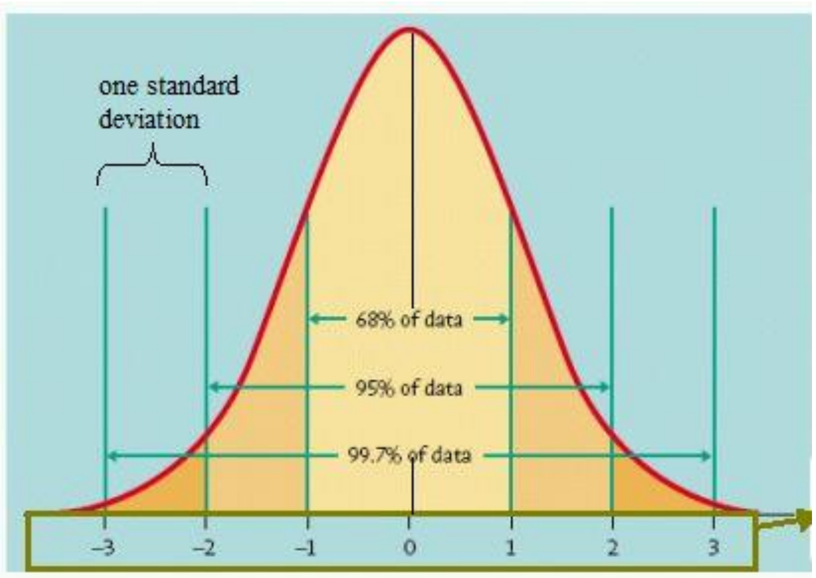
Even if the data isn't actually distributed normally, this is still often reasonable



Outliers in general

- Calculate the centroid/center of the data
- Calculate the average distance from center for all data
- Calculate standard deviation and discard points too far away

Again, many, many other techniques for doing this



Outliers for machine learning

Some good practices:

- Throw out conflicting examples
- Throw out any examples with obviously extreme feature values (i.e. many, many standard deviations away)
- Check for erroneous feature values (e.g. negative values for a feature that can only be positive)
- Let the learning algorithm/other pre-processing handle the rest

So far…

1. Throw out outlier examples

Feature pruning/selection

Good features provide us information that helps us distinguish between labels. However, not all features are good

Feature pruning is the process of removing “bad” features

Feature selection is the process of selecting “good” features

What makes a bad feature and why would we have them in our data?

Bad features

Each of you are going to generate a feature for our data set: pick 5 random binary numbers

f_1 f_2 …

label

I've already labeled these examples and I have two features

Bad features

label

1
0
1
1
0

If we have a “random” feature, i.e. a feature with random binary values, what is the probability that our feature perfectly predicts the label?

Bad features

label	f_i	probability
1	1	0.5
0	0	0.5
1	1	0.5
1	1	0.5
0	0	0.5

$0.5^5 = 0.03125 = 1/32$

Is that the only way to get perfect prediction?

Bad features

label	f_i	probability
1	0	0.5
0	1	0.5
1	0	0.5
1	0	0.5
0	1	0.5

$0.5^5 = 0.03125 = 1/32$

Total = 1/32+1/32 = 1/16

Why is this a problem?

Although these features perfectly correlate/predict the training data, they will generally NOT have any predictive power on the test set!

Bad features

label	f_i	probability
1	0	0.5
0	1	0.5
1	0	0.5
1	0	0.5
0	1	0.5

$0.5^5 = 0.03125 = 1/32$

Total = 1/32+1/32 = 1/16

Is perfect correlation the only thing we need to worry about for random features?

Bad features

label	f_i
1	1
0	0
1	1
1	0
0	0

Any correlation (particularly any strong correlation) can affect performance!

Noisy features

Adding features ***can*** give us more information, but not always

Determining if a feature is useful can be challenging

Terrain	Unicycle-type	Weather	Jacket	ML grade	Go-For-Ride?
Trail	Mountain	Rainy	Heavy	D	YES
Trail	Mountain	Sunny	Light	C-	YES
Road	Mountain	Snowy	Light	B	YES
Road	Mountain	Sunny	Heavy	A	YES
Trail	Normal	Snowy	Light	D+	NO
Trail	Normal	Rainy	Heavy	B-	NO
Road	Normal	Snowy	Heavy	C+	YES
Road	Normal	Sunny	Light	A-	NO
Trail	Normal	Sunny	Heavy	B+	NO
Trail	Normal	Snowy	Light	F	NO
Trail	Normal	Rainy	Light	C	YES

Noisy features

These can be particularly problematic in problem areas where we automatically generate features

Features

Clinton said banana repeatedly last week on tv, “banana, banana, banana”

(1, 1, 1, 0, 0, 1, 0, 0, ...)

clinton said
said banana
across the
california schools
tv banana
wrong way
capital city

Noisy features

Ideas for removing noisy/random features?

