GEOMETRIC VIEW OF DATA

David Kauchak
CS 158 – Fall 2019

Admin

Assignment 2

Assignment 1 solution posted on sakai (use them to debug!)

Assignment 1 back soon

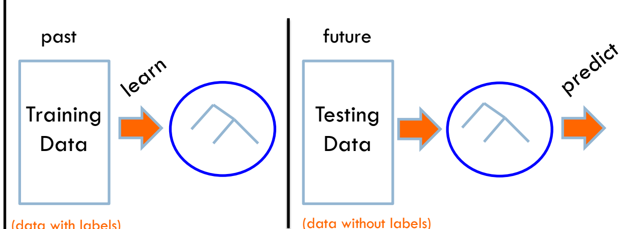
Keep reading

Videos?

Proper Experimentation

Experimental setup

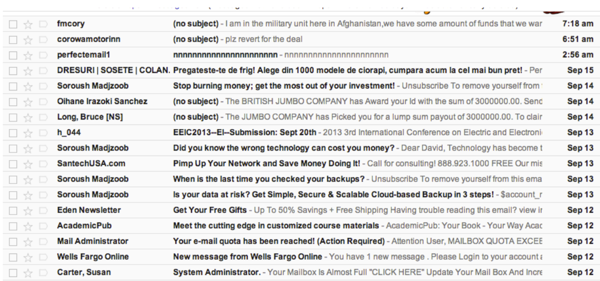
REAL WORLD USE OF ML ALGORITHMS



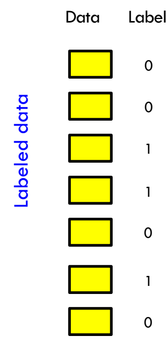
How do we tell how well we're doing?

Real-world classification

Google has labeled training data, for example from people clicking the "spam" button, but when new messages come in, they're not labeled



Classification evaluation

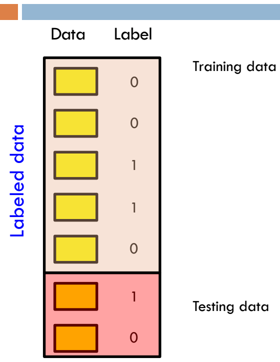


Use the labeled data we have already to create a test set with known labels!

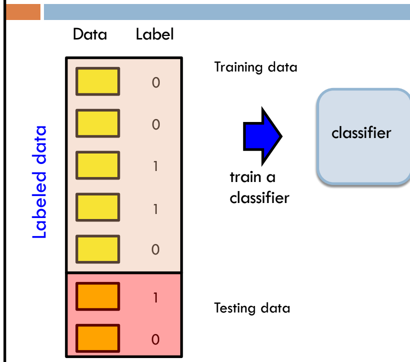
Why can we do this?

Remember, we assume there's an underlying distribution that generates both the training and test examples

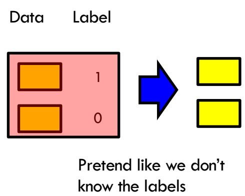
Classification evaluation



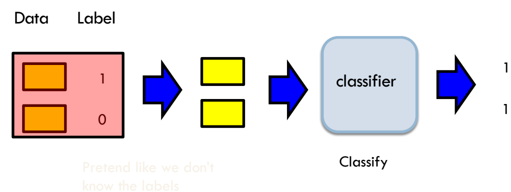
Classification evaluation



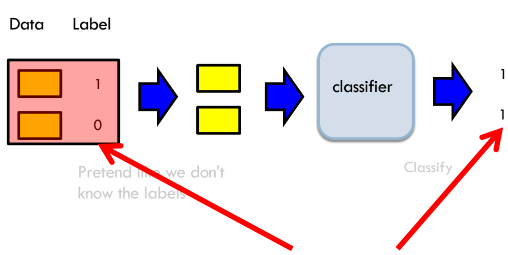
Classification evaluation



Classification evaluation



Classification evaluation

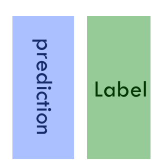


How could we score these for classification?

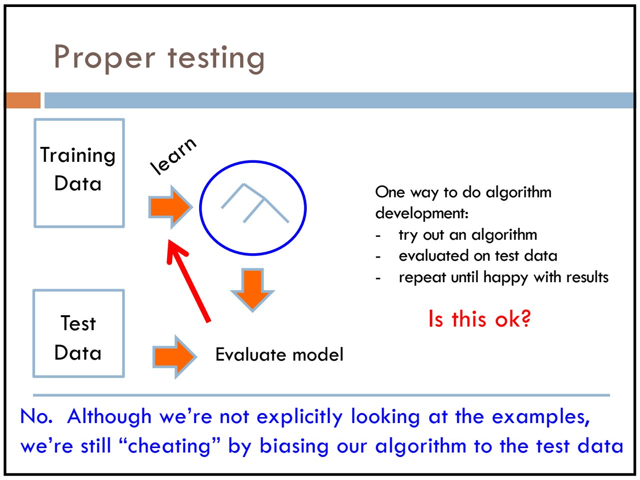
Compare predicted labels to actual labels

