COM4509/6509
Machine Learning and Adaptive Intelligence
Lecture 3: Decision Trees & Ensemble Methods

Mike Smith* and Matt Ellis

*m.t.smith@sheffield.ac.uk

Decision Trees

Detour: Entropy & Information Gain

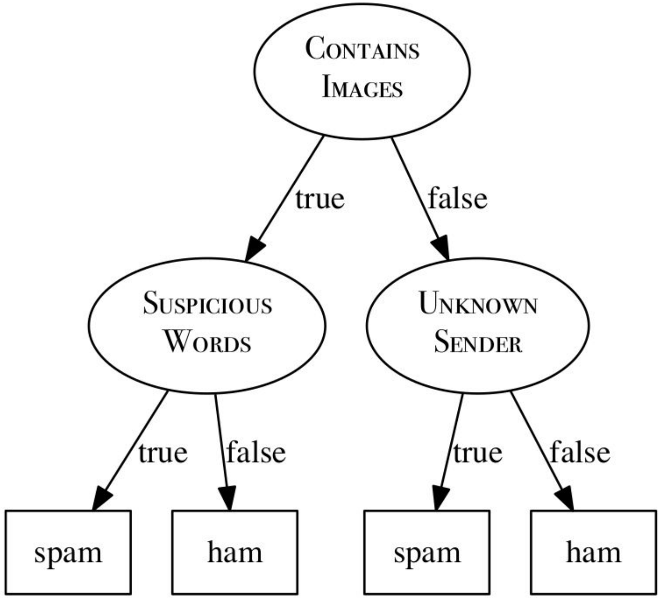
Building trees

Details about decision trees

Ensembles

Decision Trees

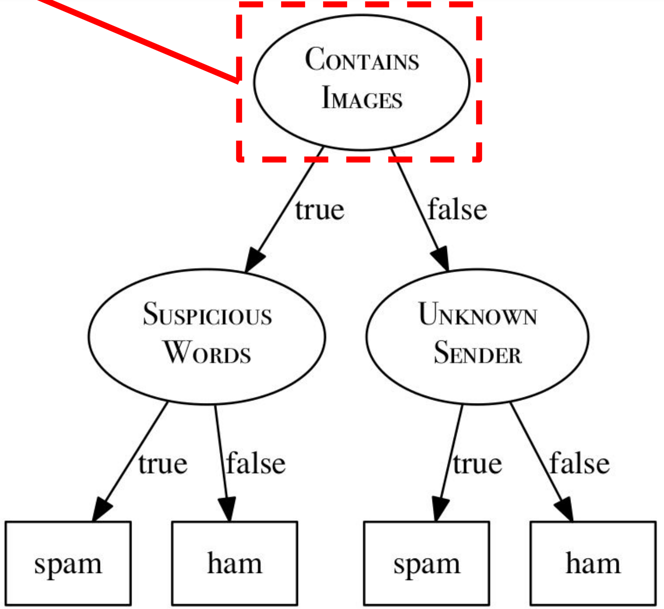
Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam



Decision Trees

- A decision tree has a **root node**,

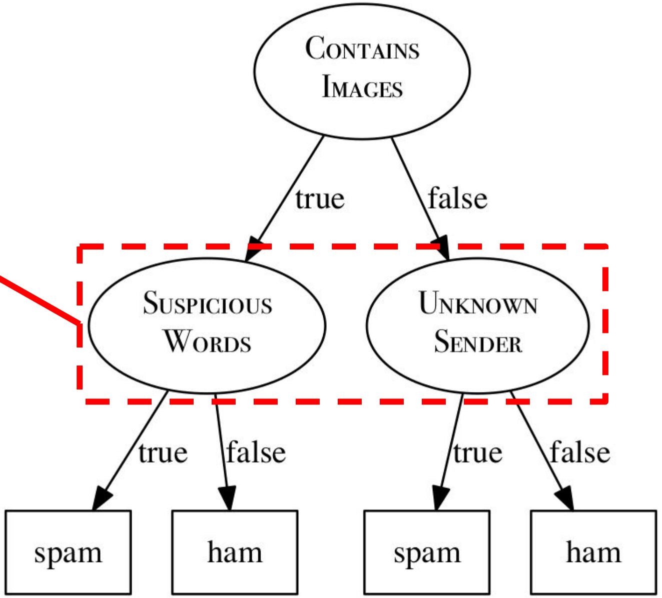
Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam



Decision Trees

- A decision tree has a **root node**,
- **interior nodes**,

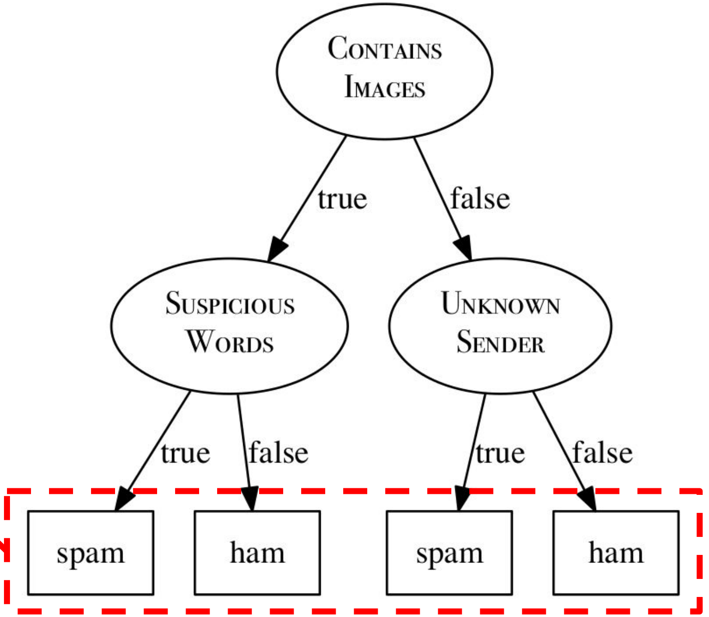
Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam



Decision Trees

- A decision tree has a **root node**,
- **interior nodes**,
- and **leaf nodes**.

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam



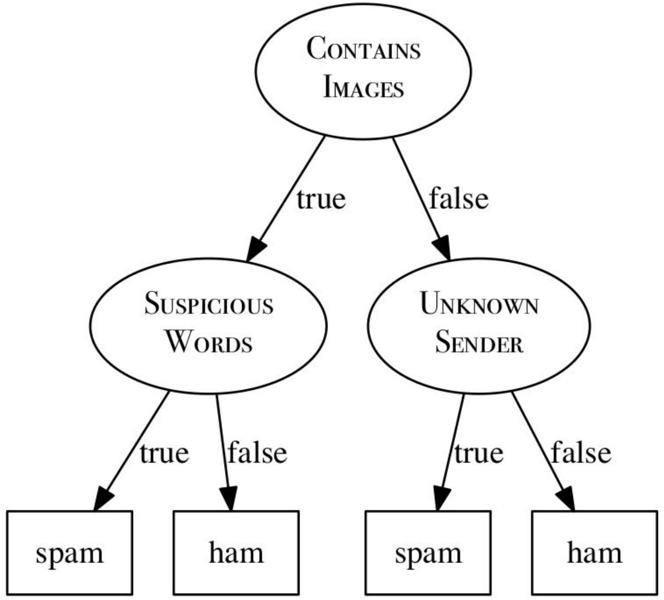
Decision Trees

- A decision tree has a **root node**,
- **interior nodes**,
- and **leaf nodes**.

The leaf nodes are predictions.

The rest of the nodes are (usually binary) decisions.

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam



Decision Trees

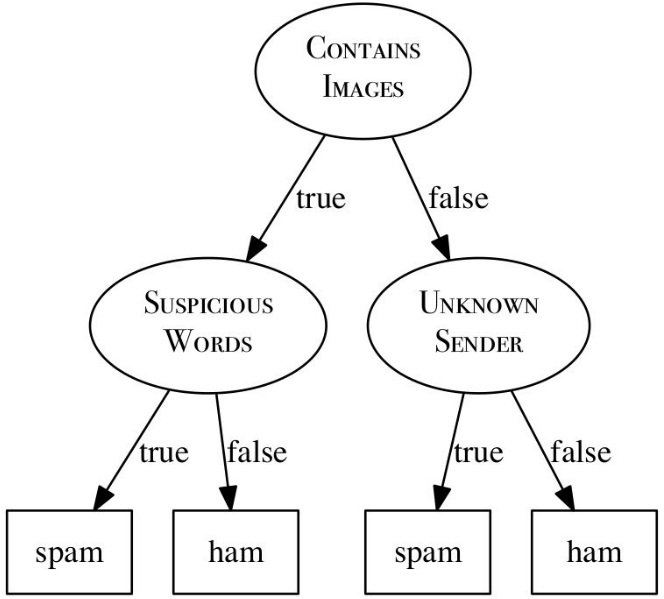
- A decision tree has a **root node**,
- **interior nodes**,
- and **leaf nodes**.

The leaf nodes are predictions.

The rest of the nodes are (usually binary) decisions.

ACTIVITY: Draw out an alternative decision tree to classify spam and ham. Can you make the tree smaller?

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam



Decision Trees

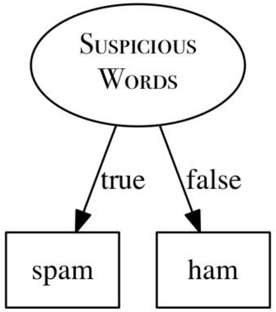
- A decision tree has a **root node**,
- **interior nodes**,
- and **leaf nodes**.

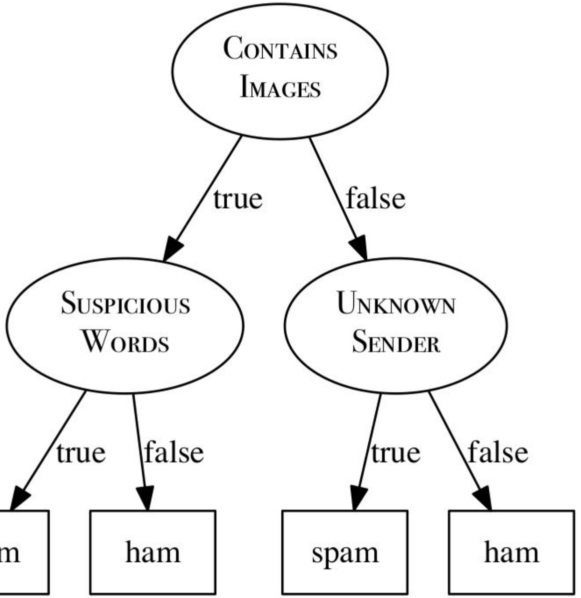
The leaf nodes are predictions.

The rest of the nodes are (usually binary) decisions.

ACTIVITY: Draw out an alternative decision tree to classify spam and ham. Can you make the tree smaller?

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

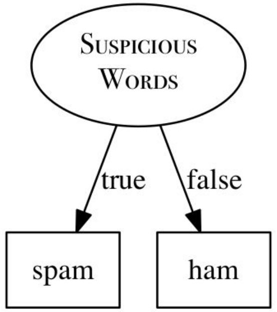


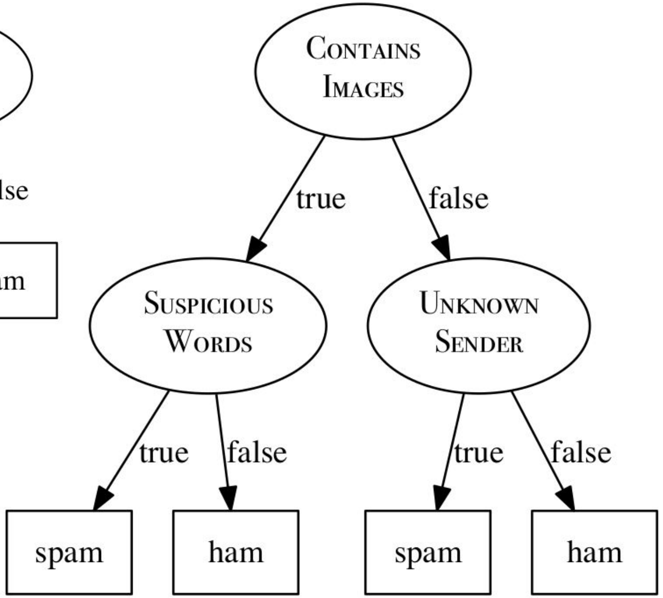


Decision Trees

- We want to use the most informative features first.
- These are the features that split the data into sets that are as 'pure' as possible.

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

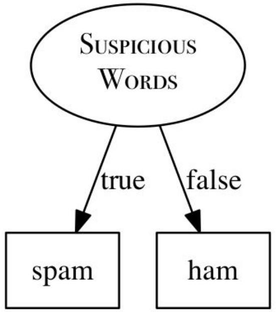


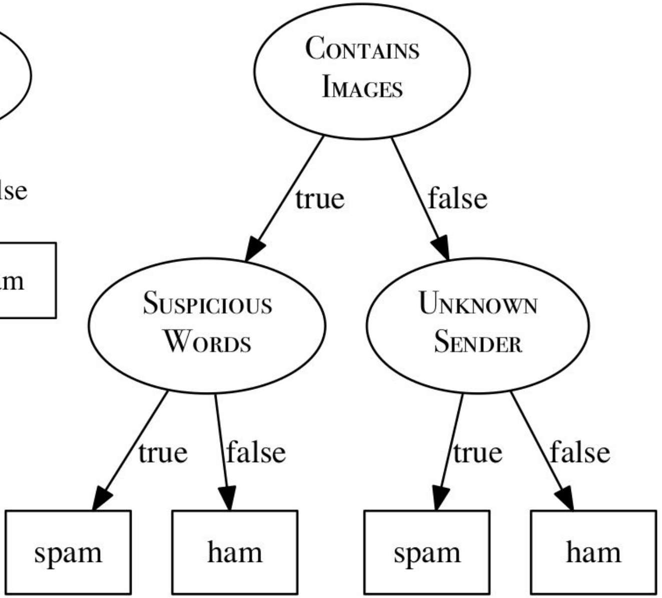


Decision Trees

- We want to use the most informative features first.
- These are the features that split the data into sets that are as 'pure' as possible.
- We need a way of assessing which features give us the most information about the class.