Terrain	Unicycle-type	Weather	Jacket	ML grade	Go-For-Ride?
Trail	Mountain	Rainy	Heavy	D	YES
Trail	Mountain	Sunny	Light	C-	YES
Road	Mountain	Snowy	Light	B	YES
Road	Mountain	Sunny	Heavy	A	YES
Trail	Normal	Snowy	Light	D+	NO
Trail	Normal	Rainy	Heavy	B-	NO
Road	Normal	Snowy	Heavy	C+	YES
Road	Normal	Sunny	Light	A-	NO
Trail	Normal	Sunny	Heavy	B+	NO
Trail	Normal	Snowy	Light	F	NO
Trail	Normal	Rainy	Light	C	YES

Removing noisy features

The expensive way:

- Split training data into train/dev
- Train a model on all features
- for each feature f:
  - Train a model on all features – f
  - Compare performance of all vs. all-f on dev set

- Remove all features where decrease in performance between all and all-f is less than some constant

Feature ablation study

Issues/concerns?

Removing noisy features

Binary features:

remove “rare” features, i.e. features that only occur (or don’t occur) a very small number of times

Real-valued features:

remove features that have low variance

In both cases, can either use thresholds, throw away lowest x%, use development data, etc.

Why?

Some rules of thumb for the number of features

Be very careful in domains where:

- the number of features > number of examples
- the number of features ≈ number of examples
- the features are generated automatically
- there is a chance of "random" features

In most of these cases, features should be removed based on some domain knowledge (i.e. problem-specific knowledge)

So far…

1. Throw out outlier examples
2. Remove noisy features
3. Pick “good” features

Feature selection

Let’s look at the problem from the other direction, that is, selecting good features.

What are good features?

How can we pick/select them?

Good features

A good feature correlates well with the label

label				
1	1	0	1	
0	0	1	1	
1	1	0	1	…
1	1	0	1	
0	0	1	0	

How can we identify this?

- training error (like for DT)
- correlation model
- statistical test
- probabilistic test
- …

Training error feature selection

- for each feature f:
  - calculate the training error if only feature f were used to pick the label

- rank each feature by this value
- pick top k, top x%, etc.
  - can use a development set to help pick k or x

So far…

1. Throw out outlier examples
2. Remove noisy features
3. Pick “good” features

Feature normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

Would our three classifiers (DT, k-NN and perceptron) learn the same models on these two data sets?

Feature normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

Decision trees don't care about scale, so they'd learn the same tree

Feature normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

k-NN: NO! The distances are biased based on feature magnitude.

$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

Feature normalization

Length	Weight	Label
4	4	Apple
7	5	Apple
5	8	Banana

Which of the two examples are closest to the first?

Length	Weight	Label
40	4	Apple
70	5	Apple
50	8	Banana

$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

Feature normalization

Length	Weight	Label
4	4	Apple
7	5	Apple
5	8	Banana

$$D = \sqrt{(7-4)^2 + (5-4)^2} = \sqrt{10}$$

$$D = \sqrt{(5-4)^2 + (8-4)^2} = \sqrt{17}$$

Length	Weight	Label
40	4	Apple
70	5	Apple
50	8	Banana

$$D = \sqrt{(70-40)^2 + (5-4)^2} = \sqrt{901}$$

$$D = \sqrt{(70-50)^2 + (8-4)^2} = \sqrt{416}$$

$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

Feature normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

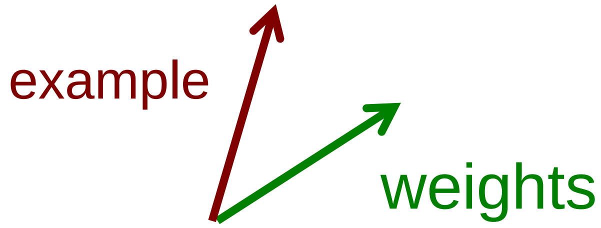
perceptron: NO!
The classification and weight update are based on the magnitude of the feature value

Geometric view of perceptron update

for each w_i:

$$w_i = w_i + f_i * \text{label}$$

Geometrically, the perceptron update rule is equivalent to “adding” the weight vector and the feature vector