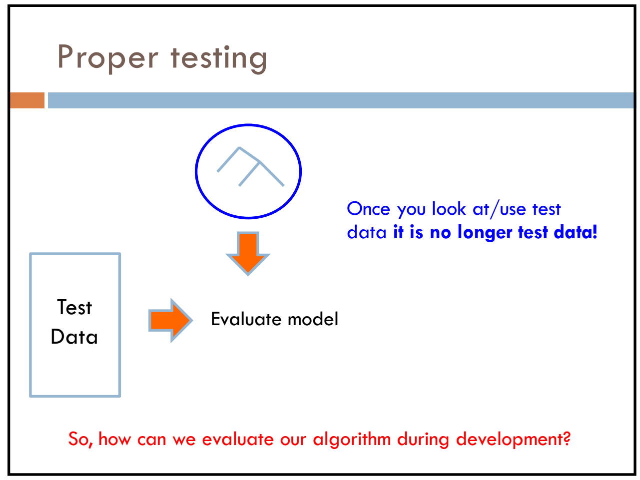
Test accuracy

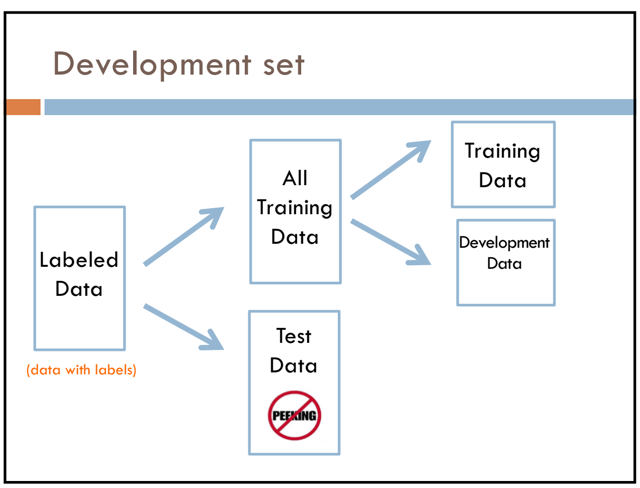
To evaluate the model, compare the predicted labels to the actual labels

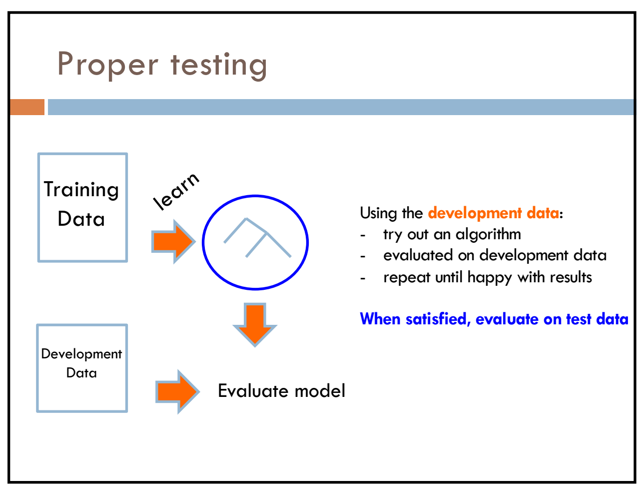


Accuracy: the proportion of examples where we correctly predicted the label

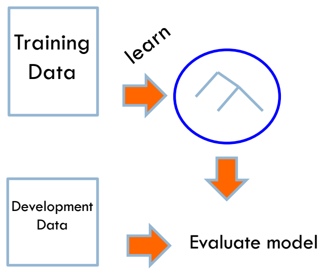

Proper testing
Training Data
learn
Test Data
Evaluate model
One way to do algorithm development:
- try out an algorithm
- evaluated on test data
- repeat until happy with results
Is this ok?
No. Although we're not explicitly looking at the examples, we're still "cheating" by biasing our algorithm to the test data


Proper testing
Once you look at/use test data it is no longer test data!
Test Data
Evaluate model
So, how can we evaluate our algorithm during development?


Development set
Labeled Data
(data with labels)
All Training Data
Training Data
Development Data
Test Data
PEEKING


Proper testing
Training Data
learn
Development Data
Evaluate model
Using the development data:
- try out an algorithm
- evaluated on development data
- repeat until happy with results
When satisfied, evaluate on test data

Proper testing

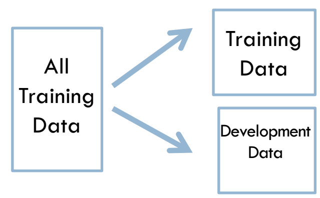


Using the **development data**:
- try out an algorithm
- evaluated on development data
- repeat until happy with results

Any problems with this?

Overfitting to development data

Be careful not to overfit to the development data!

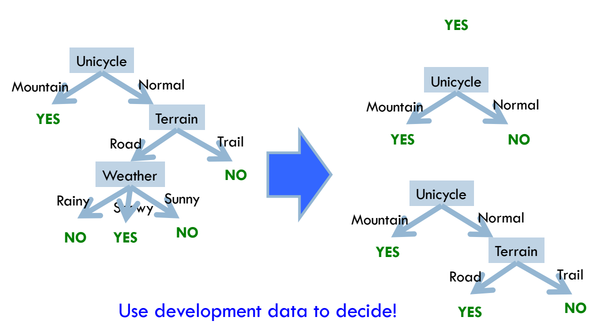


Often we'll split off development data multiple times (in fact, on the fly)… you can still overfit, but this helps avoid it

Pruning revisited

Which should we pick?

Pruning revisited



Use development data to decide!

Machine Learning: A Geometric View

Apples vs. Bananas

Weight	Color	Label
4	Red	Apple
5	Yellow	Apple
6	Yellow	Banana
3	Red	Apple
7	Yellow	Banana
8	Yellow	Banana
6	Yellow	Apple

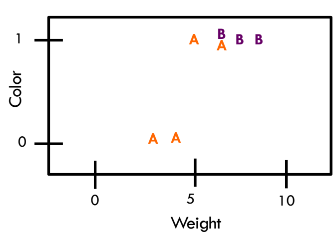
Can we visualize this data?