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

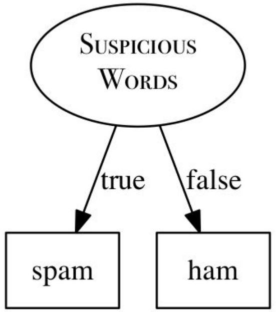


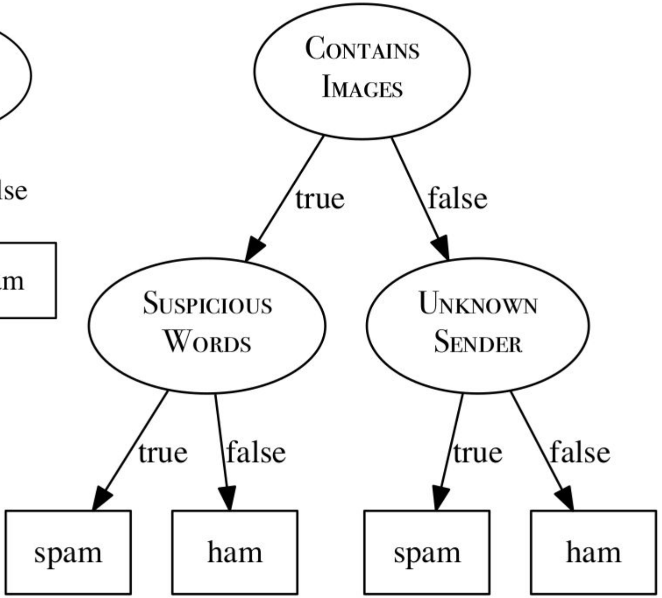


Decision Trees

- We want to use the most informative features first.
- These are the features that split the data into sets that are as ‘pure’ as possible.
- We need a way of assessing which features give us the most information about the class.
- Another way of thinking about this is how much less surprised we will be about an outcome, if we know that feature’s value.

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam





Decision Trees

Detour: Entropy & Information Gain

Building trees

Details about decision trees

Ensembles

- Boosting
- Bootstrap
- Bagging
- Random Forest

Quick reminder…

Quick Reminders!

The expectation of a function of a random variable:

$$E[g(X)] = \sum_{i=1}^{n} g(x_i) P(X = x_i)$$

The log of a product is the sum of the logs.

$$\log(A \times B) = \log A + \log B$$

Detour: Entropy and Information Gain

- How surprised should we expect to be about the outcome of an event?

Detour: Entropy and Information Gain

- How surprised should we expect to be about the outcome of an event?
- We need to pick a function, g( ) to measure our surprise about a particular outcome. But what function?

Detour: Entropy and Information Gain

- How surprised should we expect to be about the outcome of an event?
- We need to pick a function, g( ) to measure our surprise about a particular outcome. But what function?
- First: We want surprises to add.

Detour: Entropy and Information Gain

- How surprised should we expect to be about the outcome of an event?
- We need to pick a function, g( ) to measure our surprise about a particular outcome. But what function?
- First: We want surprises to add.
    - The joint probability of two independent events (A and B) happening is

      P(A=a) x P(B=b).

Detour: Entropy and Information Gain

- How surprised should we expect to be about the outcome of an event?
- We need to pick a function, g( ) to measure our surprise about a particular outcome. But what function?
- First: We want surprises to add.
    - The joint probability of two independent events (A and B) happening is

      P(A=a) x P(B=b).
    - The surprises individually are: g(A=a) and g(B=b).
    - We want the surprise of both to be: g(A=a) + g(B=b)

Detour: Entropy and Information Gain

- How surprised should we expect to be about the outcome of an event?
- We need to pick a function, g( ) to measure our surprise about a particular outcome. But what function?
- First: We want surprises to add.
  - The joint probability of two independent events (A and B) happening is
    P(A=a) x P(B=b).
  - The surprises individually are: g(A=a) and g(B=b).
  - We want the surprise of both to be: g(A=a) + g(B=b)
- What function turns the product into a sum? (hint: see last slide)

Detour: Entropy and Information Gain

- How surprised should we expect to be about the outcome of an event?
- We need to pick a function, g( ) to measure our surprise about a particular outcome. But what function?
- First: We want surprises to add.
    - The joint probability of two independent events (A and B) happening is
      P(A=a) x P(B=b).
    - The surprises individually are: g(A=a) and g(B=b).
    - We want the surprise of both to be: g(A=a) + g(B=b)
- What function turns the product into a sum? (hint: see last slide)
- Maybe the log?
- Could we use g(x) = log P(x)? Hmm…

$$\log\Big(P(A=a)\times P(B=b)\Big) = \log\Big(P(A=a)\Big) + \log\Big(P(B=b)\Big)$$

Detour: Entropy and Information Gain

- So we are thinking that the log of the probability of an event might be a good measure of surprise…
- If something is certain (P=1), then it’s not surprising at all. So its surprise is zero. So that’s good. log(1)=0.

Detour: Entropy and Information Gain

- So we are thinking that the log of the probability of an event might be a good measure of surprise…
- If something is certain (P=1), then it’s not surprising at all. So its surprise is zero. So that’s good. log(1)=0.
- Problem: If an outcome is unlikely, then it’s surprising if it happens, *but the log of a small number is negative*.

Detour: Entropy and Information Gain

- So we are thinking that the log of the probability of an event might be a good measure of surprise…
- If something is certain (P=1), then it's not surprising at all. So its surprise is zero. So that's good. log(1)=0.
- Problem: If an outcome is unlikely, then it's surprising if it happens, *but the log of a small number is negative*.
  - E.g. $\log_e 0.001 = -6.9$. We want our surprise to be BIGGER if the probability of an event is small.

Detour: Entropy and Information Gain

- So we are thinking that the log of the probability of an event might be a good measure of surprise…
- If something is certain (P=1), then it's not surprising at all. So its surprise is zero. So that's good. log(1)=0.
- Problem: If an outcome is unlikely, then it's surprising if it happens, *but the log of a small number is negative*.
    - E.g. $\log_e 0.001 = -6.9$. We want our surprise to be BIGGER if the probability of an event is small.
- We could use the NEGATIVE LOG probability to measure surprise instead?

Detour: Entropy and Information Gain

- So we are thinking that the log of the probability of an event might be a good measure of surprise…
- If something is certain (P=1), then it’s not surprising at all. So its surprise is zero. So that’s good. log(1)=0.
- Problem: If an outcome is unlikely, then it’s surprising if it happens, *but the log of a small number is negative*.
  - E.g. $\log_e 0.001 = -6.9$. We want our surprise to be BIGGER if the probability of an event is small.
- We could use the NEGATIVE LOG probability to measure surprise instead?

This is entropy: the expected negative log probability

$$E[-\log p(X)] = -\sum_{i=1}^{N} p(x) \log p(x)$$

Detour: Entropy and Information Gain

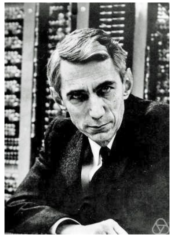
$$E[-\log p(X)] = -\sum_{i=1}^{N} p(x) \log p(x)$$

For intuition, let's look at some examples.

To start, compute (in bits/shannons) the entropy of the outcome of a single fair coin toss.

Note: The base of the log decides what units we measure the entropy in:

Base 2 - bits [or shannons]
Base e - nats
Base 10 - hartleys



Detour: Entropy and Information Gain

$$E[-\log p(X)] = -\sum_{i=1}^{N} p(x) \log p(x)$$

For intuition, let's look at some examples.

To start, compute (in bits/shannons) the entropy of the outcome of a single fair coin toss.

p(heads) = ½
p(tails) = ½

Reminder. To work out $\log_2$: What power do you need to raise 2 to, to get ½?

Detour: Entropy and Information Gain

$$E[-\log p(X)] = -\sum_{i=1}^{N} p(x) \log p(x)$$

For intuition, let's look at some examples.

To start, compute (in bits/shannons) the entropy of the outcome of a single fair coin toss.

p(heads) = ½
p(tails) = ½

$-(\frac{1}{2} \times \log_2(\frac{1}{2}) + \frac{1}{2} \times \log_2(\frac{1}{2}))$

$-(\frac{1}{2} \times -1 + \frac{1}{2} \times -1) = -(-\frac{1}{2} + -\frac{1}{2}) = -(-1) = 1$

Reminder. To work out $\log_2$: What power do you need to raise 2 to, to get ½?

So a single fair coin toss has 1 bit of entropy.

Detour: Entropy and Information Gain

If we use base 2 in our log, then we are measuring entropy in ‘bits’ or ‘shannons’.
If we use base e then we are measuring entropy in ‘nats’,
If we use base 10 then we are measuring entropy in Hartleys’.

Detour: Entropy and Information Gain

$$E[-\log p(X)] = -\sum_{i=1}^{N} p(x)\log p(x)$$

What about a coin that lands heads 80% of the time?

P(heads) = 0.8

P(tails) = 0.2

Detour: Entropy and Information Gain

$$E[-\log p(X)] = -\sum_{i=1}^{N} p(x) \log p(x)$$

What about a coin that lands heads 80% of the time?

P(heads) = 0.8　　　　P(tails) = 0.2

$-(0.8 \times \log_2(0.8) + 0.2 \times \log_2(0.2)) = 0.72$ bits

A bias coin toss has less entropy
(the expected surprise is less).

Detour: Entropy and Information Gain

$$E[-\log p(X)] = -\sum_{i=1}^{N} p(x)\log p(x)$$

We toss the bias coin 4 times. How many bits do we need to store the outcome? Can you think of an example encoding to demonstrate that we need fewer than 4 bits?

By this, I mean a way of writing down the result of the four coin tosses that, on average, will use fewer than 4 ones-and-zeros.

ACTIVITY
Remember,
P(heads) = 0.8
P(tails) = 0.2

Detour: Entropy and Information Gain

$$E[-\log p(X)] = -\sum_{i=1}^{N} p(x) \log p(x)$$

We toss the bias coin 4 times. How many bits do we need to store the outcome? Can you think of an example encoding to demonstrate that we need fewer than 4 bits?

First: The actual entropy of the four tosses (assuming independence) is 4×0.72 = 2.88 bits.

One could check this by computing the entropy over the distribution of the 16 possible outcomes:
-(P(H,H,H,H) log(P(H,H,H,H))+ P(H,H,H,T) log(P(H,H,H,T))+…etc)

Detour: Entropy and Information Gain

$$E[-\log p(X)] = -\sum_{i=1}^{N} p(x) \log p(x)$$

We toss the bias coin 4 times. How many bits do we need to store the outcome? Can you think of an example encoding to demonstrate that we need fewer than 4 bits?

First: The actual entropy of the four tosses (assuming independence) is 4×0.72 = 2.88 bits.

A simple encoding could be: If all heads just send a ‘1’; otherwise, send a ‘0’ and then four more bits to record the sequence (e.g. H=1, T=0).

E.g. H H H H -> [1] [need just one bit]
H T H H -> [0, 1, 0, 1, 1] [need five bits]
T H T H -> [0, 0, 1, 0, 1] [need five bits]

Detour: Entropy and Information Gain

$$E[-\log p(X)] = -\sum_{i=1}^{N} p(x) \log p(x)$$

We toss the bias coin 4 times. How many bits do we need to store the outcome? Can you think of an example encoding to demonstrate that we need fewer than 4 bits?

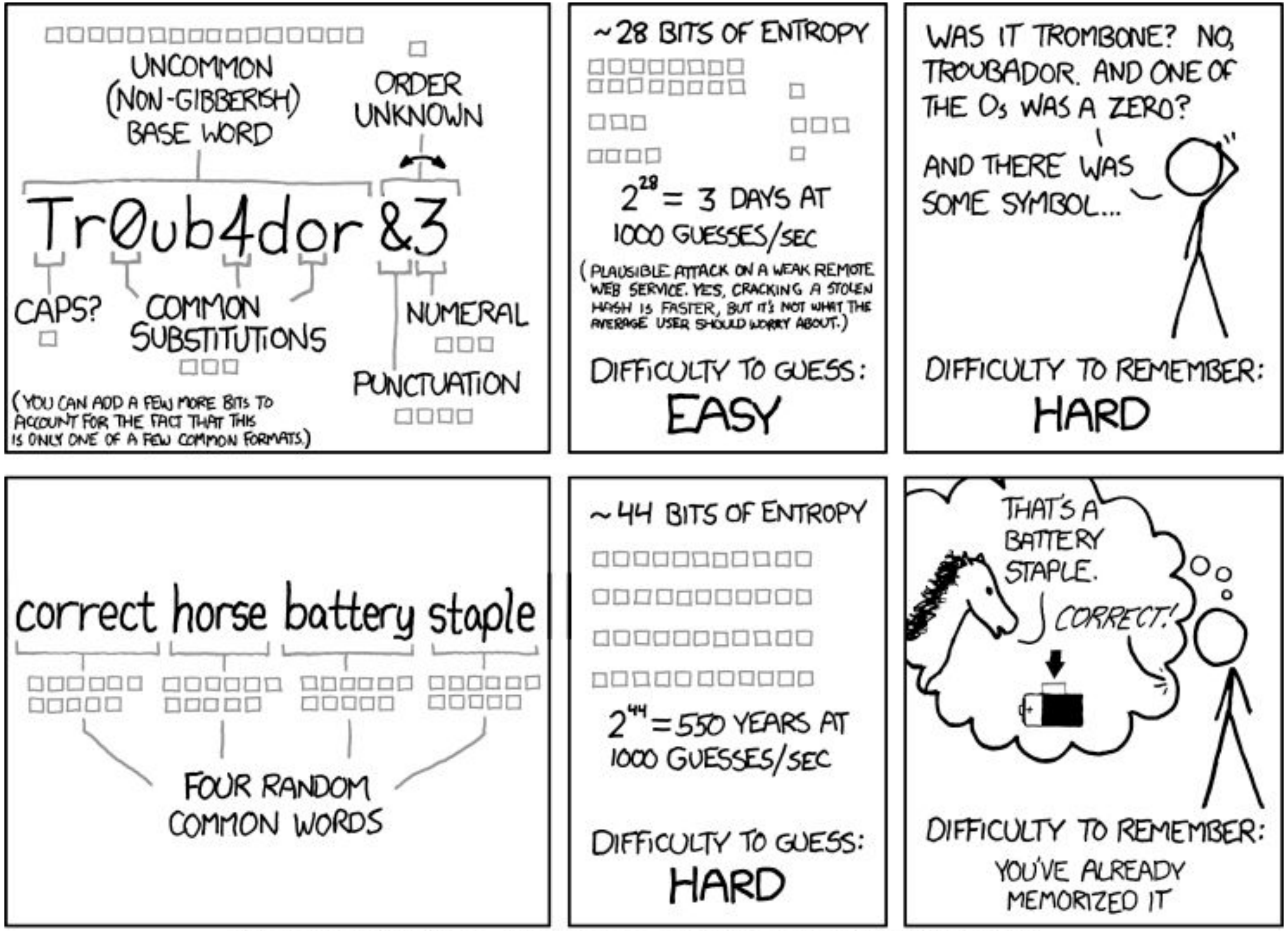
First: The actual entropy of the four tosses (assuming independence) is 4×0.72 = 2.88 bits.

