

COMP6237 Data Mining

Document Filtering

Jonathon Hare

jsh2@ecs.soton.ac.uk

Introduction

- Supervised ML - Classification
- Spam filtering
- Naïve Bayes' Spam Filtering
- Fisher's method
- Feature Engineering

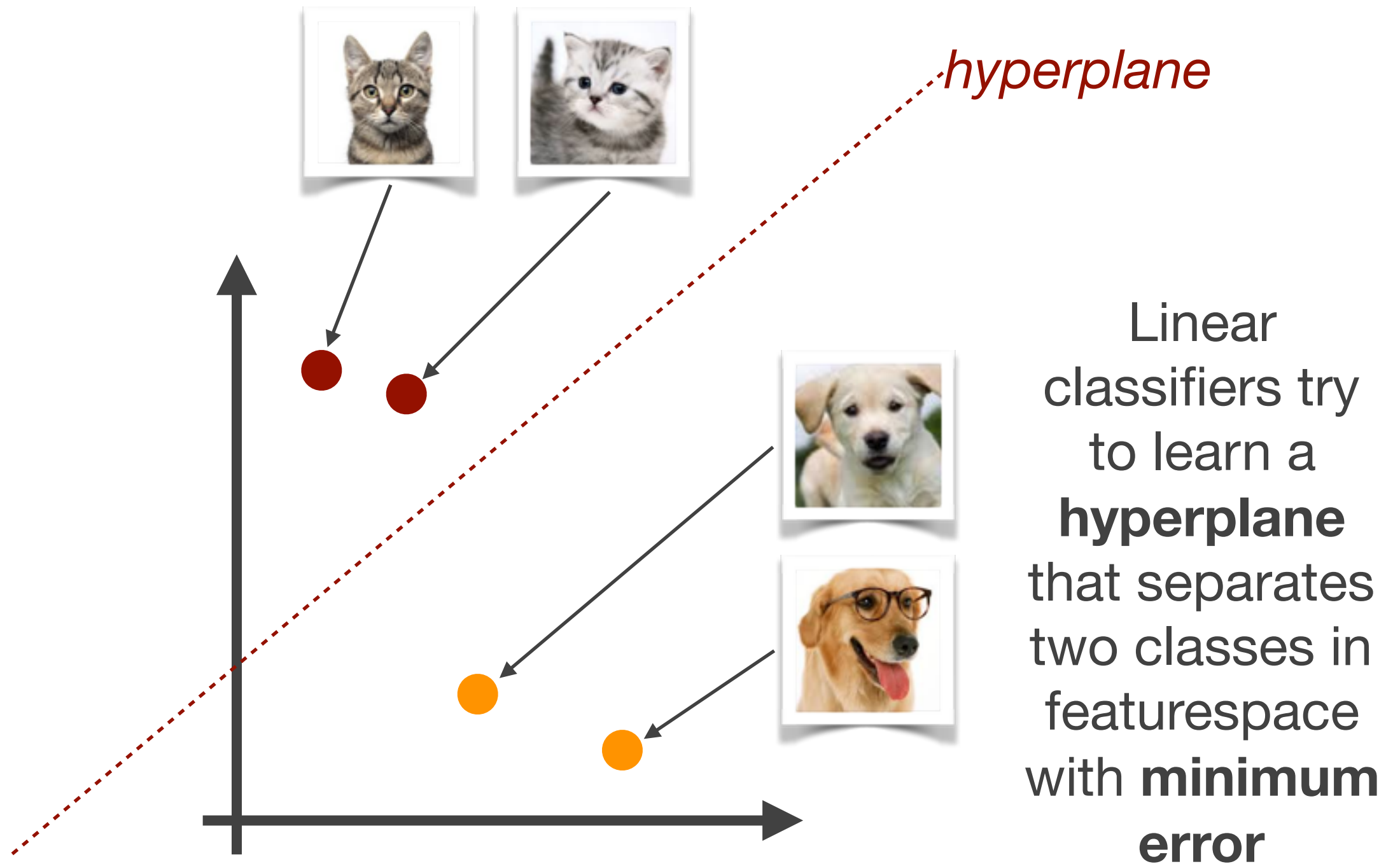
Classification 101 (recap)