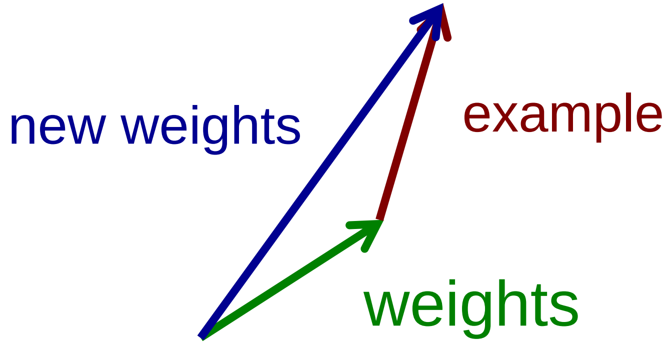


Geometric view of perceptron update

for each w_i:

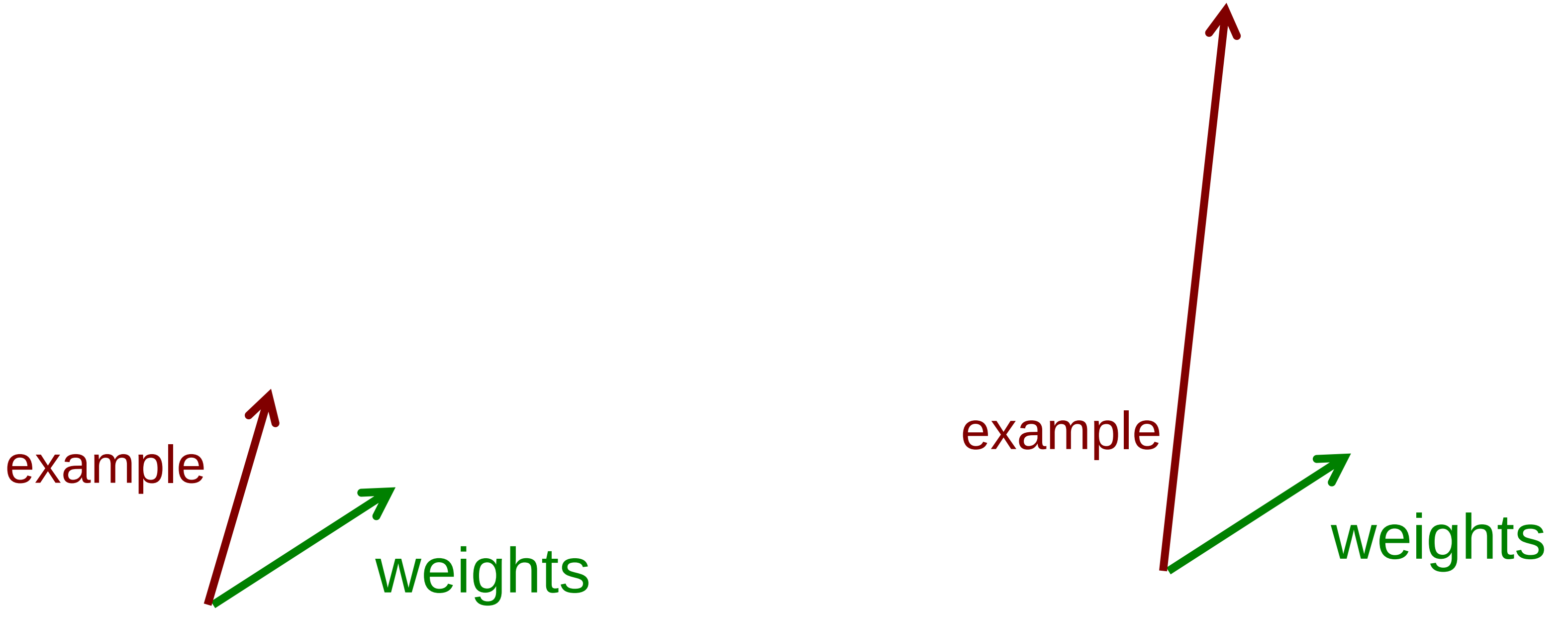
$$w_i = w_i + f_i * \text{label}$$

Geometrically, the perceptron update rule is equivalent to "adding" the weight vector and the feature vector



Geometric view of perceptron update

If the features dimensions differ in scale, it can bias the update



same f1 value, but larger f2

Geometric view of perceptron update

If the features dimensions differ in scale, it can bias the update

- different separating hyperplanes
- the larger dimension becomes much more importa

Feature normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

How do we fix this?

Feature normalization

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

Modify all values for a given feature

Normalize each feature

For each feature (over all examples):

Center: adjust the values so that the mean of that feature is 0. How do we do this?