Apples vs. Bananas

Turn features into numerical values

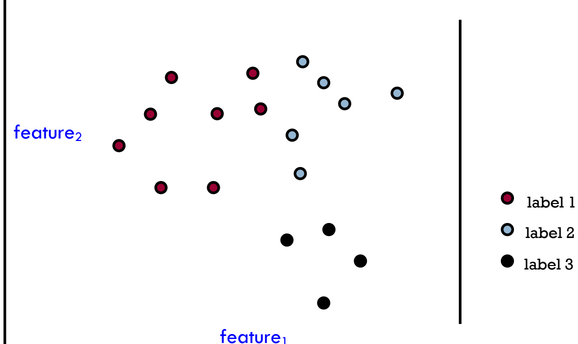
(read the book for a more detailed discussion of this)

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

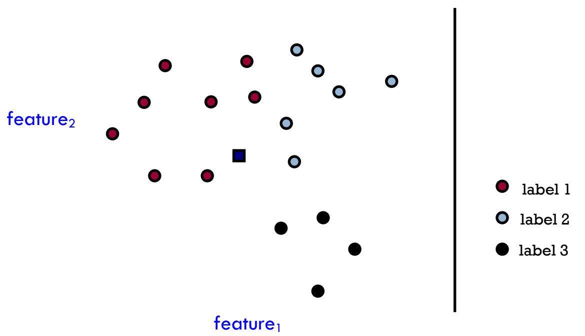


We can view examples as points in an *n*-dimensional space where *n* is the number of features

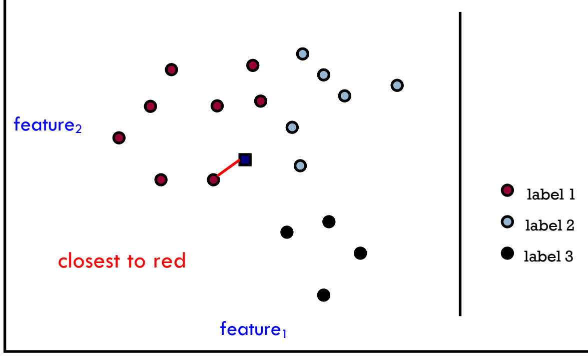
Examples in a feature space



Test example: what class?



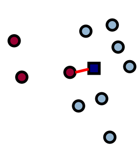
Test example: what class?



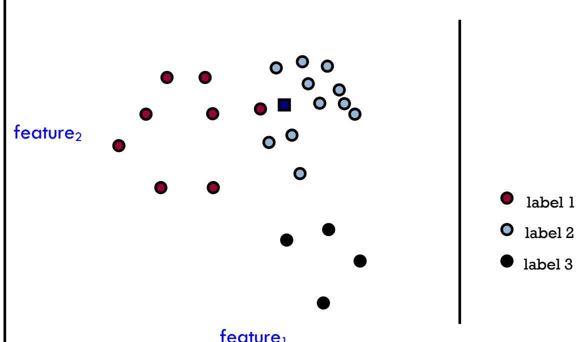
Another classification algorithm?

To classify an example ***d***:

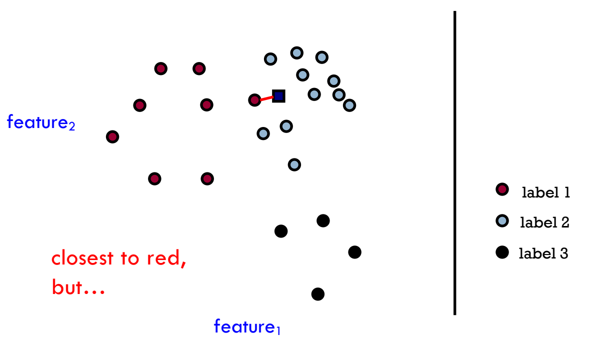
Label ***d*** with the label of the closest example to ***d*** in the training set



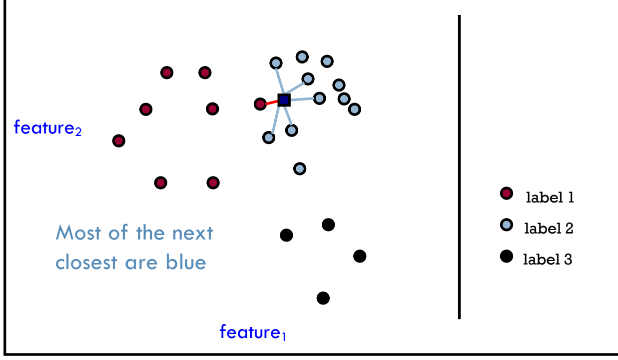
What about his example?



What about his example?



What about his example?



k-Nearest Neighbor (k-NN)

To classify an example ***d***:

- Find ***k*** nearest neighbors of ***d***
- Choose as the label the majority label within the ***k*** nearest neighbors

k-Nearest Neighbor (k-NN)

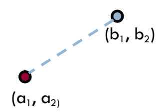
To classify an example ***d***:

- Find ***k*** *nearest* neighbors of ***d***
- Choose as the label the majority label within the ***k*** nearest neighbors

How do we measure "nearest"?

Euclidean distance

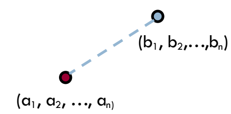
In two dimensions, how do we compute the distance?



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

Euclidean distance

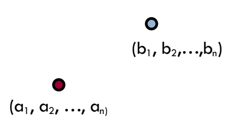
In n-dimensions, how do we compute the distance?



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

Euclidean distance

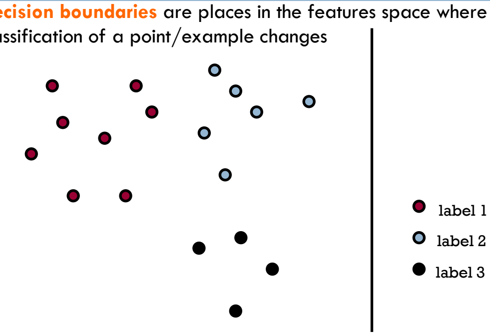
In n-dimensions, how do we compute the distance?



Measuring distance/similarity is a domain-specific problem and there are many, many different variations!

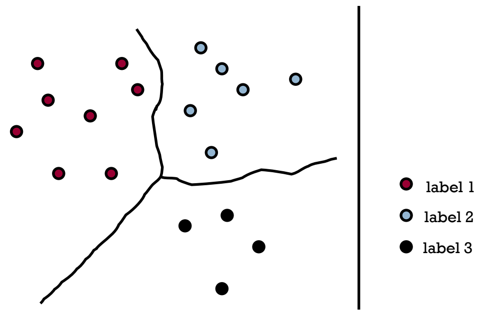
Decision boundaries

The **decision boundaries** are places in the features space where the classification of a point/example changes



Where are the decision boundaries for k-NN?