A simple encoding could be: If all heads just send a ‘1’; otherwise, send a ‘0’ and then four more bits to record the sequence (e.g. H=1, T=0).

E.g. H H H H -> [1] [need just one bit]
H T H H -> [0, 1, 0, 1, 1] [need five bits]
T H T H -> [0, 0, 1, 0, 1] [need five bits]

What’s the **expected** number of bits needed? The probability of all heads is $0.8^4 = 0.41$. So the expected number of bits is 0.41 × 1 + (1-0.41) × 5 = 3.36

On average we only need 3.36 bits.


UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor&3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)
~28 BITS OF ENTROPY
2^28 = 3 DAYS AT 1000 GUESSES/SEC
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)
DIFFICULTY TO GUESS: EASY
WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 GUESSES/SEC
DIFFICULTY TO GUESS: HARD
THAT'S A BATTERY STAPLE.
CORRECT!
DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT

THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

Detour: Entropy and Information Gain

$$E[-\log p(X)] = -\sum_{i=1}^{N} p(x) \log p(x)$$

Final example.

- You need to store a six letter english word. How many bits will you need? First: What's the naive answer? Can you be more efficient?
    - Think: Are letters independent? Are all combinations valid words?

Hint (26 letters in alphabet)

Hint (22k 6-letter words)

ACTIVITY

Detour: Entropy and Information Gain

$$E[-\log p(X)] = -\sum_{i=1}^{N} p(x) \log p(x)$$

Final example.

- You need to store a six letter english word. How many bits will you need? First: What’s the naive answer? Can you be more efficient?
    - Think: Are letters independent? Are all combinations valid words?

- Naive answer: 26 letters: $26^6 \approx 2^{28}$. So we need about 28 bits.
- We could store letter pairs, but more straightforward:
- 22,000 6-letter words in english. $\log_2(22{,}000) \approx 14$. So we really only need 14 bits!

Detour: Entropy and Information

- Our expected surprise (entropy) about an outcome might be reduced if we observe something:
    - Whether it's raining, if we recently observed dark clouds:
    - If P(Weather=rain) = 0.6 then H(Weather) = 0.97 bits

Detour: Entropy and Information

- Our expected surprise (entropy) about an outcome might be reduced if we observe something:
  - Whether it's raining, if we recently observed dark clouds:
  - If P(Weather=rain) = 0.6
    then H(Weather) = 0.97 bits
  - If P(Weather=rain|Sky=cloudy) = 0.9
    then H(Weather|Sky=cloudy) = 0.46 bits.

Detour: Entropy and Information

- Our expected surprise (entropy) about an outcome might be reduced if we observe something:
  - Whether it's raining, if we recently observed dark clouds:
  - If P(Weather=rain) = 0.6 then H(Weather) = 0.97 bits
  - If P(Weather=rain|Sky=cloudy) = 0.9 then H(Weather|Sky=cloudy) = 0.46 bits.
- We have gained some information (which has reduced our entropy, or "surprise" when it rains).

Detour: Entropy and Information Gain

- But we won’t always observe it being cloudy, sometimes we will see a clear sky. We need the expected entropy across all outcomes:
    - H(Weather|Sky) = p(Sky=cloudy) H(Weather|Sky=cloudy) + p(Sky=clear) H(Weather|Sky=clear)

Detour: Entropy and Information Gain

- But we won't always observe it being cloudy, sometimes we will see a clear sky. We need the expected entropy across all outcomes:
    - E[H(Weather|Sky)] = p(Sky=cloudy) H(Weather|Sky=cloudy) + p(Sky=clear) H(Weather|Sky=clear)
- The Information Gain is the expected difference between the entropy with and without this observation.

$$\boxed{IG(S,A) = H(S) - E[H(S|A)]} = H(S) - \sum_a p(A=a)H(S|A=a)$$

SORRY! Forgot the expected in the equation :(

Detour: Entropy and Information Gain

- But we won't always observe it being cloudy, sometimes we will see a clear sky. We need the expected entropy across all outcomes:
  - E[H(Weather|Sky)] = p(Sky=cloudy) H(Weather|Sky=cloudy) + p(Sky=clear) H(Weather|Sky=clear)
- The Information Gain is the expected difference between the entropy with and without this observation.

$$\boxed{IG(S,A) = H(S) - E[H(S|A)]} = H(S) - \sum_a p(A=a) H(S|A=a)$$

side note

This is a particular case of the Kullback–Leibler divergence (a measure of how similar distributions are - used in variational inference)

Decision Trees

Detour: Entropy & Information Gain

Building trees

Details about decision trees

Ensembles

Building Decision Trees

- Spam data: p(Class=spam) = 3/7. H(Class)=0.99 bits
- How much would expected entropy fall if we find out whether the email has images?

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

Building Decision Trees

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

- Spam data: p(Class=spam) = 3/7. H(Class)=0.99 bits
- How much would expected entropy fall if we find out whether the email has images?
- If we observe it does have images:
    - Using p(spam|images) = ⅔ → H(spam|images) = (⅔ × $\log_2$ ⅔) + (⅓ × $\log_2$ ⅓)

$$E[-\log p(X)] = -\sum_{i=1}^{N} p(x) \log p(x)$$

Building Decision Trees

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

- Spam data: p(Class=spam) = 3/7. H(Class)=0.99 bits
- How much would expected entropy fall if we find out whether the email has images?
- If we observe it does have images:

$$H(spam|images) = 0.92$$

Building Decision Trees

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

- Spam data: p(Class=spam) = 3/7. H(Class)=0.99 bits
- How much would expected entropy fall if we find out whether the email has images?
- If we observe it does have images:

  H(spam|images) = 0.92
- To compute the Information Gain we need to also compute the case without images:
  - p(spam|no images) = ¼ → H(spam|no images)

Building Decision Trees

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

- Spam data: p(Class=spam) = 3/7. H(Class)=0.99 bits
- How much would expected entropy fall if we find out whether the email has images?
- If we observe it does have images:

  H(spam|images) = 0.92
- To compute the Information Gain we need to also compute the case without images:
  - p(spam|no images) = ¼ → H(spam|no images) = $(\frac{1}{4} \times \log_2 \frac{1}{4}) + (\frac{3}{4} \times \log_2 \frac{3}{4})$

Building Decision Trees

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

- Spam data: p(Class=spam) = 3/7. H(Class)=0.99 bits
- How much would expected entropy fall if we find out whether the email has images?
- If we observe it does have images:

  H(spam|images) = 0.92
- To compute the Information Gain we need to also compute the case without images:

  H(spam|no images) = 0.81

Building Decision Trees

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

- Spam data: p(Class=spam) = 3/7. H(Class)=0.99 bits
- How much would expected entropy fall if we find out whether the email has images?
- If we observe it does have images:

  H(spam|images) = 0.92
- To compute the Information Gain we need to also compute the case without images:

  H(spam|no images) = 0.81
- We need to compute the expectation.
  - The probability of having images is 3/7…

Building Decision Trees

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

- Spam data: p(Class=spam) = 3/7. H(Class)=0.99 bits
- How much would expected entropy fall if we find out whether the email has images?
- If we observe it does have images:

  H(spam|images) = 0.92
- To compute the Information Gain we need to also compute the case without images:

  H(spam|no images) = 0.81
- We need to compute the expectation.
  - The probability of having images is 3/7. The expected entropy is computed by:
- E[H(Class|Images)] = p(images)×H(spam|images)+p(no image)×H(spam|no images)

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

Building Decision Trees

- Spam data: p(Class=spam) = 3/7. H(Class)=0.99 bits
- How much would expected entropy fall if we find out whether the email has images?
- If we observe it does have images:

  H(spam|images) = 0.92
- To compute the Information Gain we need to also compute the case without images:

  H(spam|no images) = 0.81
- We need to compute the expectation.
  - The probability of having images is 3/7. The expected entropy is computed by:
- E[H(Class|Images)] = 3/7 × 0.92 + 4/7 × 0.81

Building Decision Trees

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

- Spam data: p(Class=spam) = 3/7. H(Class)=0.99 bits
- How much would expected entropy fall if we find out whether the email has images?
- If we observe it does have images:

  H(spam|images) = 0.92
- To compute the Information Gain we need to also compute the case without images:

  H(spam|no images) = 0.81
- We need to compute the expectation.
  - The probability of having images is 3/7. The expected entropy is computed by:
- E[H(Class|Images)] = 0.86

Building Decision Trees

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

- Spam data: p(Class=spam) = 3/7. H(Class)=0.99 bits
- How much would expected entropy fall if we find out whether the email has images?
- If we observe it does have images:

  H(spam|images) = 0.92
- To compute the Information Gain we need to also compute the case without images:

  H(spam|no images) = 0.81
- We need to compute the expectation.
  - The probability of having images is 3/7. The expected entropy is computed by:
- E[H(Class|Images)] = 0.86
- The information gain:
  - H(Class) - E[H(Class|Images)] = 0.99-0.86 = 0.13 bits.

Building Decision Trees

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

- Spam data: p(Class=spam) = 3/7. H(Class)=0.99 bits
- How much would expected entropy fall if we find out whether the email has images?
- If we observe it does have images:

  H(spam|images) = 0.92
- To compute the Information Gain we need to also compute the case without images:

  H(spam|no images) = 0.81
- We need to compute the expectation.
  - The probability of having images is 3/7. The expected entropy is computed by:
- E[H(Class|Images)] = 0.86
- The information gain:
  - H(Class) - E[H(Class|Images)] = 0.99-0.86 = 0.13 bits.
- So knowing whether or not there are images will, on average, give us 0.13 bits of information.

Building Decision Trees

ACTIVITY

$$x \log x \to 0, \text{as } x \to 0$$

$$E[-\log p(X)] = -\sum_{i=1}^{N} p(x) \log p(x)$$

What is the information gain if we observe whether or not it's an unknown sender?

Entropy of class S without observing 'A'

Expected entropy of S after observing 'A'.

Expected entropy of S after observing 'A'.

$$IG(S,A) = \boxed{H(S)} - \boxed{E[H(S|A)]} = H(S) - \boxed{\sum_a p(A=a) \boxed{H(S|A=a)}}$$

Entropy of S after observing A=a.

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

Building Decision Trees

What is the information gain if we observe whether or not it’s an unknown sender?

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

Building Decision Trees

$x \log x \to 0$, as $x \to 0$

What is the information gain if we observe whether or not it's an unknown sender?

H(Class) - E[H(Class|UnknownSender] = ~~0.99-0.23 = 0.76 bits~~ = 0.99 - 0.52 = 0.47 bits.

Unknown sender is a better choice of feature to use to split the data (than Contains Images).

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

Correction! Thanks to Chengsong Lu for spotting this. The correct answer is that the expected entropy is 0.52, so the Information gain is about 0.47.

The mistake was in my python code: I accidentally added 2/7 twice.

```
>>> (5/7)*(0.2*np.log2(0.2) + 0.8*np.log2(0.8))~~+(2/7)~~ + (2/7)*(0*np.log2(0) + 1*np.log2(1))
-0.2299
```

Decision Trees

Detour: Entropy & Information Gain

Building trees

Details about decision trees

- ID3
- Purity
- Other algorithms
- Pruning

Ensembles

Decision Trees (ID3)

The ID3 (Iterative Dichotomiser 3) algorithm builds the decision tree by iteratively selecting the feature that leads to the greatest information gain.

The ID3 Algorithm

```
function ID3(data,labels,attributes):
  -  Create a root node for the (sub)tree
  -  If the labels of the data are all the same, or attributes is empty:
         Return a leaf node with majority label
  -  Pick attribute (A) that maximises Information Gain, make this how nodes splits data.
  -  For each possible value, v of the attribute A:
         Add a new tree branch, corresponding to A=v.
         If no data has A=v:
             Return a leaf node with most common label in data
         Else:
             Return ID3(data[A=v], labels[A=v], Attributes – {A})
  -  End
```

The ID3 Algorithm

```
function ID3(data,labels,attributes):
  -   Create a root node for the (sub)tree
  -   If the labels of the data are all the same, or attributes is empty:
            Return a leaf node with majority label
  -   Pick attribute (A) that maximises Information Gain, make this how nodes splits data.
  -   For each possible value, v of the attribute A:
            Add a new tree branch, corresponding to A=v.
            If no data has A=v:
                  Return a leaf node with most common label in data
            Else:
                  Return ID3(data[A=v], labels[A=v], Attributes – {A})
  -   End
```

The ID3 Algorithm

```
function ID3(data,labels,attributes):
  -   Create a root node for the (sub)tree
  -   If the labels of the data are all the same, or attributes is empty:
          Return a leaf node with majority label
  -   Pick attribute (A) that maximises Information Gain, make this how nodes splits data.
  -   For each possible value, v of the attribute A:
          Add a new tree branch, corresponding to A=v.
          If no data has A=v:
              Return a leaf node with most common label in data
          Else:
              Return ID3(data[A=v], labels[A=v], Attributes – {A})
  -   End
```

The ID3 Algorithm

```
function ID3(data,labels,attributes):
  -  Create a root node for the (sub)tree
  -  If the labels of the data are all the same, or attributes is empty:
         Return a leaf node with majority label
  -  Pick attribute (A) that maximises Information Gain, make this how nodes splits data.
  -  For each possible value, v of the attribute A:
         Add a new tree branch, corresponding to A=v.
         If no data has A=v:
             Return a leaf node with most common label in data
         Else:
             Return ID3(data[A=v], labels[A=v], Attributes – {A})
  -  End
```

Return the subtree computed by recursively calling this method on the subset of data in which attribute A=v, with the attribute excluded from the list.

The ID3 Algorithm

```
function ID3(data,labels,attributes):
  -   Create a root node for the (sub)tree
  -   If the labels of the data are all the same, or attributes is empty:
          Return a leaf node with majority label
  -   Pick attribute (A) that maximises Information Gain, make this how nodes splits data.
  -   For each possible value, v of the attribute A:
          Add a new tree branch, corresponding to A=v.
          If no data has A=v:
              Return a leaf node with most common label in data
          Else:
              Return ID3(data[A=v], labels[A=v], Attributes – {A})
  -   End
```

Return the subtree computed by recursively calling this method on the subset of data in which attribute A=v, with the attribute excluded from the list.

Information Gain Ratio

-

The ID3 Algorithm

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	tru	true	spam
false	fal	[illegible]	ham
false	fa	[illegible]	ham
false	false	[illegible]	ham
false	false	[illegible]	ham
true	true	false	spam

HOMEWORK: Apply to spam example

```
function ID3(data,labels,attributes):
    -   Create a root node for the (sub)t
    -   If the labels of the data are all th
            Return a leaf node with ma
    -   Pick attribute (A) that maximises
    -   For each possible value, v of the attribute A:
            Add a new tree branch, corresponding to A=v.
            If no data has A=v:
                Return a leaf node with most common label in data
            Else:
                Return ID3(data[A=v], labels[A=v], Attributes – {A})
    -   End
```

Return the subtree computed by recursively calling this method on the subset of data in which attribute A=v, with the attribute excluded from the list.