Cat or Dog

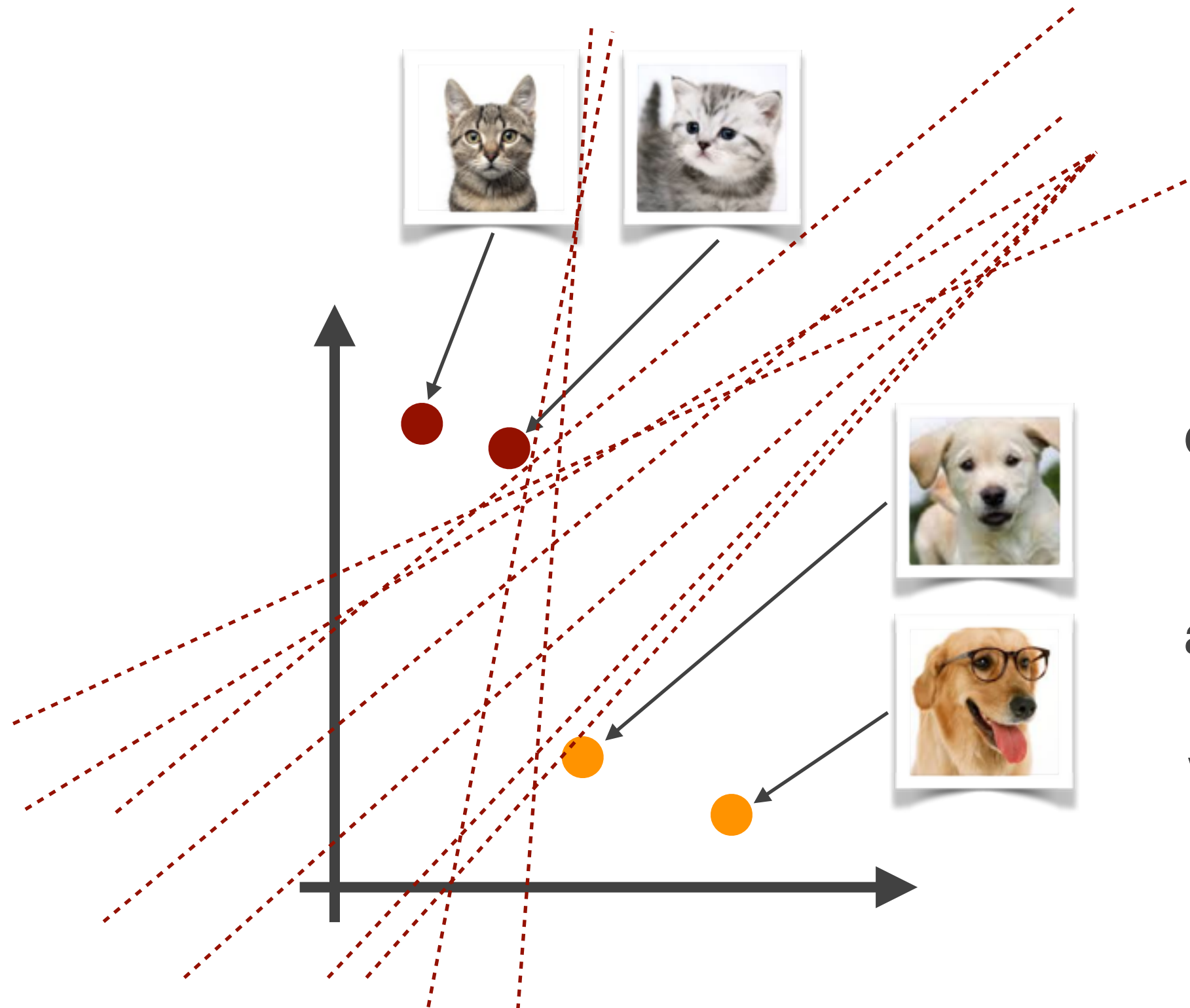
- **Classification** is the process of assigning a **class label** to an object (typically represented by a vector in a feature space).
- A **supervised machine-learning algorithm** uses a set of pre-labelled training data to learn how to assign class labels to vectors (and the corresponding objects).
- A **binary classifier** only has two classes
- A **multiclass** classifier has many classes.



Linear classifiers



Linear classifiers



Lots of
hyperplanes
to choose
from...
different linear
classification
algorithms
apply differing
constraints
when learning
the classifier