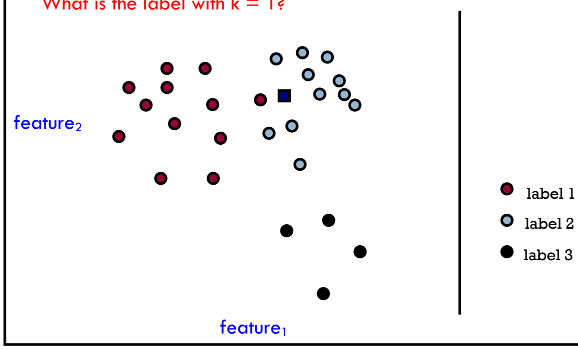
k-NN decision boundaries



k-NN gives locally defined decision boundaries between classes

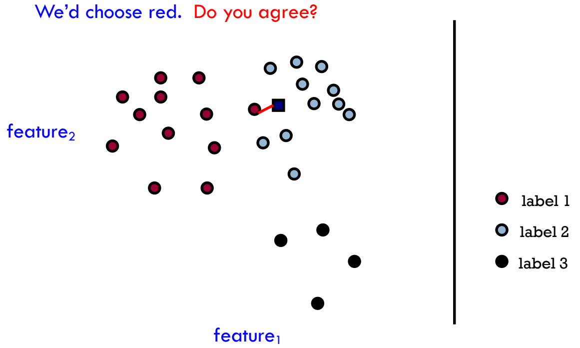
Choosing k

What is the label with k = 1?



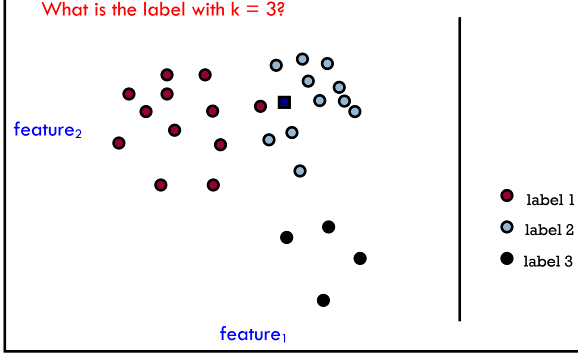
Choosing k

We'd choose red. Do you agree?



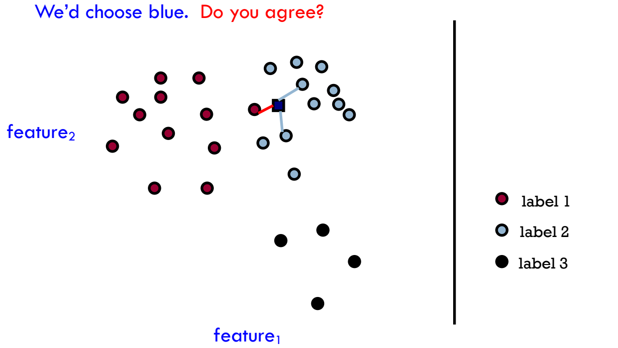
Choosing k

What is the label with k = 3?



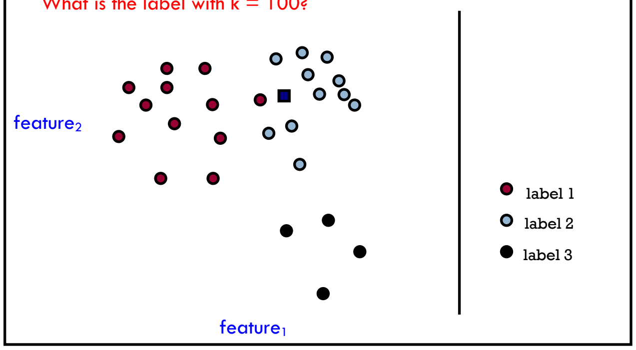
Choosing k

We'd choose blue. Do you agree?



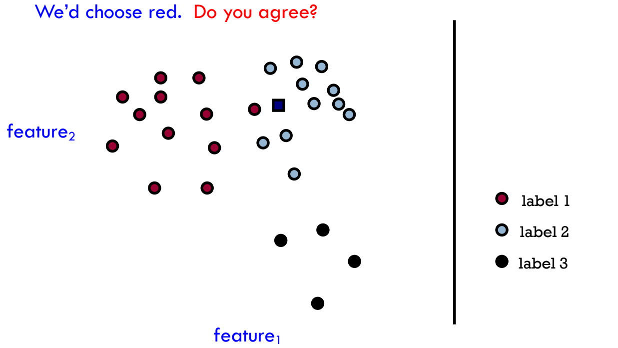
Choosing k

What is the label with k = 100?

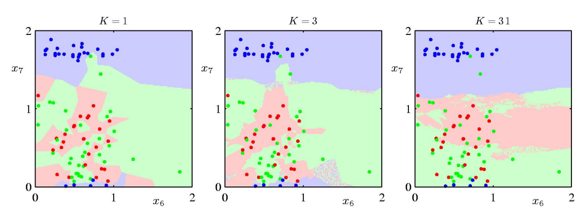


Choosing k

We'd choose red. Do you agree?



The impact of k



What is the role of k?
How does it relate to overfitting and underfitting?
How did we control this for decision trees?

k-Nearest Neighbor (k-NN)

To classify an example ***d***:

- Find ***k*** nearest neighbors of ***d***
- Choose as the class the majority class within the ***k*** nearest neighbors

How do we choose *k*?

How to pick k

Common heuristics:

- often 3, 5, 7
- choose an odd number to avoid ties

Use development data

k-NN variants

To classify an example ***d***:

- Find ***k*** nearest neighbors of ***d***
- Choose as the class the majority class within the ***k*** nearest neighbors

Any variation ideas?