Other Choices of Purity

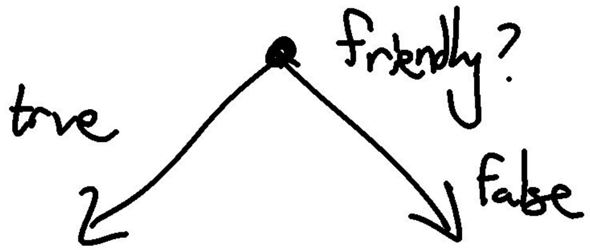
- Computing Purity: ID3 & C4.5 use Information Gain. CART uses the Gini index.
    - Gini index: subtract the sum of squared probabilities of each class from one.

$$Gini\,(t, \mathcal{D}) = 1 - \sum_{l \in levels(t)} P(t = l)^2$$

-
    - The Gini index can be thought of as calculating how often you would misclassify an instance in the dataset if you classified it based on the probability distribution of the target feature in the dataset (from Mauricio)

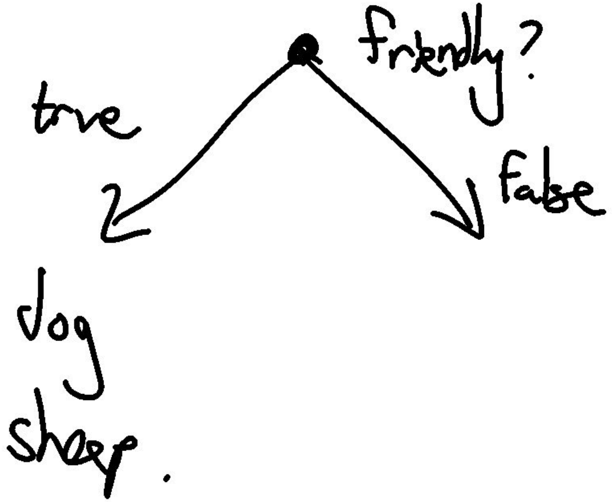
    - Gini index favours larger partitions (distributions) and is very easy to implement whereas information gain supports smaller partitions (distributions) with various distinct values. (from www.analyticssteps.com)

Gini Purity Index



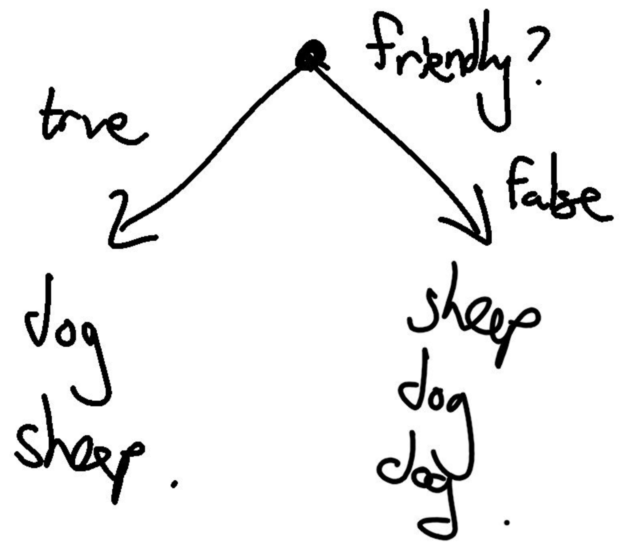
fluffy	big	friendly	animal
t	t	f	sheep
t	f	t	dog
t	f	f	dog
f	t	f	dog
t	t	t	sheep.

Gini Purity Index



fluffy	big	friendly	animal
t	t	f	sheep
t	f	t	dog
t	f	f	dog
f	t	f	dog
t	t	t	sheep.

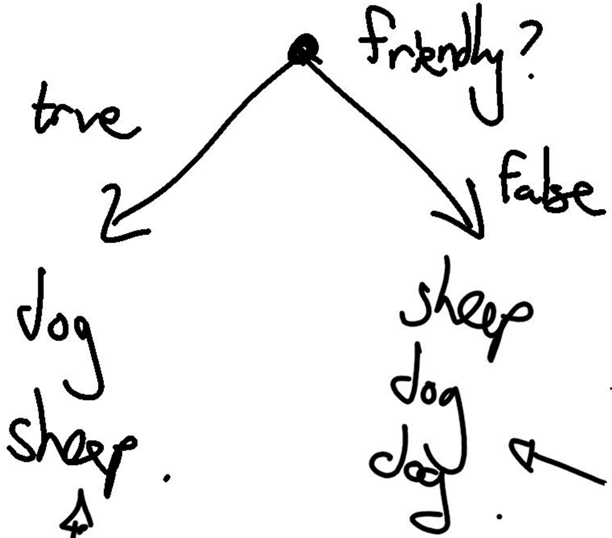
Gini Purity Index



fluffy	big	friendly	animal
t	t	f	sheep
t	f	t	dog
t	f	f	dog
f	t	f	dog
t	t	t	sheep.

Gini Purity Index

fluffy	big	friendly	animal
t	t	f	sheep
t	f	t	dog
t	f	f	dog
f	t	f	dog
t	t	t	sheep.

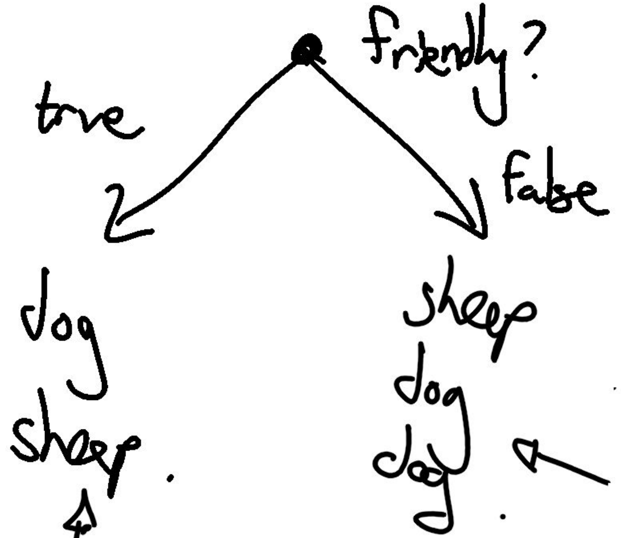


$$1-\left[\left(\frac{1}{3}\right)^2+\left(\frac{2}{3}\right)^2\right]=1-\frac{5}{9}=\frac{4}{9}$$

$$1-\left[\left(\frac{1}{2}\right)^2+\left(\frac{1}{2}\right)^2\right]=1-\frac{1}{2}=\frac{1}{2}$$

We need to weight these by the number of items in each set:

Gini Purity Index



fluffy	big	friendly	animal
t	t	f	sheep
t	f	t	dog
t	f	f	dog
f	t	f	dog
t	t	t	sheep.

$$1-\left[\left(\frac{1}{3}\right)^2+\left(\frac{2}{3}\right)^2\right]=1-\frac{5}{9}=\frac{4}{9}\times\frac{3}{5}$$

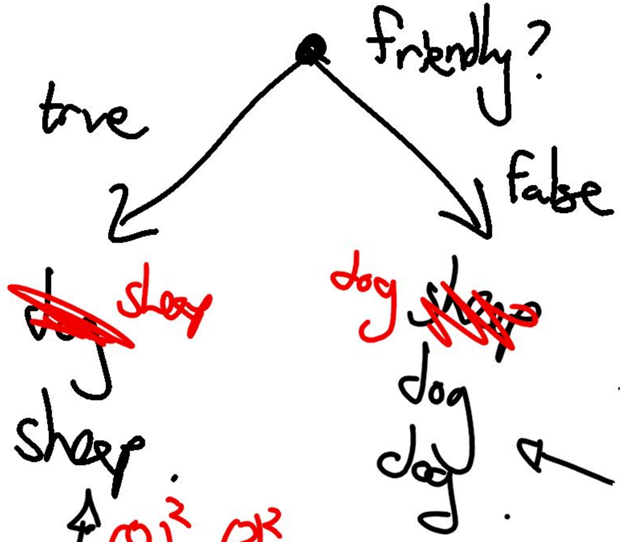
$$1-\left[\left(\frac{1}{2}\right)^2+\left(\frac{1}{2}\right)^2\right]=1-\frac{1}{2}=\frac{1}{2}\times\frac{2}{5}$$

We need to weight these by the number of items in each set:

$$=0.46$$

Gini Purity Index

If we had chosen something that perfectly split the data... the Gini Index = 0 → best!



$$\left(\frac{0}{2}\right)^2+\left(\frac{2}{2}\right)^2$$

$$1-\left[\left(\cancel{\frac{1}{2}}\right)^2+\left(\frac{1}{2}\right)^2\right] = 1-\cancel{\frac{1}{2}}\,1 = \cancel{\frac{1}{2}}\,0 \times\frac{2}{5}$$

$$\left(\frac{0}{3}\right)^2+\left(\frac{3}{3}\right)^2$$

$$1-\left[\left(\cancel{\frac{1}{3}}\right)^2+\left(\frac{2}{3}\right)^2\right] = 1-\cancel{\frac{5}{9}}\,1 = \cancel{\frac{4}{9}}\,0 \times\frac{3}{5}$$

We need to weight these by the number of items in each set:

$= \cancel{0.4 + 0.6} = 0$

Other Decision Tree Algorithms

- If your output is continuous (you’re doing regression) and you’ll need **Regression Trees**.
    - We compute the **variance** of a set of the data, instead of its information gain, to decide which feature to use.
- If you have a continuous input
    - Before splitting you find the threshold that maximises our purity metric (e.g. information gain) for each feature.

Note:
For regression tress…
We use the mean of the set at the leaf node to assign a value to the leaf.

Decision Trees can suffer from Overfitting

The likelihood of over-fitting occurring increases as a tree gets deeper because the resulting classifications are based on **smaller and smaller subsets** as the dataset is partitioned after each feature test in the path.

– Mauricio

Pruning

Pre-pruning

Stop growing the tree if the decrease in the error is not sufficient to justify the extra complexity of adding an extra subtree.

Post-pruning

Allow the algorithm to grow the tree as much as it likes and then prune the tree of the branches that cause over-fitting.

Test pruned tree on validation set (if it doesn’t do worse, then we should prune).

- Decision Trees
  - Example
  - Which tree to use? (want to use features early that split data into sets that are as pure as possible)
- Entropy & Information Gain
- Building trees:
  - ID3 | CART
  - Other stuff: continuous features, etc.
- Bootstrap [Resample -> get distribution]
- Boosting [e.g. AdaBoost: committee of weak learners -> weighted vote]
  - Bootstrap + Bagging + Decision Trees -> Random Forest
- Bagging ["bootstrap aggregation"] - Bootstrap -> model -> take average
  - When applied to random forests also only a subset of features are available to each tree - increases independence.

Decision Trees

Detour: Entropy & Information Gain

Building trees

Details about decision trees

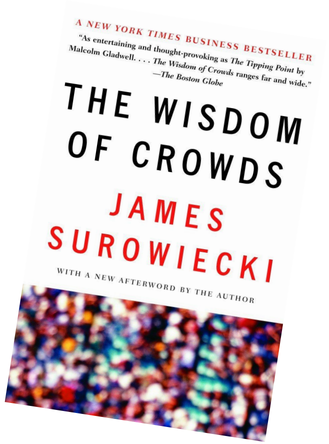
Ensembles

Boosting

- ‘A procedure that combines the outputs of many “weak” classifiers to produce a powerful “committee”’
  – Hastie, p337.

Outline:

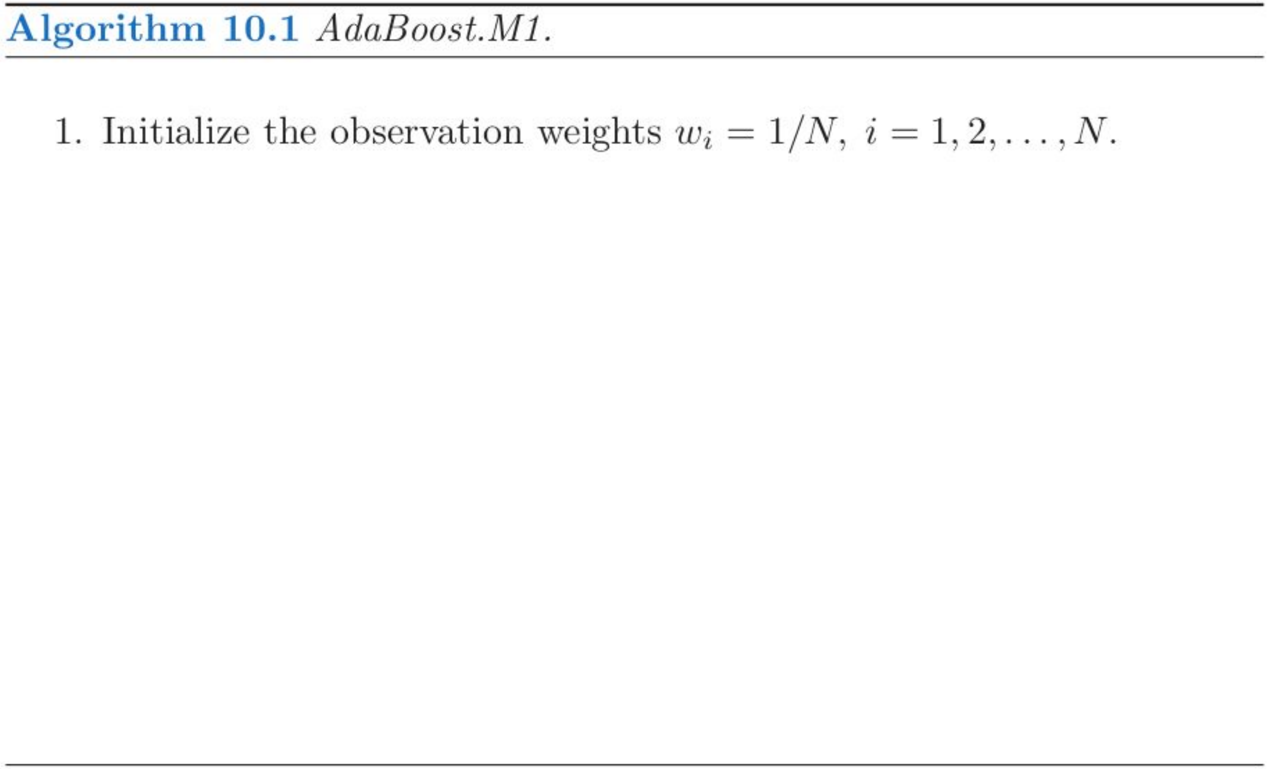
- Iteratively train multiple models on the same data.
- But put more weight on points misclassified by the previous model.
- Combine the models by weighting them sort of by their accuracy.