Linear classifiers



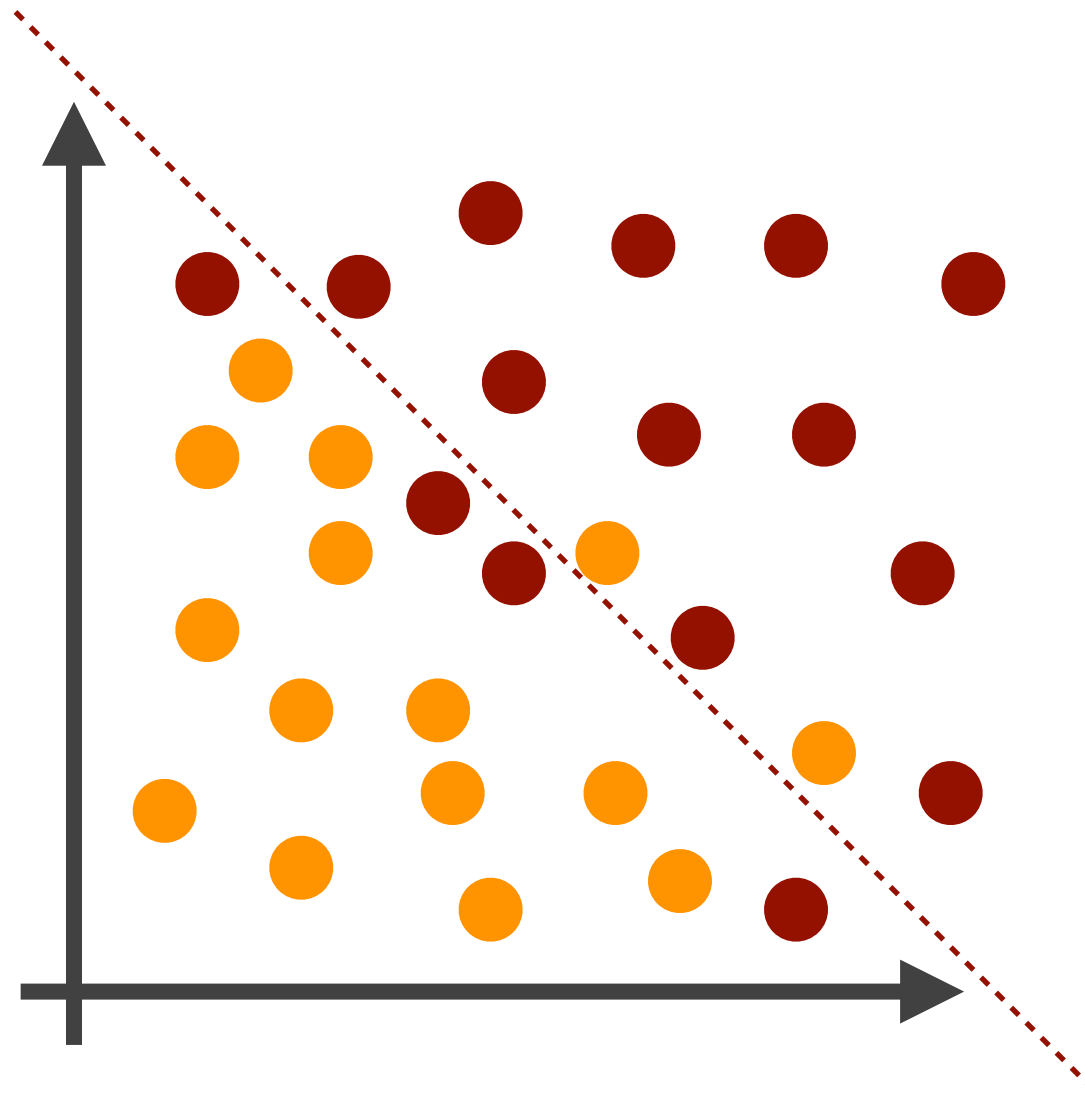
hyperplane

Cats

Dogs

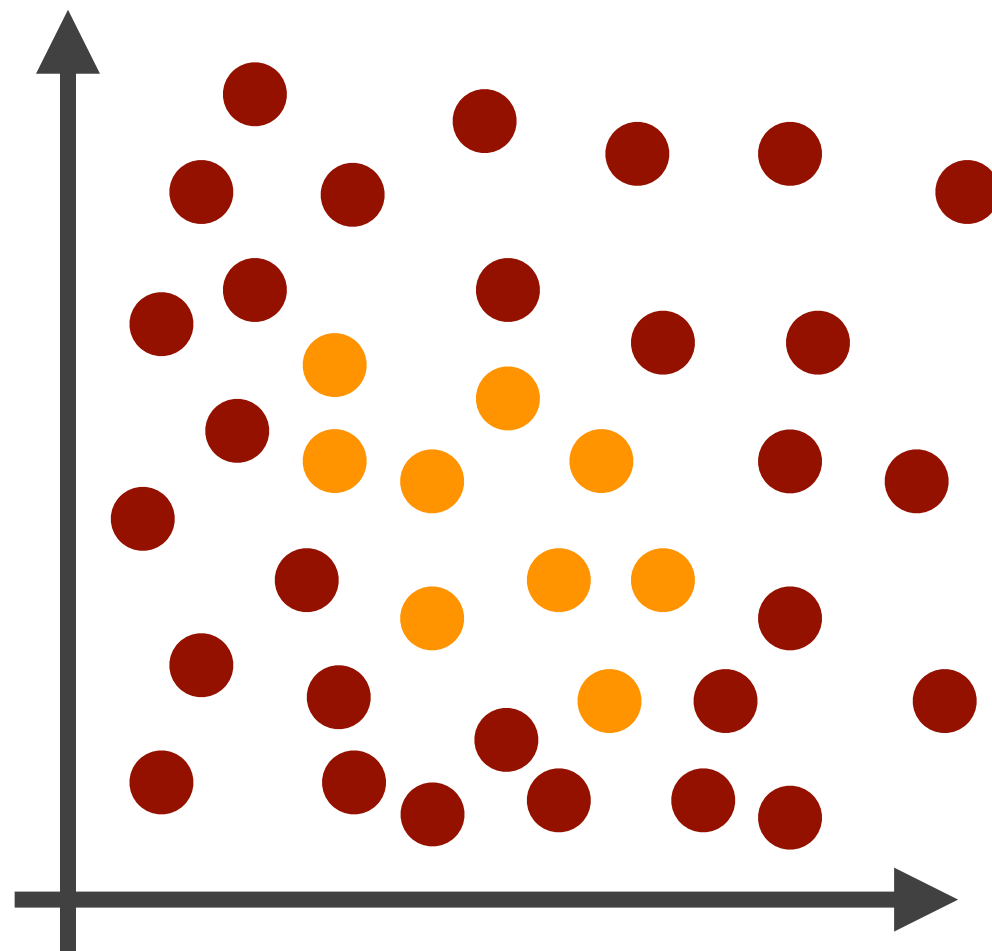
To classify a new image, you just need to check what side of the hyperplane it is on