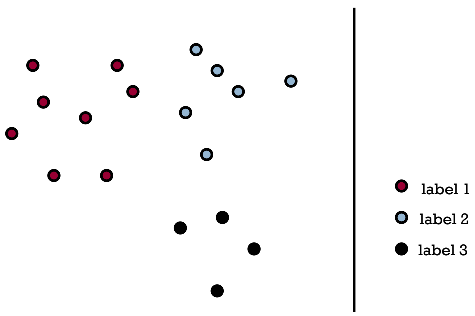
k-NN variations

Instead of *k* nearest neighbors, count majority from all examples within a fixed distance

Weighted *k*-NN:

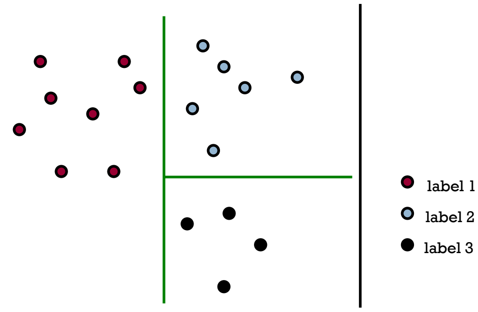
- Right now, all examples are treated equally
- weight the "vote" of the examples, so that closer examples have more vote/weight
- often use some sort of exponential decay

Decision boundaries for decision trees



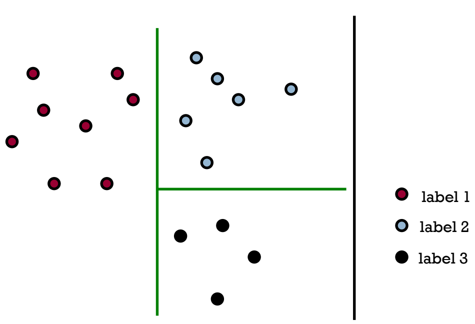
What are the decision boundaries for decision trees like?

Decision boundaries for decision trees



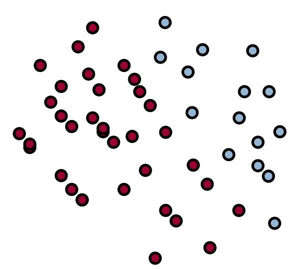
Axis-aligned splits/cuts of the data

Decision boundaries for decision trees



What types of data sets will DT work poorly on?

Problems for DT



Decision trees vs. *k*-NN

Which is faster to train?

Which is faster to classify?

Do they use the features in the same way to label the examples?

Decision trees vs. *k*-NN

Which is faster to train?

k-NN doesn't require any training!

Which is faster to classify?

For most data sets, decision trees

Do they use the features in the same way to label the examples?

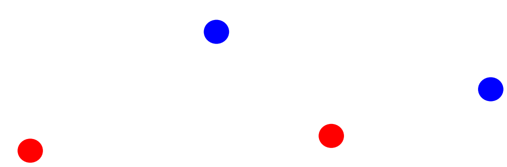
k-NN treats all features equally! Decision trees "select" important features

Machine learning models

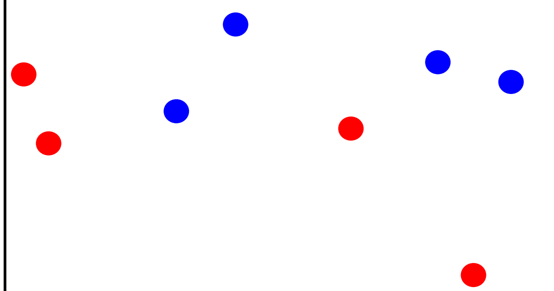
Some machine learning approaches make strong assumptions about the data
- If the assumptions are true it can often lead to better performance
- If the assumptions aren't true, the approach can fail miserably

Other approaches don't make many assumptions about the data
- This can allow us to learn from more varied data
- But, they are more prone to overfitting
- and generally require more training data

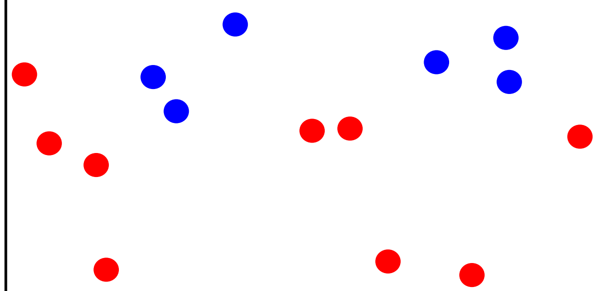
What is the data generating distribution?



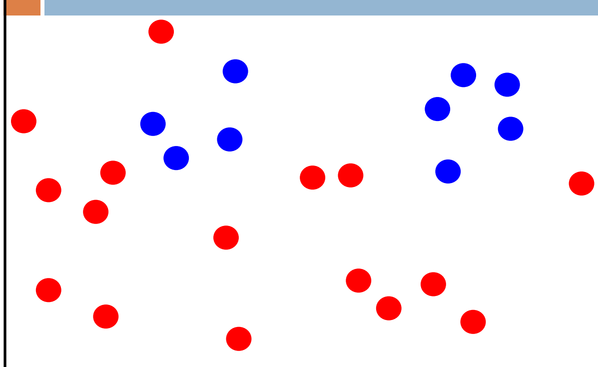
What is the data generating distribution?



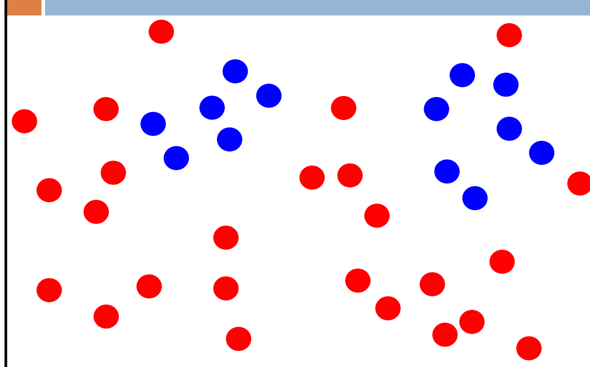
What is the data generating distribution?