A NEW YORK TIMES BUSINESS BESTSELLER
“As entertaining and thought-provoking as The Tipping Point by Malcolm Gladwell. . . . The Wisdom of Crowds ranges far and wide.” —The Boston Globe
THE WISDOM OF CROWDS
JAMES SUROWIECKI
WITH A NEW AFTERWORD BY THE AUTHOR

Resembles bagging [bootstrap aggregating], however ‘the connection is at best superficial and that boosting is fundamentally different’.

AdaBoost

For the two class problem, encoded by $Y \in \{-1, 1\}$



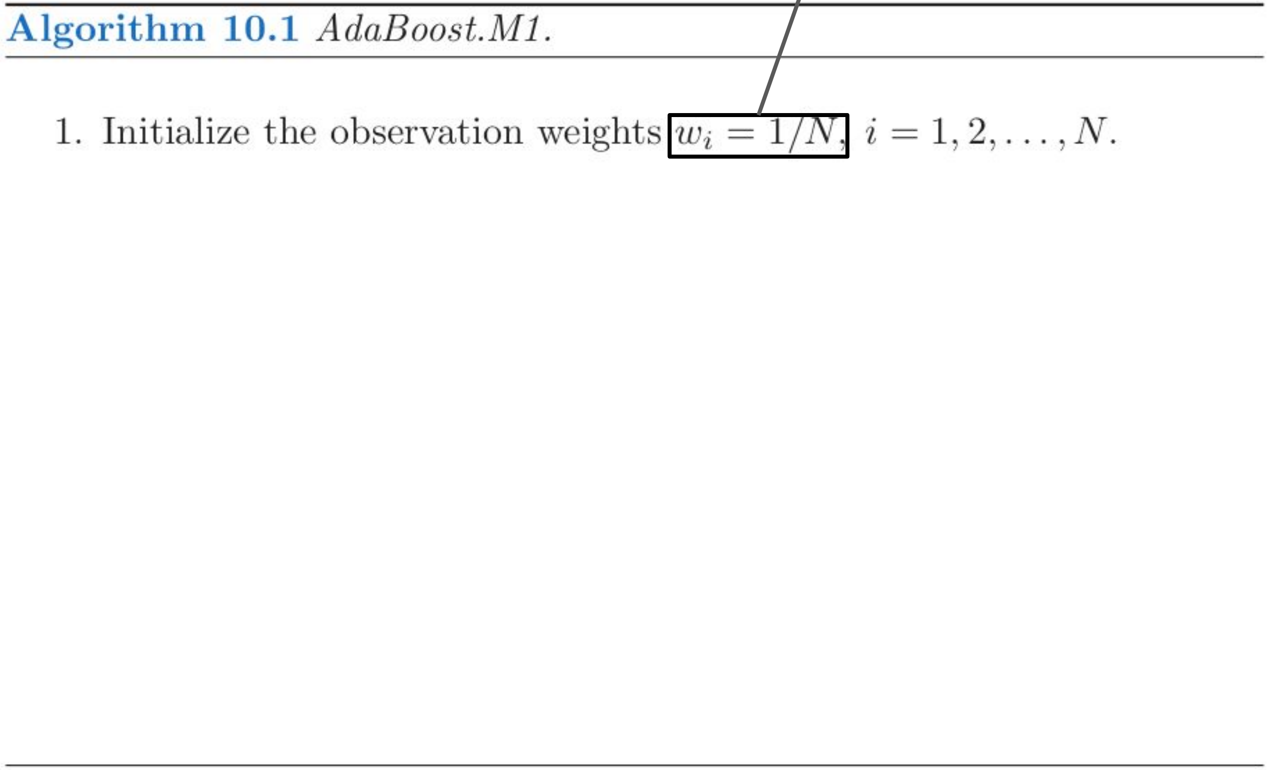
Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.

AdaBoost

For the two class problem, encoded by $Y \in \{-1, 1\}$

Initialise the weights of each training point to be equal.



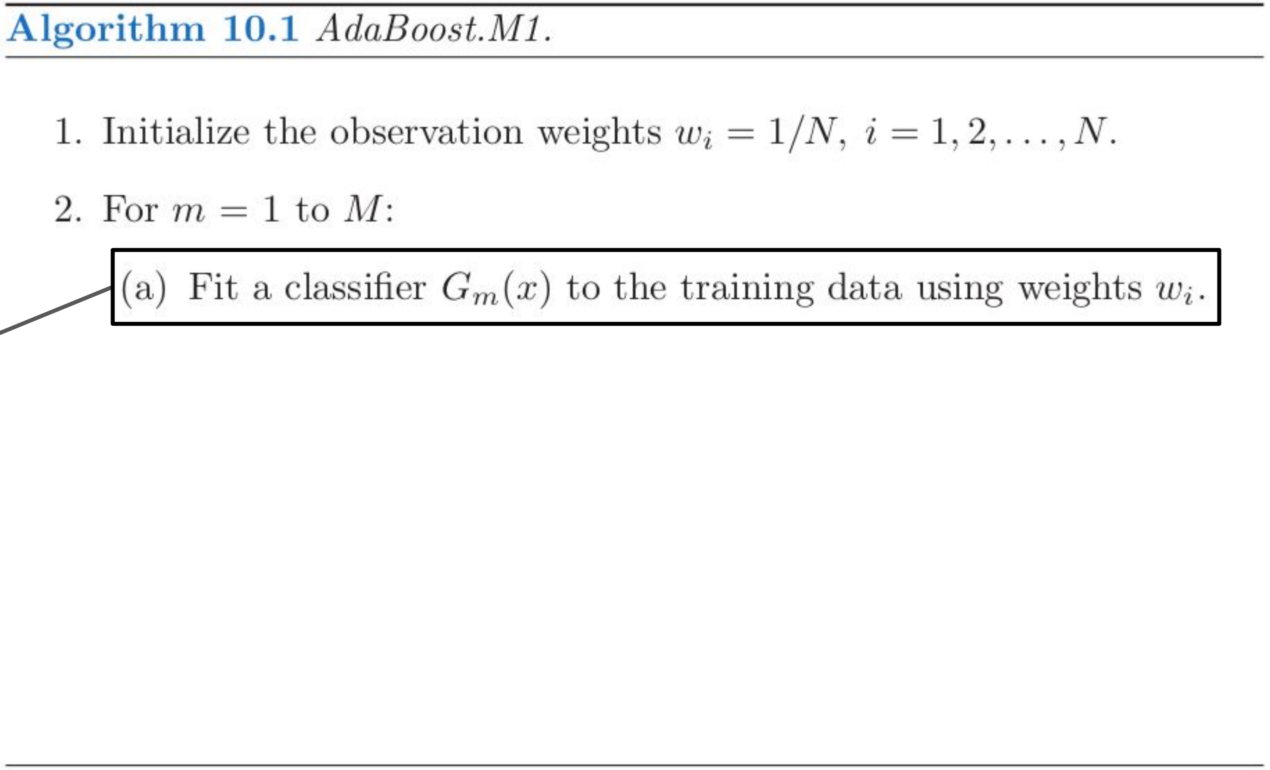
Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $\boxed{w_i = 1/N,}\ i = 1, 2, \ldots, N$.

AdaBoost

For the two class problem, encoded by $Y \in \{-1, 1\}$

Fit a ‘weak’ classifier to the training data, with the influence of each training point weighted by w_i.

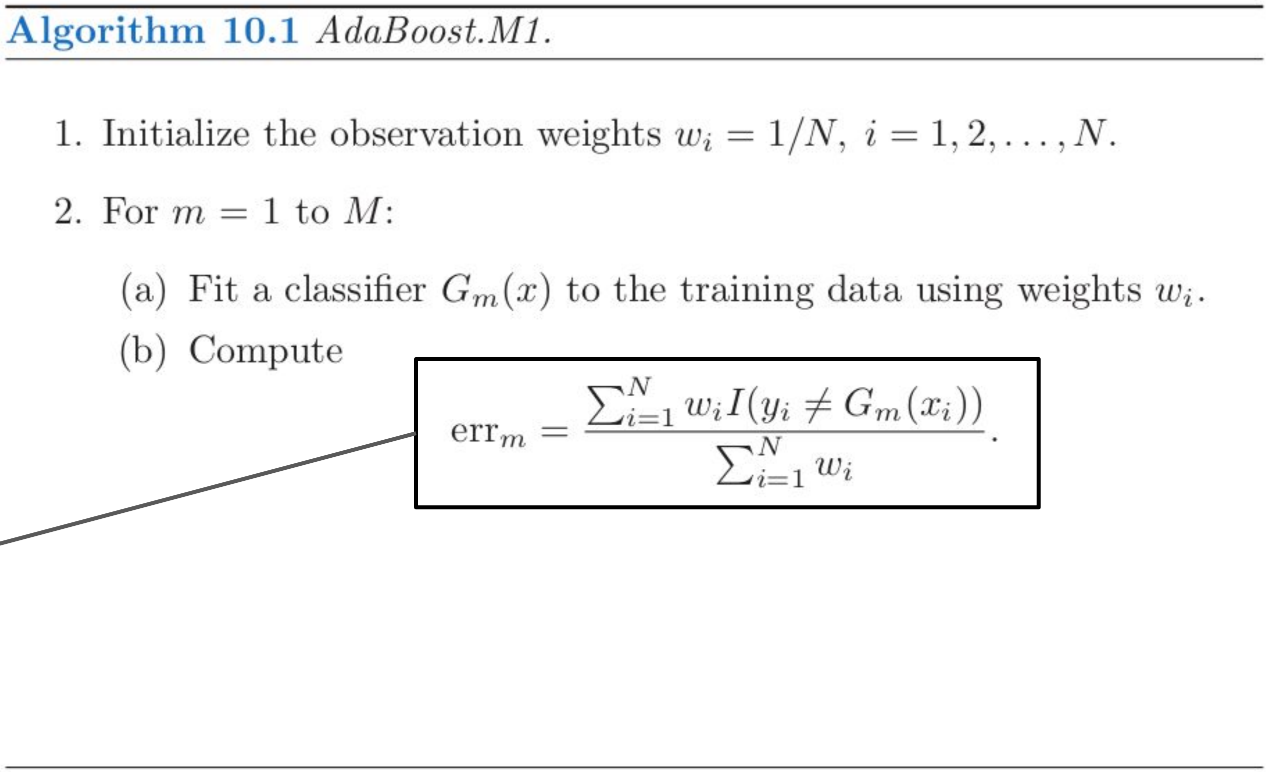


Algorithm 10.1 *AdaBoost.M1.*

```
1. Initialize the observation weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$:
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
```

AdaBoost

For the two class problem, encoded by $Y \in \{-1, 1\}$



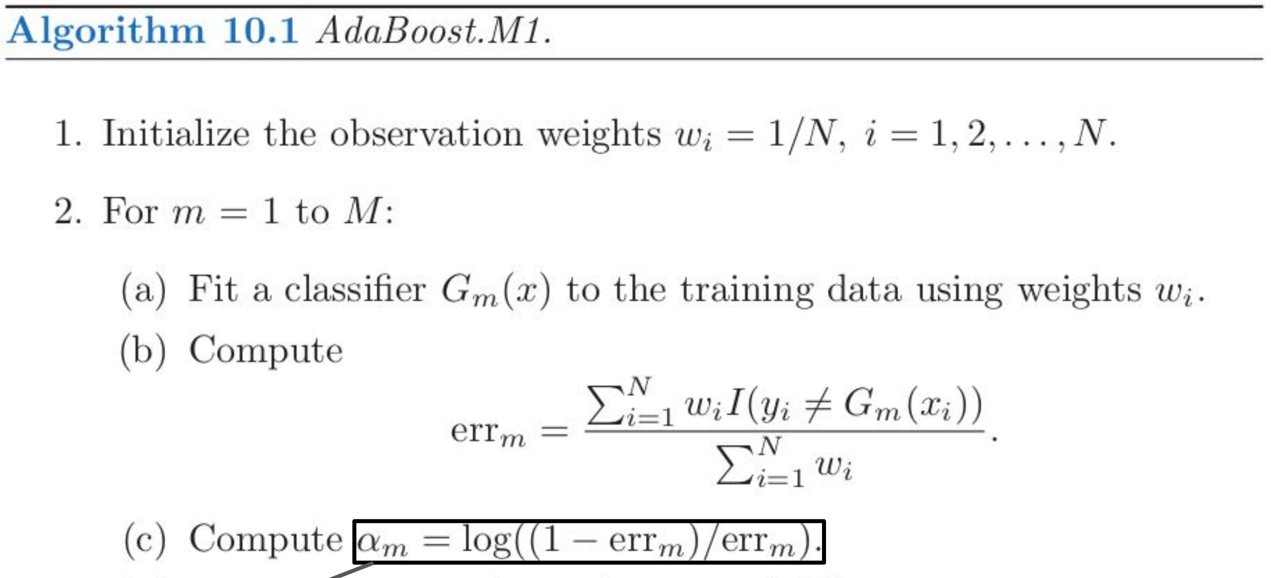
Algorithm 10.1 *AdaBoost.M1.*

```
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$:
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
    (b) Compute
        $\boxed{\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.}$
```

Compute ratio of weighted misclassifications.