Non-linear binary classifiers



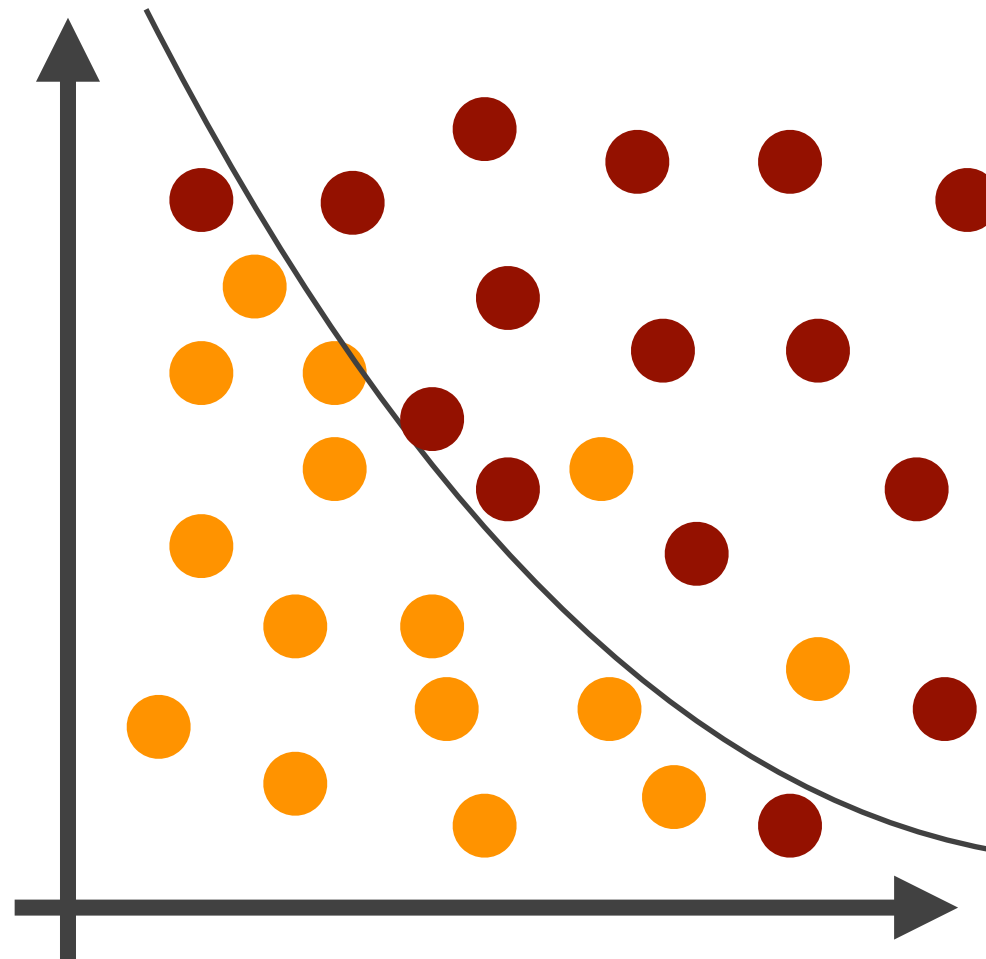
Linear
classifiers
work best
when the data
is linearly
separable...

Non-linear binary classifiers



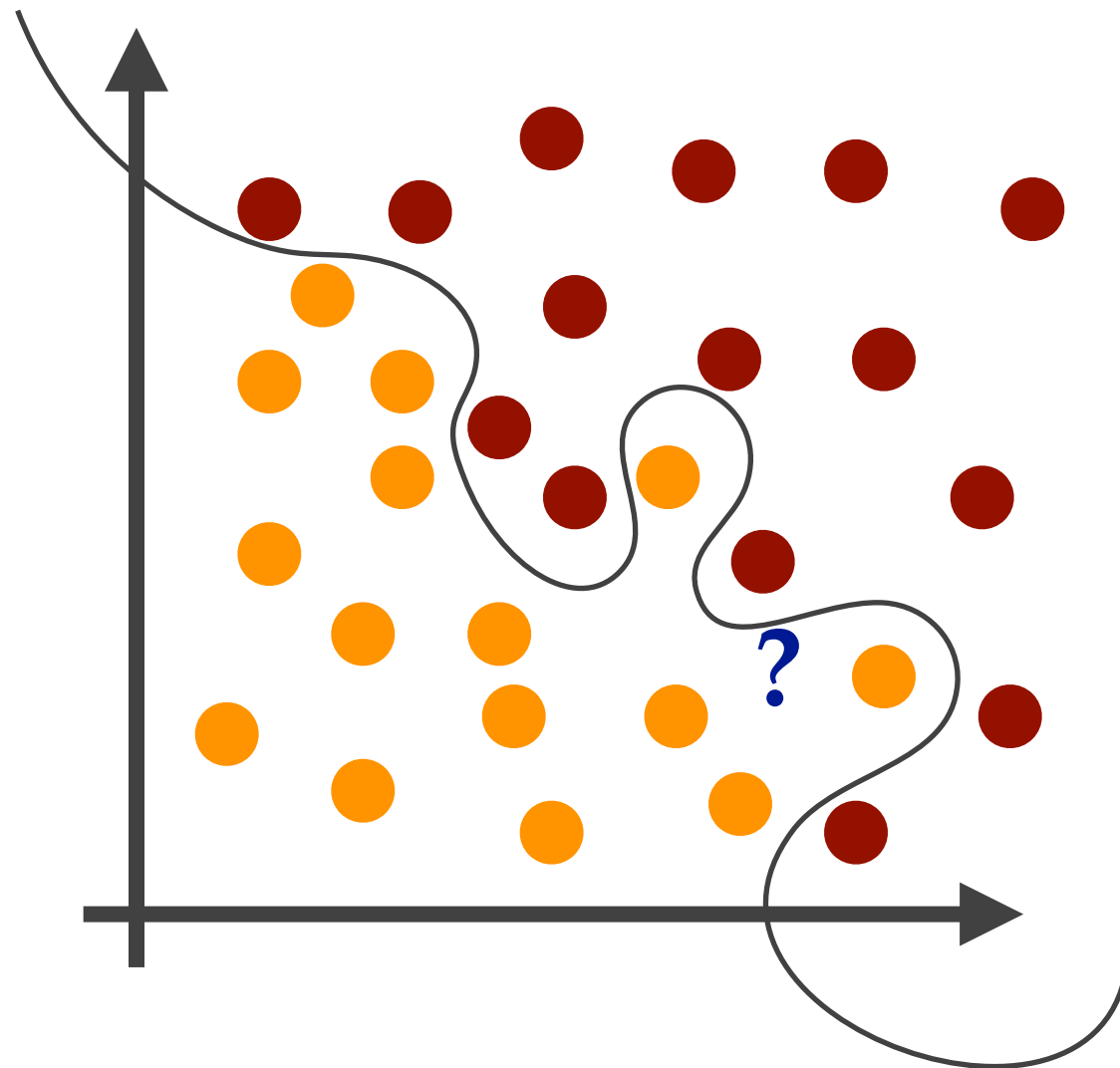
No hope for a
linear
classifier!