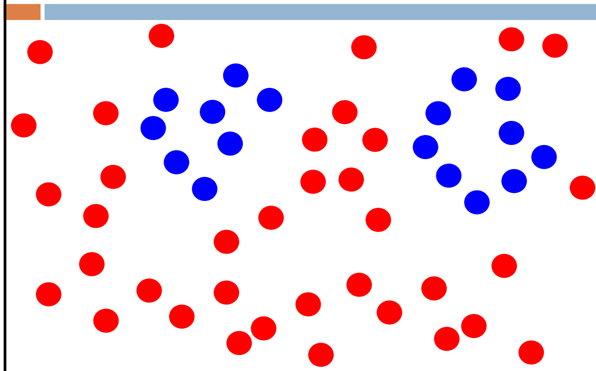
What is the data generating distribution?



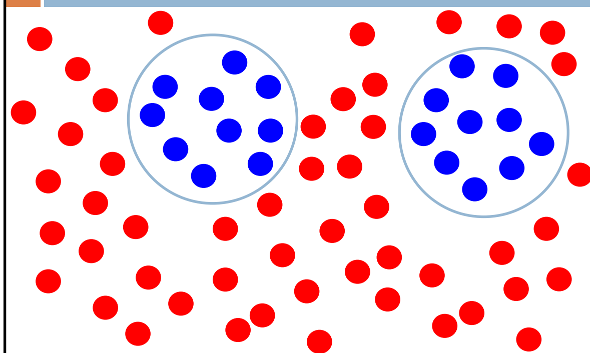
What is the data generating distribution?



What is the data generating distribution?



Actual model

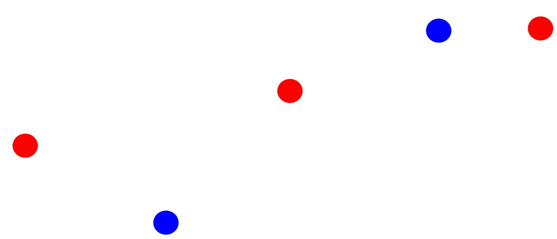


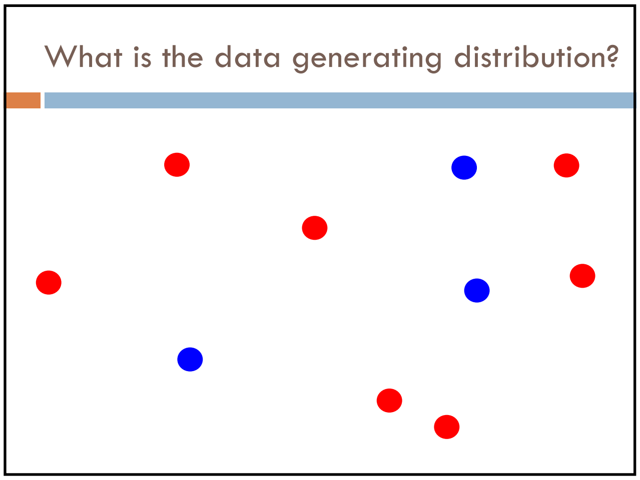
Model assumptions

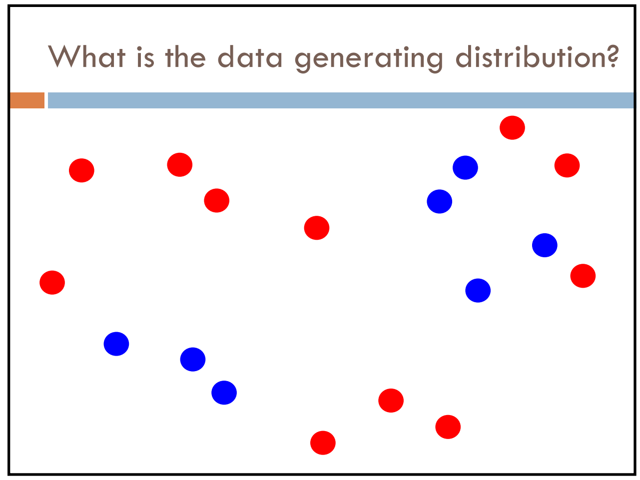
If you don't have strong assumptions about the model, it can take you a longer to learn

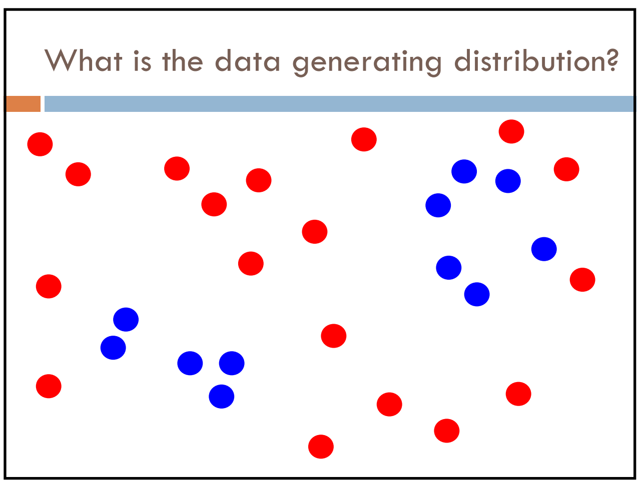
Assume now that our model of the blue class is two circles

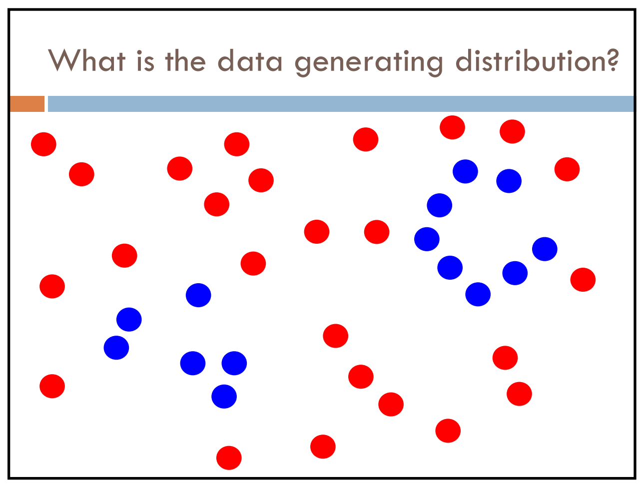
What is the data generating distribution?