AdaBoost

For the two class problem, encoded by $Y \in \{-1, 1\}$



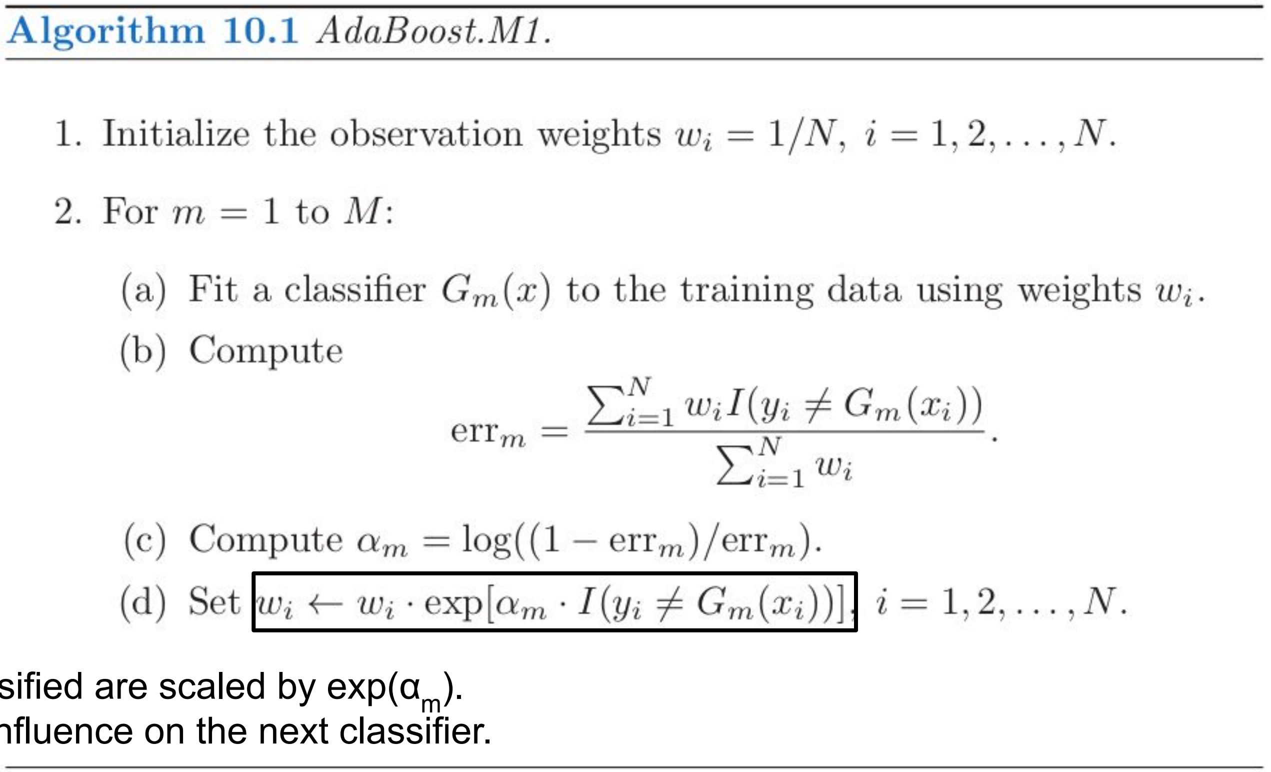
Algorithm 10.1 *AdaBoost.M1.*

```
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$:
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
    (b) Compute
            $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
    (c) Compute $\boxed{\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m).}$
```

We'll weight each model by α_m, its log of (correct/incorrect ratios).
So if a model gets [weighted] 90% correct, the α_m= log 9.
Side note: This choice is due to the choice of an exponential loss function. See Section 10.4, p343 of Hastie (2017) for a derivation.

AdaBoost

For the two class problem, encoded by $Y \in \{-1, 1\}$



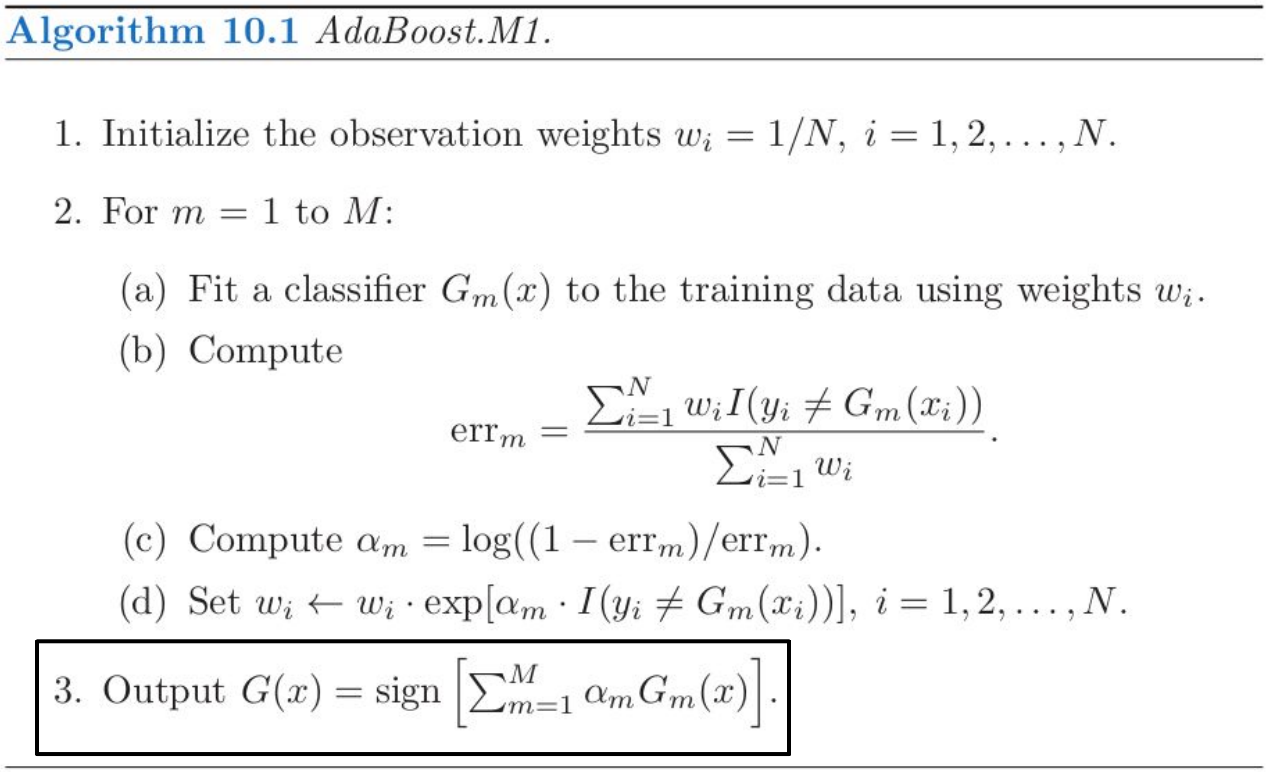
Algorithm 10.1 *AdaBoost.M1.*

```
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$:
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
    (b) Compute
        $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
    (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
    (d) Set $\boxed{w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]},\ i = 1, 2, \ldots, N$.
```

Observations incorrectly classified are scaled by $\exp(\alpha_m)$.
This increases their relative influence on the next classifier.

AdaBoost

For the two class problem, encoded by $Y \in \{-1, 1\}$



Algorithm 10.1 *AdaBoost.M1.*

```
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$:
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
    (b) Compute
            $\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$.
    (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
    (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.
```

Finally, output weighted sum of models to predict a given input x.

The summation works as 0 is the threshold (due to encoding of Y).

AdaBoost Example…

Bootstrapping

Really different from Boosting!
Unrelated!

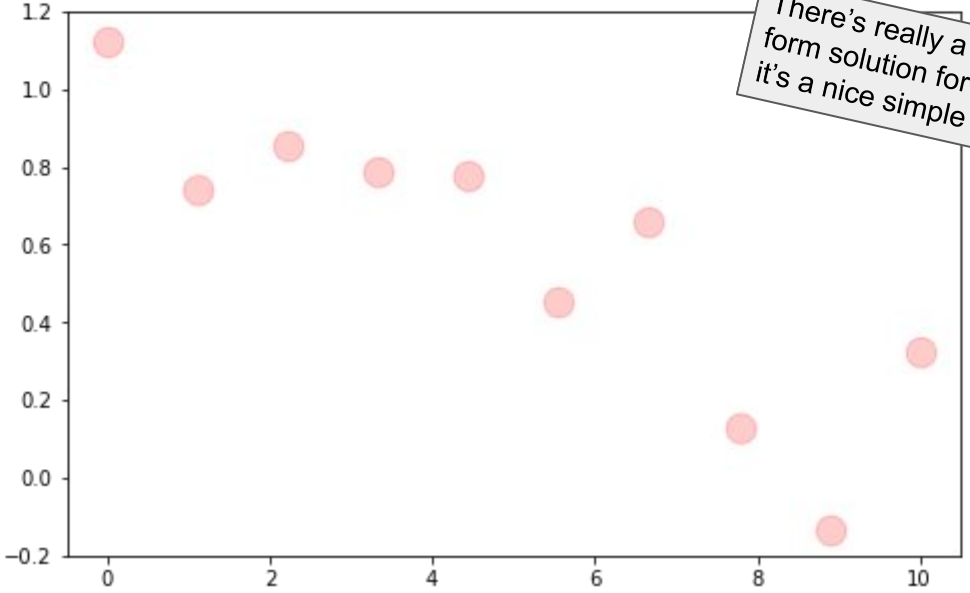
Definition: Any method that uses random sampling with replacement, to estimate distributions or statistics.

Bootstrapping

Definition: Any method that uses random sampling with replacement, to estimate distributions or statistics.

Example: Linear regression

What is the uncertainty in our predictions?



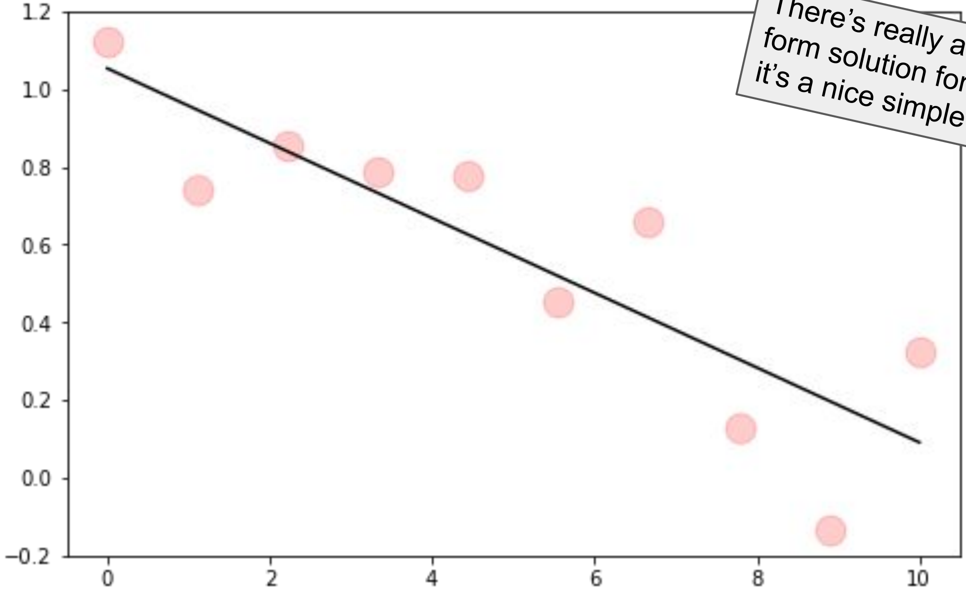
There's really a closed form solution for this, but it's a nice simple example!

Bootstrapping

Definition: Any method that uses random sampling with replacement, to estimate distributions or statistics.

Example: Linear regression

What is the uncertainty in our predictions?

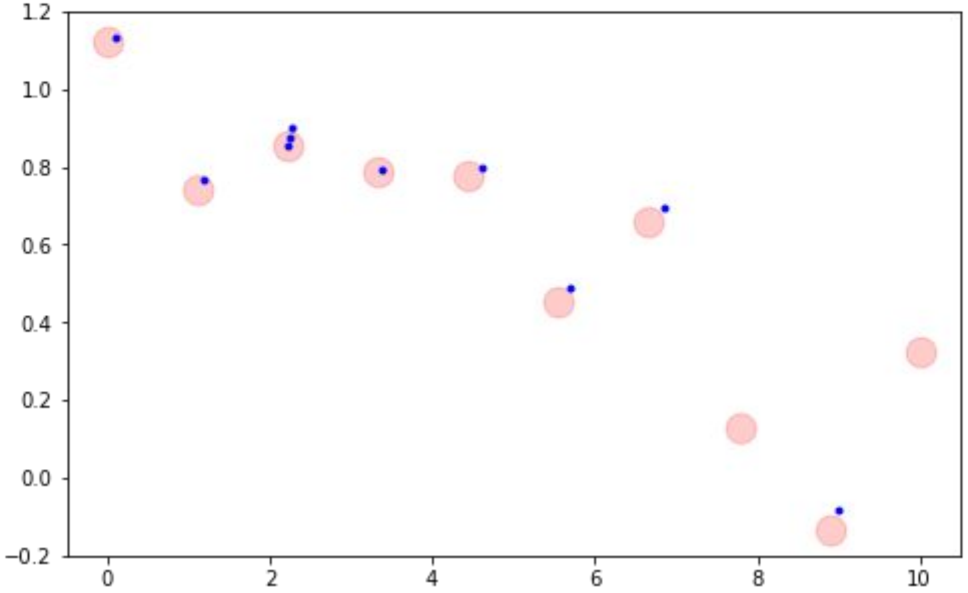


There's really a closed form solution for this, but it's a nice simple example!

Bootstrapping

We sample *with replacement* the ten points in our sample.

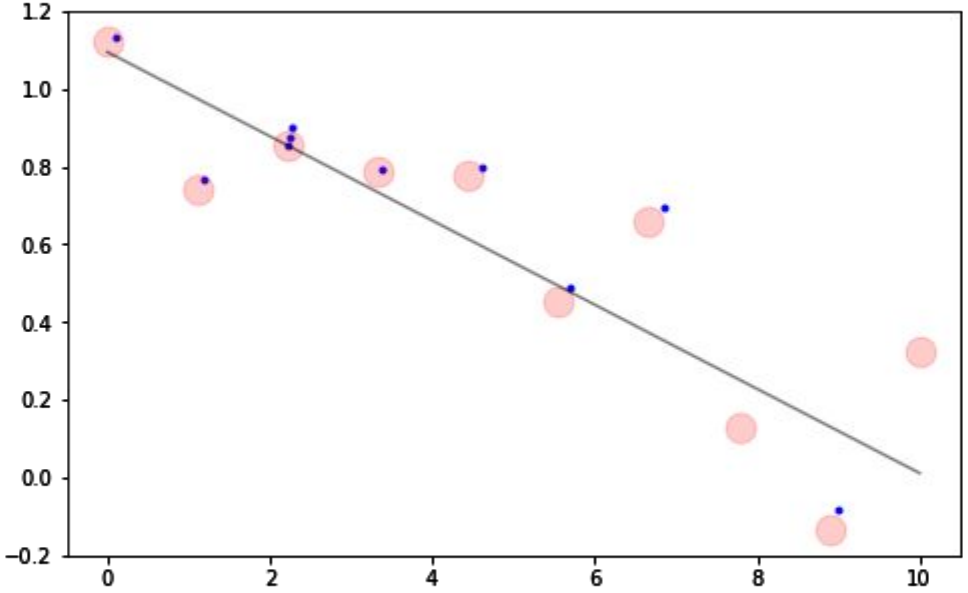
We then fit our linear model to this sample



Bootstrapping

We sample *with replacement* the ten points in our sample.