Non-linear binary classifiers



Non-linear
binary
classifiers,
such as
**Kernel
Support
Vector
Machines**
learn non-
linear decision
boundaries

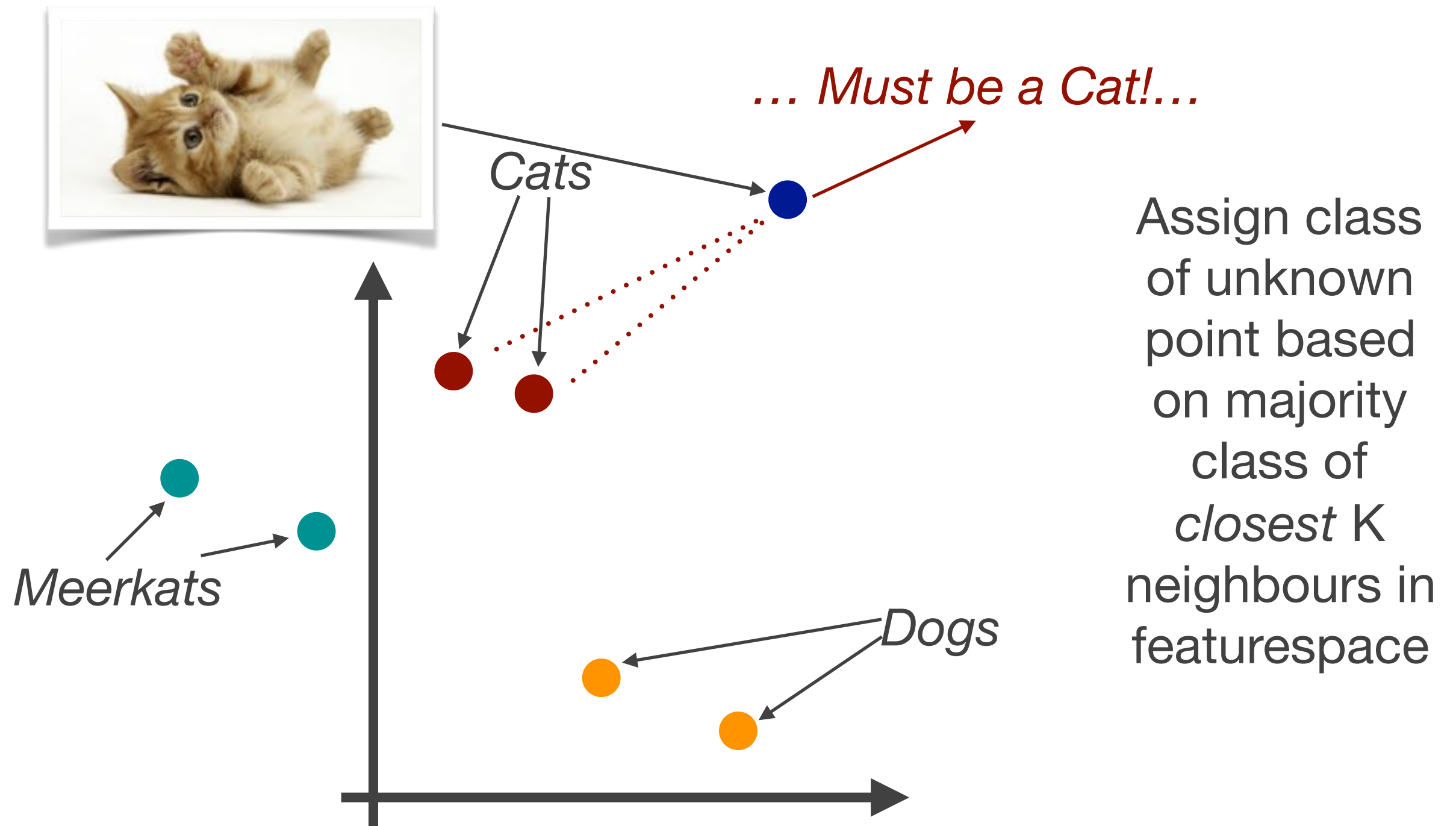
Non-linear binary classifiers



Have to be careful... you might lose generality by overfitting

Use *regularisation* to penalise models becoming too complex

Multiclass classifiers: KNN



Multiclass linear classifiers

- A linear classifier is by definition binary
- So, how can we solve multiclass problems with linear classifiers?
 - One versus All (OvA)/One versus Rest (OvR)
 - one classifier per class
 - One versus One (OvO)
 - $K(K - 1) / 2$ classifiers

Problem statement: Filtering Spam

- Assume we want to build a spam/ham classifier for emails (or other documents)
 - We need a classifier that can be easily updated online
 - Probabilistic approaches to classification are a natural fit...



Early (rule-based) spam filters

- In the early days of ubiquitous email spam started to become a problem
 - Early spam detection systems were hand-coded
 - ...Using rules - e.g.:
 - Emails that over-used capital letters were likely to be spam
 - Certain terms were strong indicators:
 - Viagra & similar pharmaceutical products!
 - Overly colourful or garish colours in embedded html
 - emails consisting of just an embedded picture

Early (rule-based) spam filters

- Unfortunately, this had a few problems...
 - parents sending emails in all uppercase
 - the spammers getting smart & adapting their strategies to circumvent the rules encoded in the software
 - lack of personalisation
 - what the user actually wanted to receive emails about drugs or mail-order brides!?
- Solution: use machine learning to build classifiers that suit the user

Learning to classify text: Naïve Bayes Classification

Initial features: bag of words

- We'll start by considering a simple bag of word representation
 - Simple tokenisation: split on non-letters
 - Simple pre-processing: convert to lower case
 - Don't count; just record presence/absence of a term
 - We'll refer to each *term* as a *feature*
- Better/more complex features are available - coming up later!