Normalize each feature

For each feature (over all examples):

Center: adjust the values so that the mean of that feature is 0: subtract the mean from all values

Rescale/adjust feature values to avoid magnitude bias. Ideas?

Normalize each feature

For each feature (over all examples):

Center: adjust the values so that the mean of that feature is 0: subtract the mean from all values

Rescale/adjust feature values to avoid magnitude bias:

- Variance scaling: divide each value by the std dev
- Absolute scaling: divide each value by the largest value
- Data normalization > http://www.analytictech.com/ba762/handouts/normalization.htm

Pros/cons of either scaling technique?

So far…

1. Throw out outlier examples
2. Remove noisy features
3. Pick “good” features
4. Normalize feature values
   1. center data
   2. scale data (either variance or absolute)

Example normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

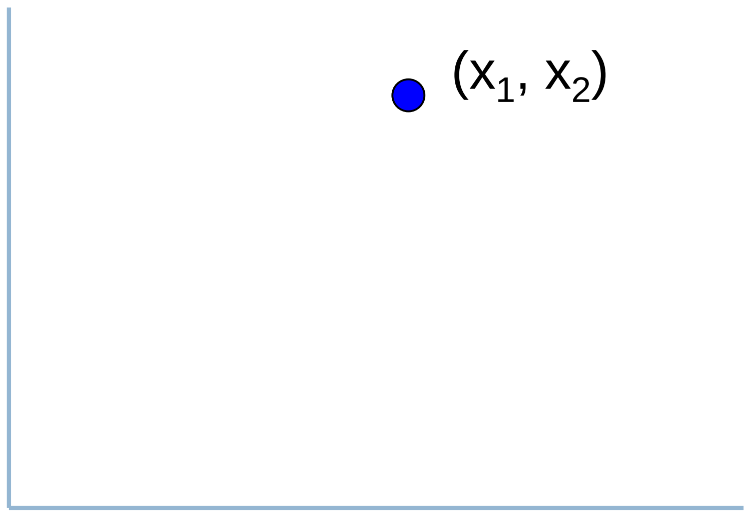
Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
70	60	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Any problem with this?
Solutions?

Example length normalization

Make all examples roughly the same scale, e.g. make all have length = 1

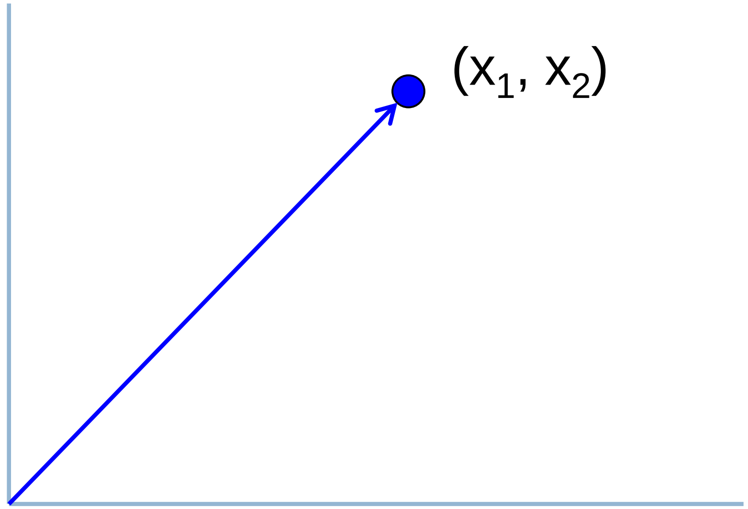
What is the length of this example/vector?



Example length normalization

Make all examples roughly the same scale, e.g. make all have length = 1

What is the length of this example/vector?

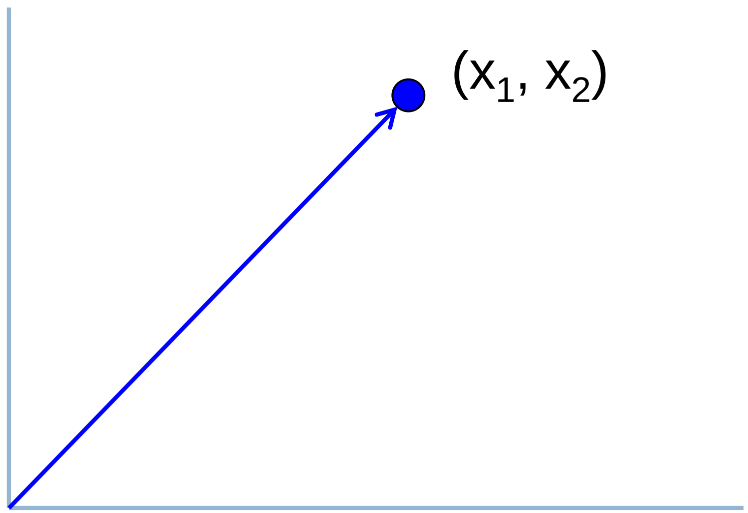


$$length(x) = \|x\| = \sqrt{x_1^2 + x_2^2}$$

Example length normalization

Make all examples roughly the same scale, e.g. make all have length = 1

What is the length of this example/vector?