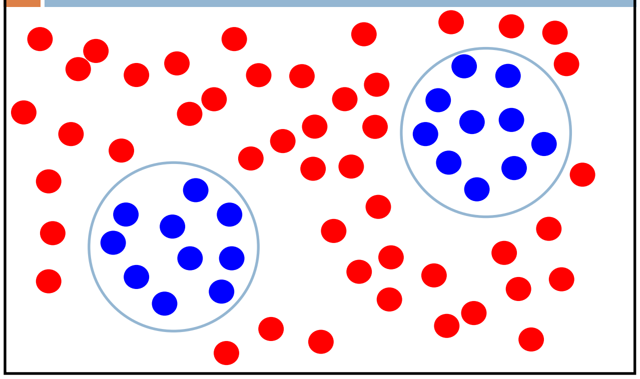

What is the data generating distribution?


What is the data generating distribution?

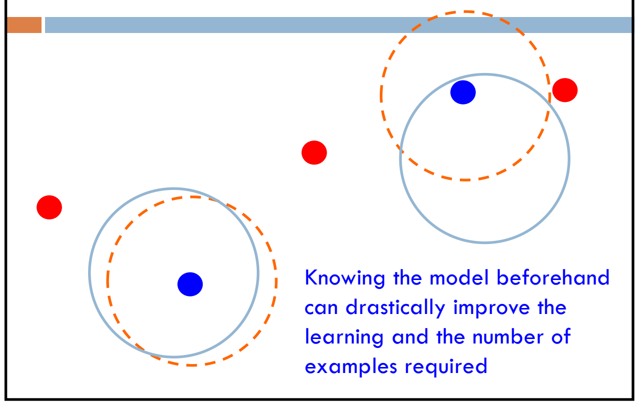

What is the data generating distribution?


What is the data generating distribution?

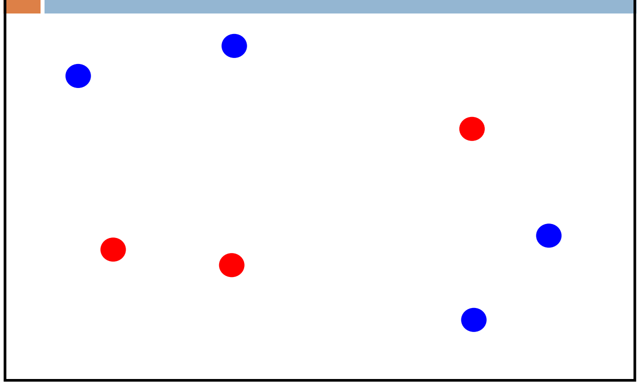
Actual model



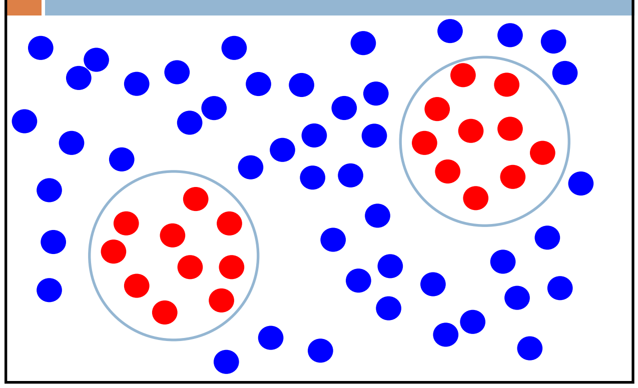
What is the data generating distribution?



What is the data generating distribution?



Make sure your assumption is correct, though!

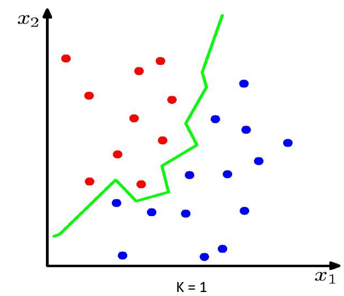


Machine learning models

What are the *model* assumptions (if any) that *k*-NN and decision trees make about the data?

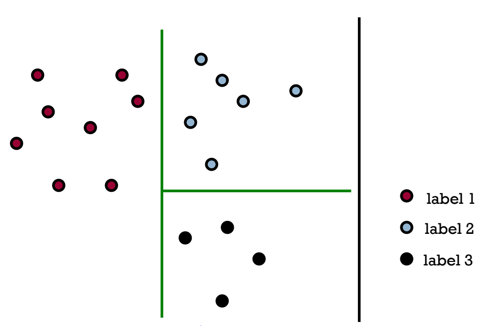
Are there data sets that could never be learned correctly by either?

k-NN model



No model assumptions. Assumes that proximity relates to class

Decision tree model



Axis-aligned splits/cuts of the data

Bias

The "bias" of a model is how strong the model assumptions are.

low-bias classifiers make minimal assumptions about the data (*k*-NN and DT are generally considered low bias)

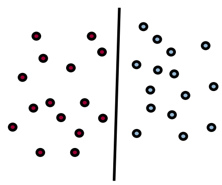
high-bias classifiers make strong assumptions about the data

Linear models

A strong high-bias assumption is *linear separability*:

- in 2 dimensions, can separate classes by a line
- in higher dimensions, need hyperplanes

A *linear model* is a model that assumes the data is linearly separable



Hyperplanes

A hyperplane is line/plane in a high-dimensional space

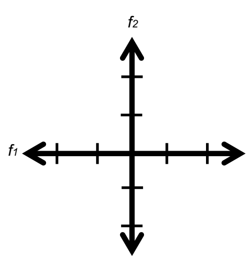
What defines a line?