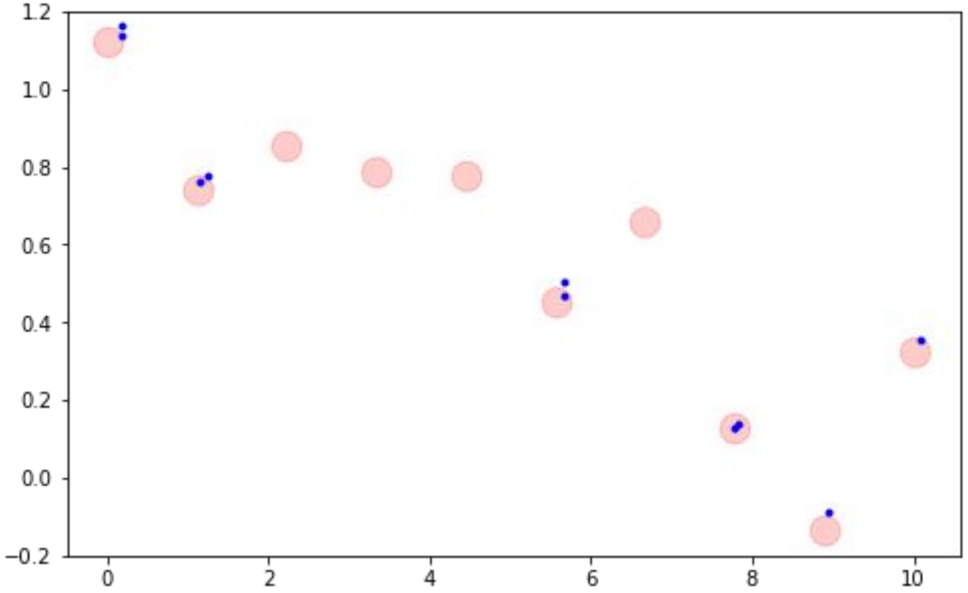
We then fit our linear model to this sample



Bootstrapping

Sample again *with replacement*.

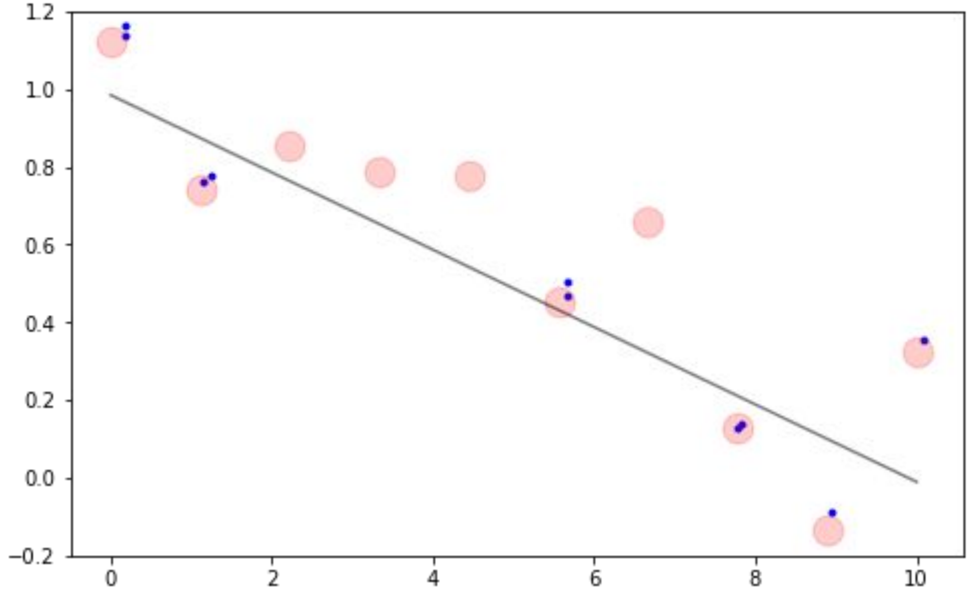
Fit our linear model to this new sample



Bootstrapping

Sample again *with replacement*.

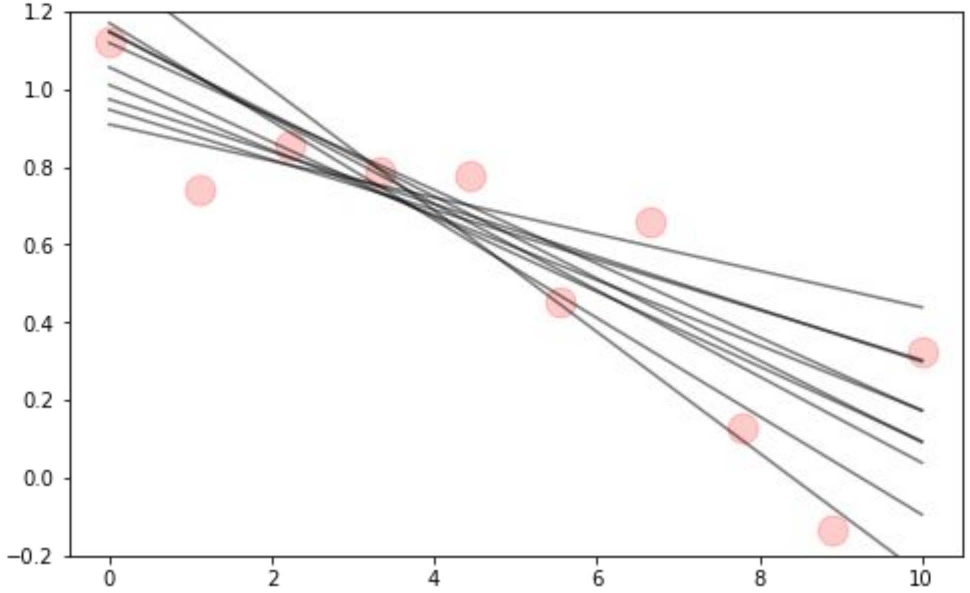
Fit our linear model to this new sample



Bootstrapping

Resample lots of times.

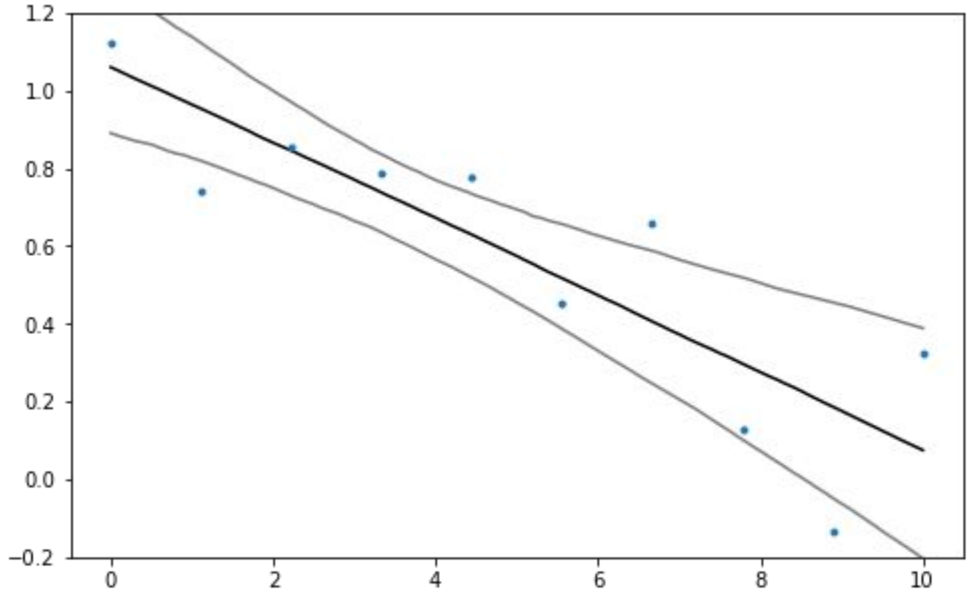
This gives us an approximation to the distribution over the population's statistics of interest.



Bootstrapping

Resample lots of times.

This gives us an approximation to the distribution over the population's statistics of interest.



95% CI on latent function

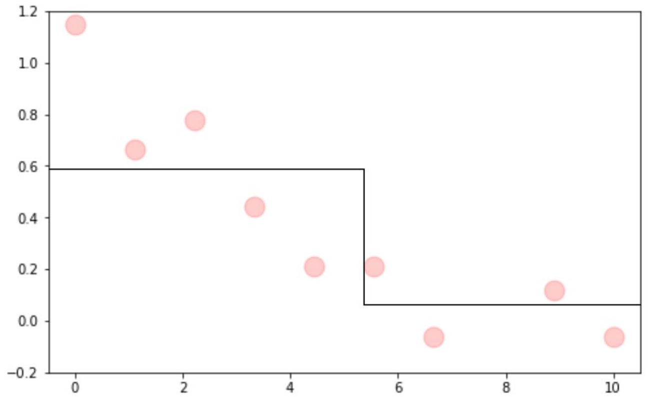
Bootstrap **Agg**regat**ing** (Bagging)

Section 8.4 in Hastie shows that the bootstrap mean approximates the posterior average.

Average the statistic of interest from the result of the model fit to each bootstrap sample.

This approach helps when applied to the output of models or algorithms that are “*unstable*”, e.g. decision trees, neural networks: These have high variance, but low bias.

E.g. a single ‘stump’ decision tree on our data might look like this. Resampling will lead to potentially a very different tree.



Bootstrap **Agg**regat**in**g (Bagging)

Section 8.4 in Hastie shows that the bootstrap mean approximates the posterior average.

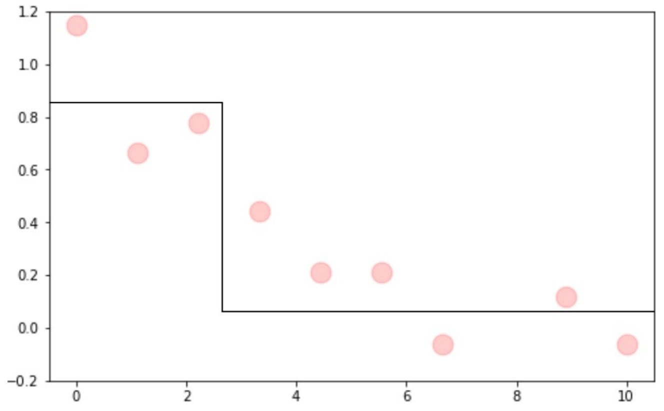
Side note: Because bootstrapping leaves out data from each sample, we can use these as held out validation data.

Average the statistic of interest from the result of the model fit to each bootstrap sample.

This approach helps when applied to the output of models or algorithms that are "*unstable*", e.g. decision trees, neural networks: These have high variance, but low bias.

E.g. a single 'stump' decision tree on our data might look like this. Small changes in the data can lead to potentially a very different tree.

Averaging lots of these together helps!



Bagging example

Bagging for Random Forest

- Bagging works by making samples ~independent.
- To further encourage this we do **subspace sampling**: Only a subset of features are provided at each node. Why is this so important?
    - Remember the variance of the distribution of the sample mean for a population with variance σ^2, is σ^2/N, assuming N independent samples.
    - If they have correlation ρ then the variance is only reduced to: $\rho\sigma^2 + \frac{1-\rho}{N}\sigma^2$

- For regression we find the average output by all the decision trees.
- For classification we take the majority.
    - Note that both cases, all the trees are weighted equally.

Bagging + subspace sampling for decision trees = random forest

The idea in random forests is to improve the variance reduction of bagging by **reducing the correlation between the trees**, without increasing the variance too much. This is achieved in the tree-growing process through random selection of the input variables.

– Hastie (p588)

Random Forest Example

First we take a bootstrap sample

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

Original

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
true	true	true	spam
false	false	false	ham
false	false	true	ham
false	false	true	ham
false	false	true	ham

Sample

Random Forest Example

At each decision node we do subspace sampling (often only keep √D of the number of features).

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

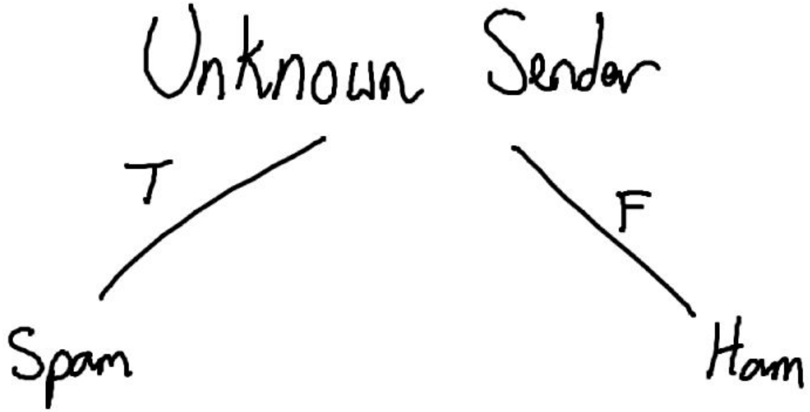
Original

	Unknown Sender		Class
	false		spam
	true		spam
	true		spam
	false		ham
	false		ham
	false		ham
	false		ham

Sample

Random Forest Example

We build the decision tree…

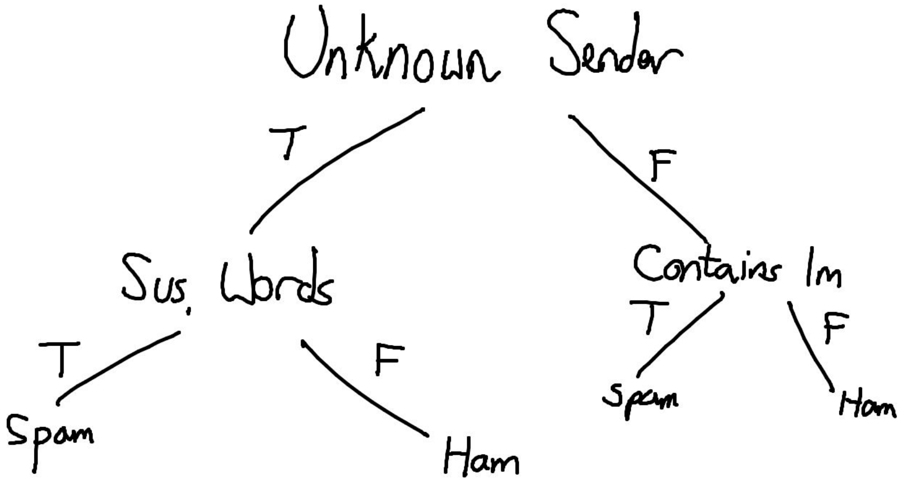


	Unknown Sender		Class
	false		spam
	true		spam
	true		spam
	false		ham
	false		ham
	false		ham
	false		ham