$$length(x) = \|x\| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$$

Example length normalization

Make all examples have length = 1

Divide each feature value by ||x||

- Prevents a single example from being too impactful
- Equivalent to projecting each example onto a unit sphere

$$length(x) = \|x\| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$$

So far…

1. Throw out outlier examples
2. Remove noisy features
3. Pick “good” features
4. Normalize feature values
    1. center data
    2. scale data (either variance or absolute)
5. Normalize example length
6. Finally, train your model!

What about testing?

training data
(labeled examples)

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

pre-process data

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

"better" training data

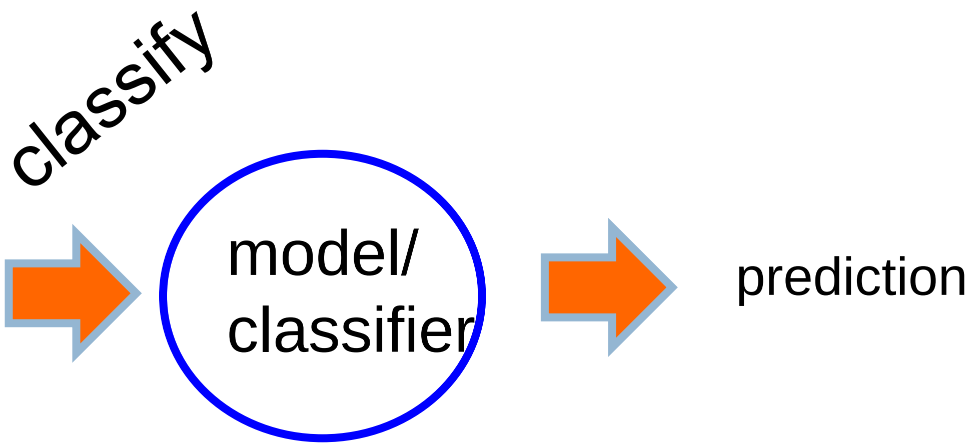
learn

model/
classifier

What about testing?

test data

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES



What about testing?

test data

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

pre-process data

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

classify

model/
classifier

prediction

How do we preprocess the test data?

Test data preprocessing

1. Throw out outlier examples
2. Remove noisy features
3. Pick “good” features
4. Normalize feature values
    1. center data
    2. scale data (either variance or absolute)
5. Normalize example length

Which of these do we need to do on test data? Any issues?

Test data preprocessing

1. ~~Throw out outlier examples~~
2. Remove irrelevant/noisy features
3. Pick "good" features
4. Normalize feature values
   1. center data
   2. scale data (either variance or absolute)
5. Normalize example length

Remove/pick same features

Do these

Do this

Whatever you do on training, you have to do the EXACT same on testing!

Normalizing test data

For each feature (over all examples):

Center: adjust the values so that the mean of that feature is 0: subtract the mean from all values

Rescale/adjust feature values to avoid magnitude bias:

- Variance scaling: divide each value by the std dev
- Absolute scaling: divide each value by the largest value

What values do we use when normalizing testing dat

Normalizing test data

For each feature (over all examples):

Center: adjust the values so that the mean of that feature is 0: subtract the mean from all values

Rescale/adjust feature values to avoid magnitude bias:

- Variance scaling: divide each value by the std dev
- Absolute scaling: divide each value by the largest value

Save these from training normalization!

Normalizing test data

training data
(labeled examples)

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

pre-process data

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

learn

model/
classifier

mean, std dev, max,…

test data

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

classify

model/
classifier

prediction

Features pre-processing summary

Many techniques for preprocessing data

Which will work well will depend on the data and the classifier

Try them out and evaluate how they affect performance on dev data

Make sure to do **exact same** pre-processing on train and test

1. Throw out outlier examples
2. Remove noisy features
3. Pick “good” features
4. Normalize feature values
    1. center data
    2. scale data (either variance or absolute)
5. Normalize example length