Counting features

- Given a set of labelled training documents (spam/ham)
 - create a table of how many times a document containing a feature has occurred in each category:

| | money | viagra | dad | mom | dinner | mail | ... |
|------|-------|--------|-----|-----|--------|------|-----|
| SPAM | 1 | 15 | 0 | 2 | 1 | 8 | |
| HAM | 0 | 0 | 3 | 3 | 8 | 2 | |

- create a table of how many documents fall into each category:

| SPAM | 25 |
|------|----|
| HAM | 72 |

Conditional probability of a feature given a category

- What is the probability that a feature occurs in a given category?
 - $p(f|c) = n(\text{docs with feature in category}) / n(\text{docs in category})$
 - This is a *conditional probability*
 - read $p(f|c)$ as *probability of f given c*
 - examples:
 - $p(\text{viagra}|\text{SPAM}) = 15/25 = 0.6$
 - i.e. there is a 60% chance a spam doc contains viagra
 - $p(\text{mail}|\text{HAM}) = 2/72 = 0.027$

| | |
|------|----|
| SPAM | 25 |
| HAM | 72 |

| | money | viagra | dad | mom | dinner | mail | ... |
|------|-------|--------|-----|-----|--------|------|-----|
| SPAM | 1 | 15 | 0 | 2 | 1 | 8 | |
| HAM | 0 | 0 | 3 | 3 | 8 | 2 | |

Smoothing probability estimates

- Consider $p(\text{money}|\text{HAM})$
 - probability is 0
 - So *money* should never be expected to appear in HAM documents?
- Need a better way to estimate the conditional $p(f|c)$ that accounts for infrequently seen features (or more formally *insufficient sample size*)

| | |
|------|----|
| SPAM | 25 |
| HAM | 72 |

| | money | viagra | dad | mom | dinner | mail | ... |
|------|-------|--------|-----|-----|--------|------|-----|
| SPAM | 1 | 15 | 0 | 2 | 1 | 8 | |
| HAM | 0 | 0 | 3 | 3 | 8 | 2 | |

Starting with a reasonable guess

- Introduce an *assumed* probability
 - A starting point when you have little evidence
 - Might choose based on some external evidence of knowledge
- Produce a weighted estimate for the conditional probability based on the assumed and the raw computed probability - e.g.:

$$p_w(f|c) = (\text{weight} * \text{assumed} + \text{count} * p_{\text{raw}}(f|c)) / (\text{count} + \text{weight})$$

where count is the number of times f occurs across all categories

- *Why is this valid?*

Computing the probability of a document

- Want to compute conditional probability of an entire document rather than a single word
- Going to make a naïve simplifying assumption
 - All features are independent of each other
 - This is clearly not actually going to be true - certain words are almost certainly likely to appear together...
 - But in practice it doesn't seem to matter & will still work even if technically incorrect!
 - Does mean that we can't use the computed probability directly however
 - will need to compare probabilities and pick most likely