What defines a hyperplane?

Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$$0 = w_1 f_1 + w_2 f_2$$



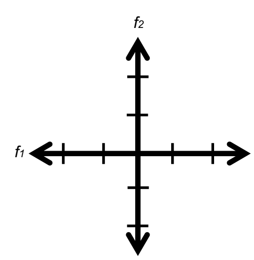
Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$$0 = w_1 f_1 + w_2 f_2$$

$$0 = 1 f_1 + 2 f_2$$

-2	1
-1	0.5
0	0
1	-0.5
2	-1



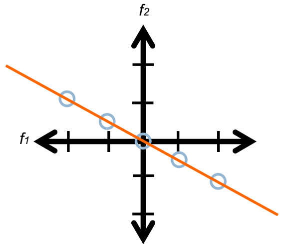
Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$$0 = w_1 f_1 + w_2 f_2$$

$$0 = 1 f_1 + 2 f_2$$

-2	1
-1	0.5
0	0
1	-0.5
2	-1



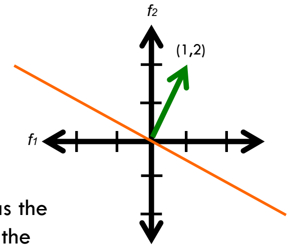
Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$$0 = w_1 f_1 + w_2 f_2$$

$$0 = 1 f_1 + 2 f_2$$

w=(1,2)

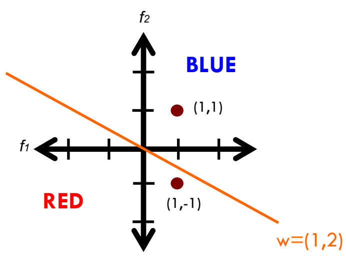


We can also view it as the line perpendicular to the *weight vector*

Classifying with a line

Mathematically, how can we classify points based on a line?

$$0 = 1 f_1 + 2 f_2$$



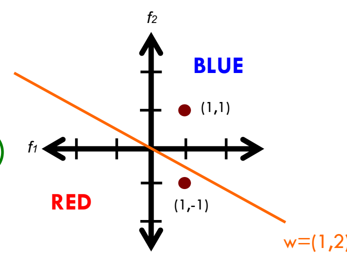
Classifying with a line

Mathematically, how can we classify points based on a line?

$$0 = 1 f_1 + 2 f_2$$

(1,1): $1*1+2*1=3$

(1,-1): $1*1+2*-1=-1$



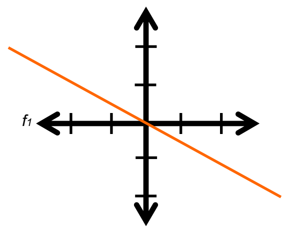
The sign indicates which side of the line

Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$$0 = w_1 f_1 + w_2 f_2$$

$$0 = 1 f_1 + 2 f_2$$



How do we move the line off of the origin?

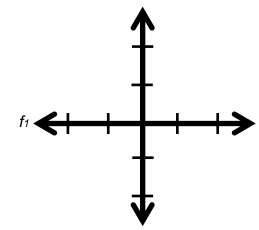
Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$$a = w_1 f_1 + w_2 f_2$$

$$-1 = 1 f_1 + 2 f_2$$

-2
-1
0
1
2



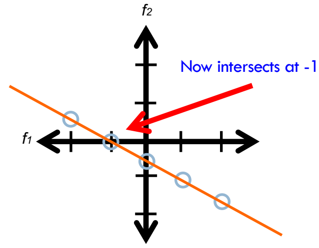
Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$$a = w_1 f_1 + w_2 f_2$$

$$-1 = 1 f_1 + 2 f_2$$

-2	0.5
-1	0
0	-0.5
1	-1
2	-1.5



Linear models

A linear model in n-dimensional space (i.e. n features) is define by n+1 weights:

In two dimensions, a line:

$$0 = w_1 f_1 + w_2 f_2 + b$$ (where b = -a)

In three dimensions, a plane:

$$0 = w_1 f_1 + w_2 f_2 + w_3 f_3 + b$$

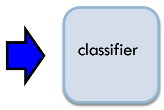
In n-dimensions, a *hyperplane*

$$0 = b + \sum_{i=1}^{n} w_i f_i$$

Classifying with a linear model

We can classify with a linear model by checking the sign:

$f_1, f_2, \ldots, f_n$



$b + \sum_{i=1}^{n} w_i f_i > 0$ Positive example

$b + \sum_{i=1}^{n} w_i f_i < 0$ Negative example

An aside: a thought experiment

What is a 100,000-dimensional space like?

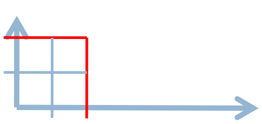
You're a 1-D creature, and you decide to buy a 2-unit apartment

2 rooms (very, skinny rooms)

Another thought experiment

What is a 100,000-dimensional space like?

Your job's going well and you're making good money. You upgrade to a 2-D apartment with 2-units per dimension

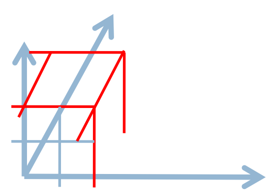


4 rooms (very, flat rooms)

Another thought experiment

What is a 100,000-dimensional space like?

You get promoted again and start having kids and decide to upgrade to another dimension.



8 rooms (very, normal rooms)

Each time you add a dimension, the amount of space you have to work with goes up exponentially

Another thought experiment

What is a 100,000-dimensional space like?

Larry Page steps down as CEO of google and they ask you if you'd like the job. You decide to upgrade to a 100,000 dimensional apartment.

How much room do you have?
Can you have a big party?

$2^{100,000}$ rooms (it's very quiet and lonely…) = ~10^{30} rooms per person if you invited everyone on the planet

The challenge

Our intuitions about space/distance don't scale with dimensions!