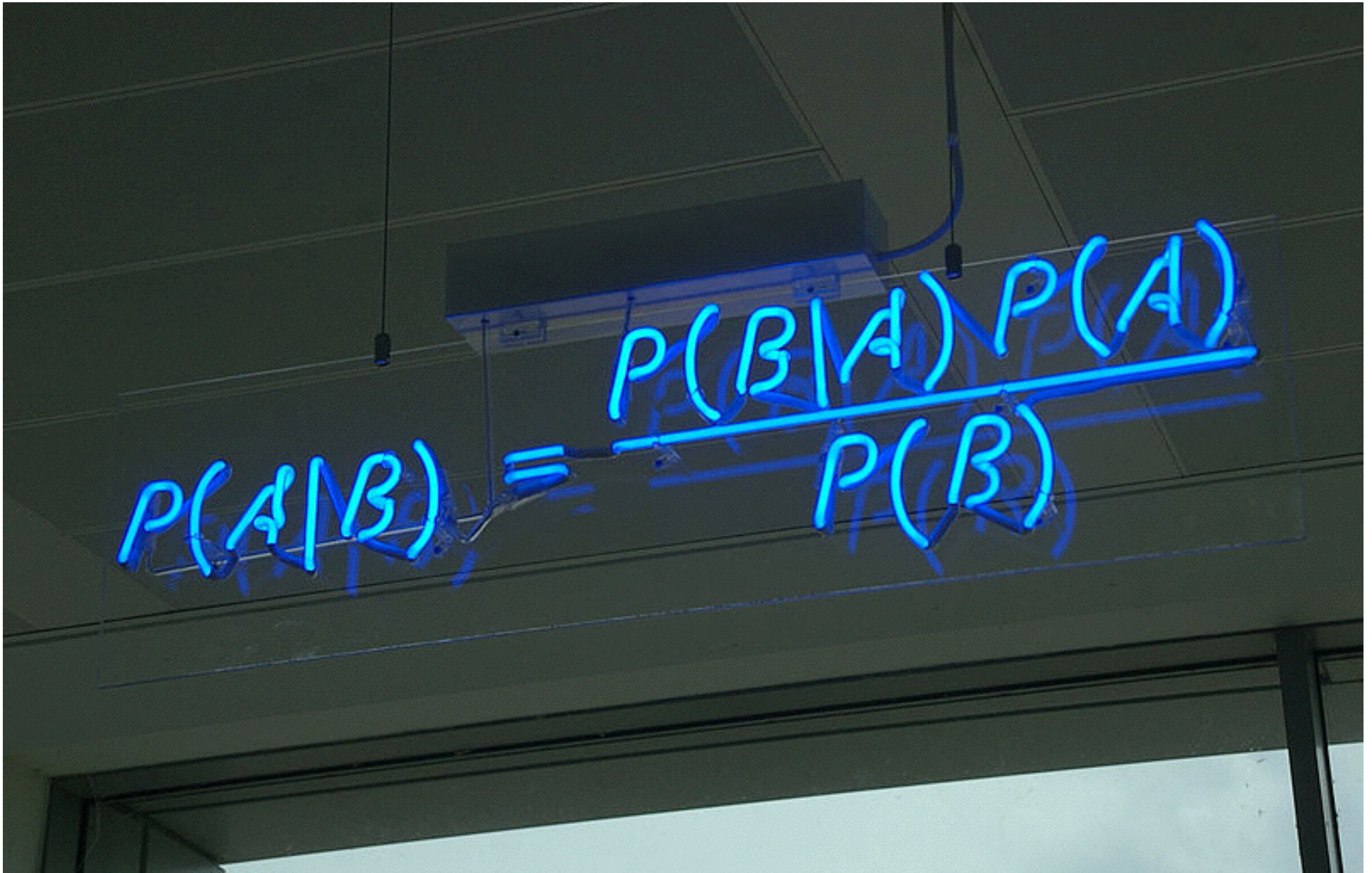
Computing the probability of a document

- The Naïve assumption allows us to express the document conditional as the product of the conditional feature probabilities:

$$p(d|c) = \prod_{f \in d} p(f|c)$$

- Often referred to as the **likelihood** of the document given category
- It's clear that $p(d|c)$ isn't actually useful by itself
 - We really want to be able to compute the probability of a class given a document $p(c|d)$
 - (aka the **posterior**)

Bayes' Theorem



A photograph of a blue neon sign mounted on a ceiling, displaying the formula for Bayes' Theorem. The sign is illuminated and shows the equation $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$. The background is a dark ceiling with visible grid lines and some wiring.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayes' Theorem

- So: $p(c|d) = p(d|c) * p(c) / p(d)$
 - $p(c)$ is the probability that a randomly selected document will be in category c
 - Often referred to as the **prior** of c
 - can be estimated empirically: the number of documents in category c divided by the total number of documents
 - or just set assuming all classes are equally probable
 - $p(d)$ is actually irrelevant if we only care about the magnitude of the probabilities as it's the same for all categories

Bayes' Theorem

Posterior probability \propto Likelihood \times Prior probability

Choosing the best category

- $p(c|d) \propto p(d|c) * p(c)$ can be calculated for all categories
- Simplest solution to assigning a category to a document is to choose the category with the **largest** $p(c|d)$
 - ***Maximum a Posteriori (MAP)***
 - This assumes that the categories are considered equal
 - In the case of SPAM/HAM this isn't true!
 - cost of misclassifying document as SPAM is considerably higher than misclassifying as HAM

Choosing the best category

- Alternative approach:
 - Compare the ratio of the $p(c|d)$ for the top-two categories against a threshold
 - e.g. for HAM/SPAM case:
`if ($p(SPAM|d)/p(HAM|d) > 3$) then SPAM else HAM`

Aside: Implementation note - small probabilities

- Warning: attempting to compute

$$p(c|d) \propto p(c) \prod_{f \in d} p(f|c)$$

directly could be a disaster!

- Many of the $p(f|c)$ would be very small
 - leading to **floating-point underflow**
- Solution - take logs:

$$\log(p(c|d)) \propto \log(p(c)) + \sum_{f \in d} \log(p(f|c))$$

Aside: A Bayesian Prior

- Where did $p_w(f|c) = (\text{weight} * \text{assumed} + \text{count} * p_{raw}(f|c)) / (\text{count} + \text{weight})$ come from?
- Bayesian statistics
 - Assumption that r.v.'s are drawn from probability distributions rather than point samples
 - In this case we're assuming the HAM/SPAM classification for a feature f is a *binomial* random variable with a *beta distribution* prior
- **Very important note: Naïve Bayes doesn't necessarily mean a Bayesian approach**

Learning to classify text: Fisher's Method

Fisher's method: Overview

- The naïve bayes method used the feature likelihoods to compute a whole document probability
- Fisher's method
 - computes the probability of each category for each feature
 - uses a statistical test to see if the set of combined probabilities is more or less likely than a random set
 - *Note: assumes independence of features*



Bayes' Theorem: Extended form

- Simple form:

$$P(A \mid B) = \frac{P(B \mid A) P(A)}{P(B)}$$

- Law of total probability:

$$P(B) = \sum_j P(B \mid A_j) P(A_j)$$

- Extended form:

$$P(A_i \mid B) = \frac{P(B \mid A_i) P(A_i)}{\sum_j P(B \mid A_j) P(A_j)}$$

Category probabilities

- Want to calculate $P(c|f)$
 - Bayes' extended form lets us do this, but need $P(c)$ for all c
 - could estimate from data
 - or could make **unbiased estimates** - all $P(c)$ equally likely
- For our SPAM/HAM classifier if there is no *a priori* reason to assume HAM over SPAM:
 $P(c) = P(HAM) = P(SPAM) = 0.5$
 - **$P(c | f) = P(f | c) / (P(f | HAM) + P(f | SPAM))$**

Combining probabilities

- Fishers' method combines k probabilities (“p-values”), $P(c=C|f_k)$, from each test into a single test statistic, X^2 :

$$X_{2k}^2 \sim -2 \sum_{i=1}^k \ln(p(c = C|f_i))$$

- If the p-values are independent this X^2 statistic will follow a *chi-squared distribution* in $2k$ degrees of freedom
- Thus we can compute a combined p-value:

$$p_C = K^{-1}\left(-2 \sum_{i=1}^k \ln(p(c = C|f_i)), 2k\right) = K^{-1}\left(-2 \ln\left(\prod_{i=1}^k p(c = C|f_i)\right), 2k\right)$$

where K^{-1} is the inverse chi-squared function

Making classifications

- Many possible approaches
- Popular one:

$$I = (1 + p_{SPAM} - p_{HAM}) / 2$$

I tends to 1 if document is SPAM and to 0 if document is HAM

Improved Text Features

- The simple BOW type features we looked at earlier are OK, but can we do better?
- Feature engineering

Fields

- Emails have structure:
 - sender; to; cc
 - title; subject; body
 - etc
- Can we use these to create features?
 - e.g. introduce the sender as a feature...

Virtual features

- SPAM emails might have language features that exhibit certain characteristics
 - THEY MIGHT CONTAIN LOTS OF SHOUTING!!!
- Could engineer a virtual feature to measure this
 - If more than 30% of words are uppercase then record the presence of a *virtual* “uppercase” feature

| | money | viagra | dad | mom | dinner | mail | v.Uppercase | ... |
|------|-------|--------|-----|-----|--------|------|-------------|-----|
| SPAM | 1 | 15 | 0 | 2 | 1 | 8 | 5 | |
| HAM | 0 | 0 | 3 | 3 | 8 | 2 | 0 | |

Word Context: N-Grams

- Rather than looking at individual words, could use pairs of words or triplets of words as the base feature
 - More generally called *n-grams*

ratherthan thanlooking lookingat atindividual individualwords

ratherthanlooking thanlookingat lookingatindividual

- Advantages: context capture; names
- Disadvantages: feature explosion

Deeper parsing of text

- Could we more intelligently parse the text to better know what features we should record?
 - *Natural Language Processing*
 - *POS-tagging*
 - *NE-Extraction*

A dog is chasing a boy on the playground

Det Noun Aux Verb Det Noun Prep Det Noun

Lexical analysis
(part-of-speech tagging)

Noun Phrase

Complex Verb

Noun Phrase

Noun Phrase

Verb Phrase

Prep Phrase

Verb Phrase

Sentence

Semantic analysis

Dog(d1).
Boy(b1).
Playground(p1).
Chasing(d1,b1,p1).

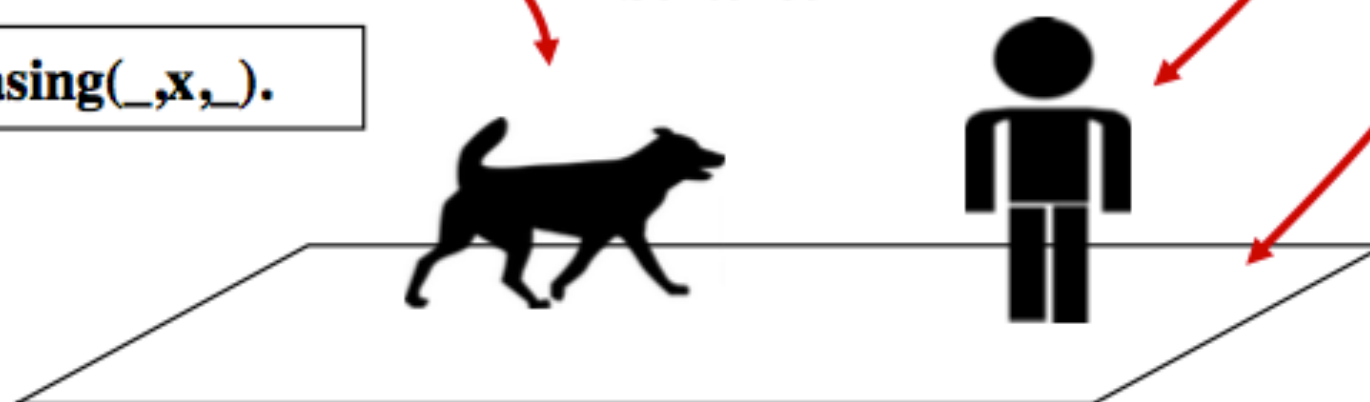
Scared(x) if Chasing(_,x,_).

Scared(b1)

Inference

Syntactic analysis
(Parsing)

A person saying this may
be reminding another person
to
get the dog back...
Pragmatic analysis
(speech act)



NLP is hard!

- POS tagging
 - *State-of-the-art: Close to 97% accuracy for English*
 - “He turned off the motorway.” vs “He turned off the PC.”
- General complete parsing
 - “A man saw a boy with a telescope.”
- **Robust NLP tends to be *shallow***

Named Entities

Jim bought 300 shares of Acme Corp. in 2006.



[Jim]_{Person} bought 300 shares of [Acme
Corp.]_{Organisation} in [2006]_{Time}.

Could we use this as a basis for better features?

Summary

- Learning models that categorise data is a core part of data mining
 - Many different supervised machine learning techniques could be used
 - For problems like SPAM/HAM probabilistic approaches are attractive
 - Easy to implement & computationally efficient
 - Directly interpretable
 - Online
 - ...but independence assumption of features can be a problem
- Irrespective of classification technique, choice of features is key