Sample

Random Forest Example

We build the decision tree… at each step using a subsample of features



		Contains Images	Class
		true	spam
		true	spam
		true	spam
		false	ham
		true	ham
		true	ham
		true	ham

Sample

Random Forest Example

We repeat. Take another bootstrap sample…

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

Original

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	false	true	spam
true	false	true	spam
false	false	false	ham
false	false	false	ham
false	false	false	ham
true	true	false	spam

Sample

Random Forest Example

We repeat. Take another bootstrap sample… subsample at each node…

Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam
true	true	true	spam
false	false	false	ham
false	false	false	ham
false	false	true	ham
false	false	false	ham
true	true	false	spam

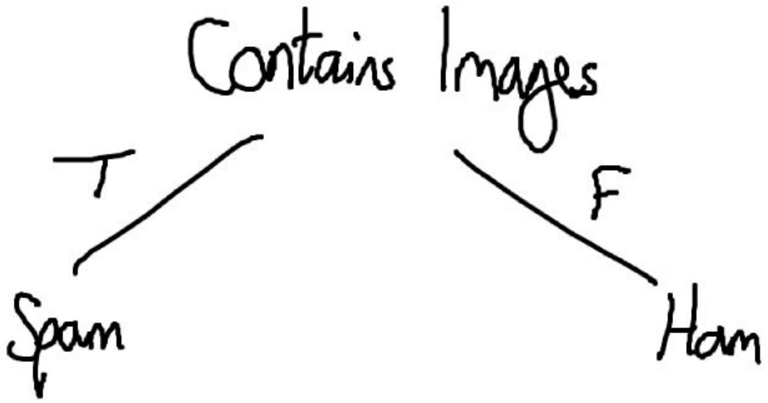
Original

Contains Images	Class
true	spam
true	spam
true	spam
false	ham
false	ham
false	ham
false	spam

Sample

Random Forest Example

We repeat. Take another bootstrap sample… subsample at each node… building up the decision tree…

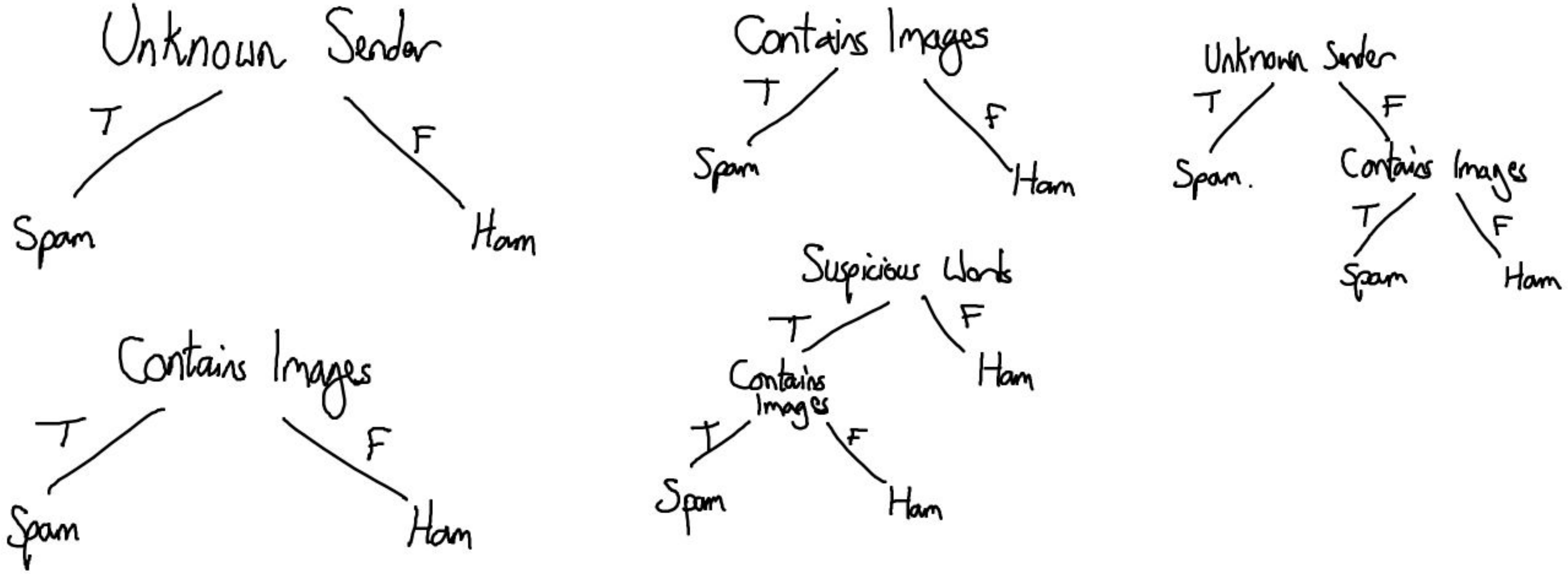


	Contains Images	Class
	true	spam
	true	spam
	true	spam
	false	ham
	false	ham
	false	ham
	false	spam

Sample

Random Forest Example

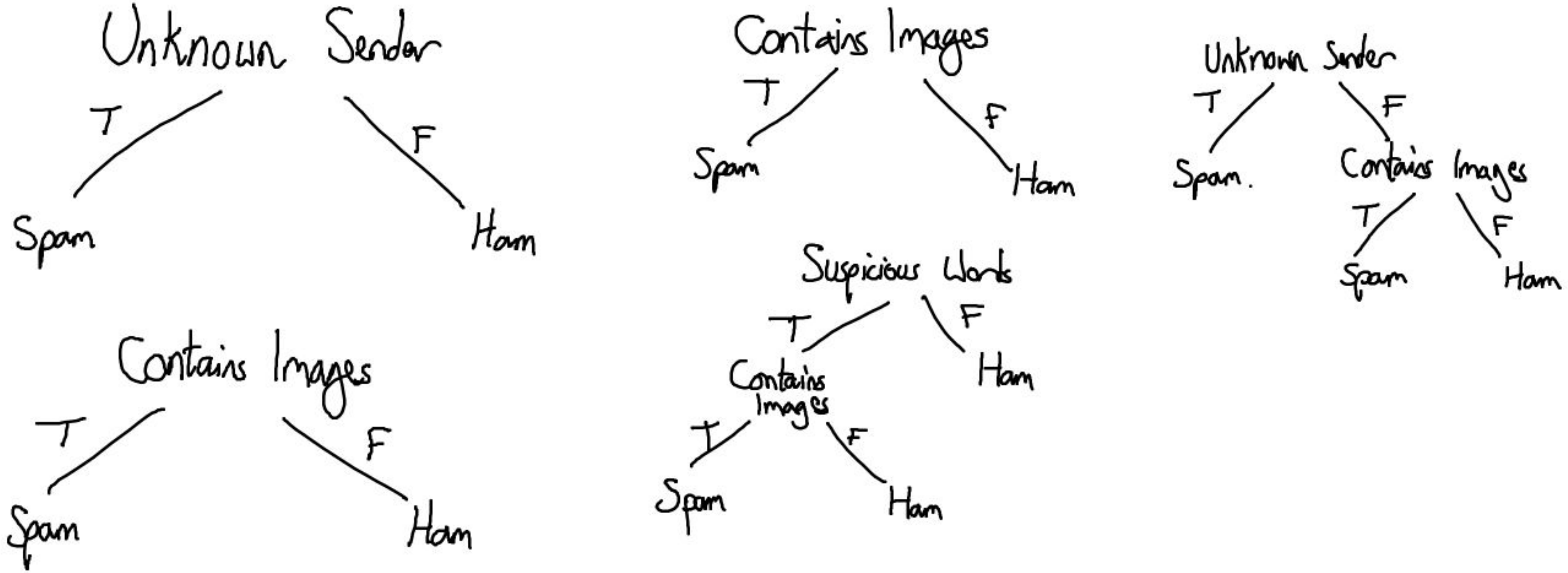
We keep repeating until we have lots of decision trees.



Random Forest Example

We keep repeating until we have lots of decision trees.

If classifying we take the majority vote

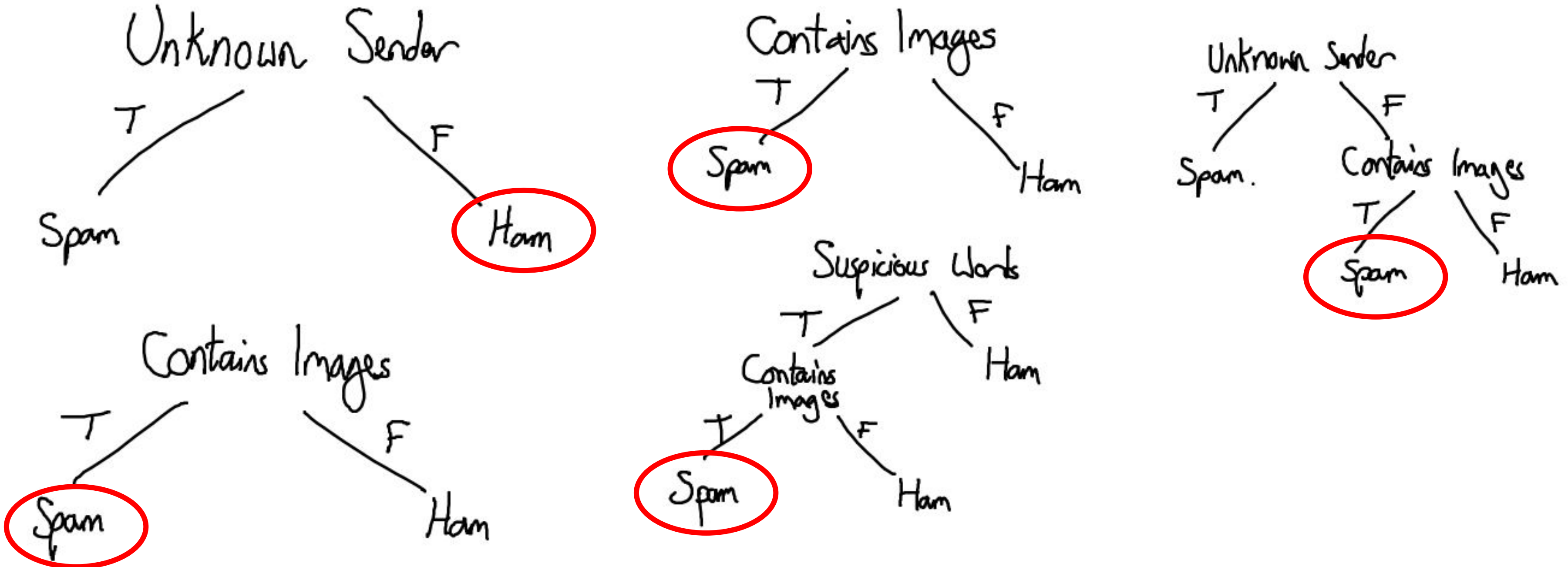


Suspicious Words	Unknown Sender	Contains Images	Class
true	false	true	spam

Random Forest Example

We keep repeating until we have lots of decision trees.

If classifying we take the majority vote: Spam.



Summary

It’s not necessarily going to give the best results, but…

- It is really versatile and doesn’t need much tuning / preprocessing.

Summary

It’s not necessarily going to give the best results, but…

- It is really versatile and doesn’t need much tuning / preprocessing.
- Works with high dimensions (thanks to the subspace sampling?)

Summary

It’s not necessarily going to give the best results, but…

- It is really versatile and doesn’t need much tuning / preprocessing.
- Works with high dimensions (thanks to the subspace sampling?)
- It can be parallelised

Summary

It’s not necessarily going to give the best results, but…

- It is really versatile and doesn’t need much tuning / preprocessing.
- Works with high dimensions (thanks to the subspace sampling?)
- It can be parallelised
- It deals well with awkward data (outliers, non-linearities, unbalanced, etc).

Summary

It’s not necessarily going to give the best results, but…

- It is really versatile and doesn’t need much tuning / preprocessing.
- Works with high dimensions (thanks to the subspace sampling?)
- It can be parallelised
- It deals well with awkward data (outliers, non-linearities, unbalanced, etc).

The problems:

- We are unable to bring our own prior knowledge about the model.

Summary

It’s not necessarily going to give the best results, but…

- It is really versatile and doesn’t need much tuning / preprocessing.
- Works with high dimensions (thanks to the subspace sampling?)
- It can be parallelised
- It deals well with awkward data (outliers, non-linearities, unbalanced, etc).

The problems:

- We are unable to bring our own prior knowledge about the model.
- It suffers from common problems of overfitting & failing to extrapolate.
- Very uninterpretable.

Random Forest Summary

It’s not necessarily going to give the best results, but…

- It is really versatile and doesn’t need much tuning / preprocessing.
- Works with high dimensions (thanks to the subspace sampling?)
- It can be parallelised
- It deals well with awkward data (outliers, non-linearities, unbalanced, etc).

The problems:

- We are unable to bring our own prior knowledge about the model.
- It suffers from common problems of overfitting & failing to extrapolate.
- Very uninterpretable.

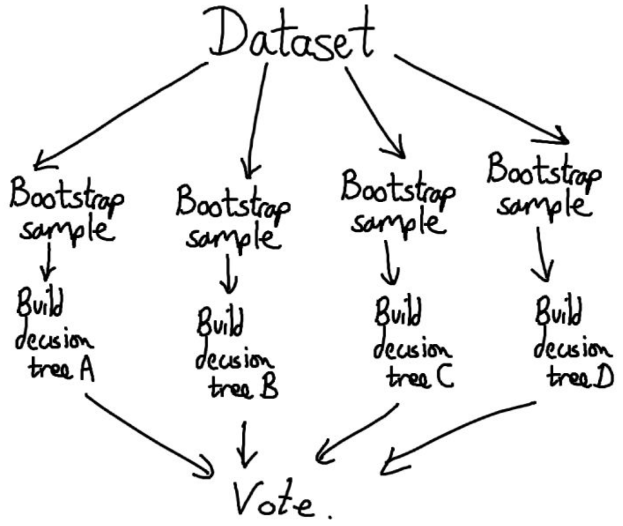
Just like other ‘general purpose’ ML: It can be good but we might want to build a model that is specific to our problem.

Summary / Take Home

- **Decision trees** are built by selecting a feature that maximises a measure of **‘purity’** in the sets created.
  - **Information Gain** is one measure of purity: Expected change in entropy
  - **Entropy** is negative expected log probability

Ensemble Methods

- **Bagging**: Find the mean of several bootstrap samples.
- **Random forest**: A bagged decision tree classifier, in which a subset of features is used at each split. The trees as weighted equally to make the prediction.
- **Boosting**: An alternative approach in which we weight the training points depending on how well they were classified previously. We weight the trees when making a prediction.



Random forest

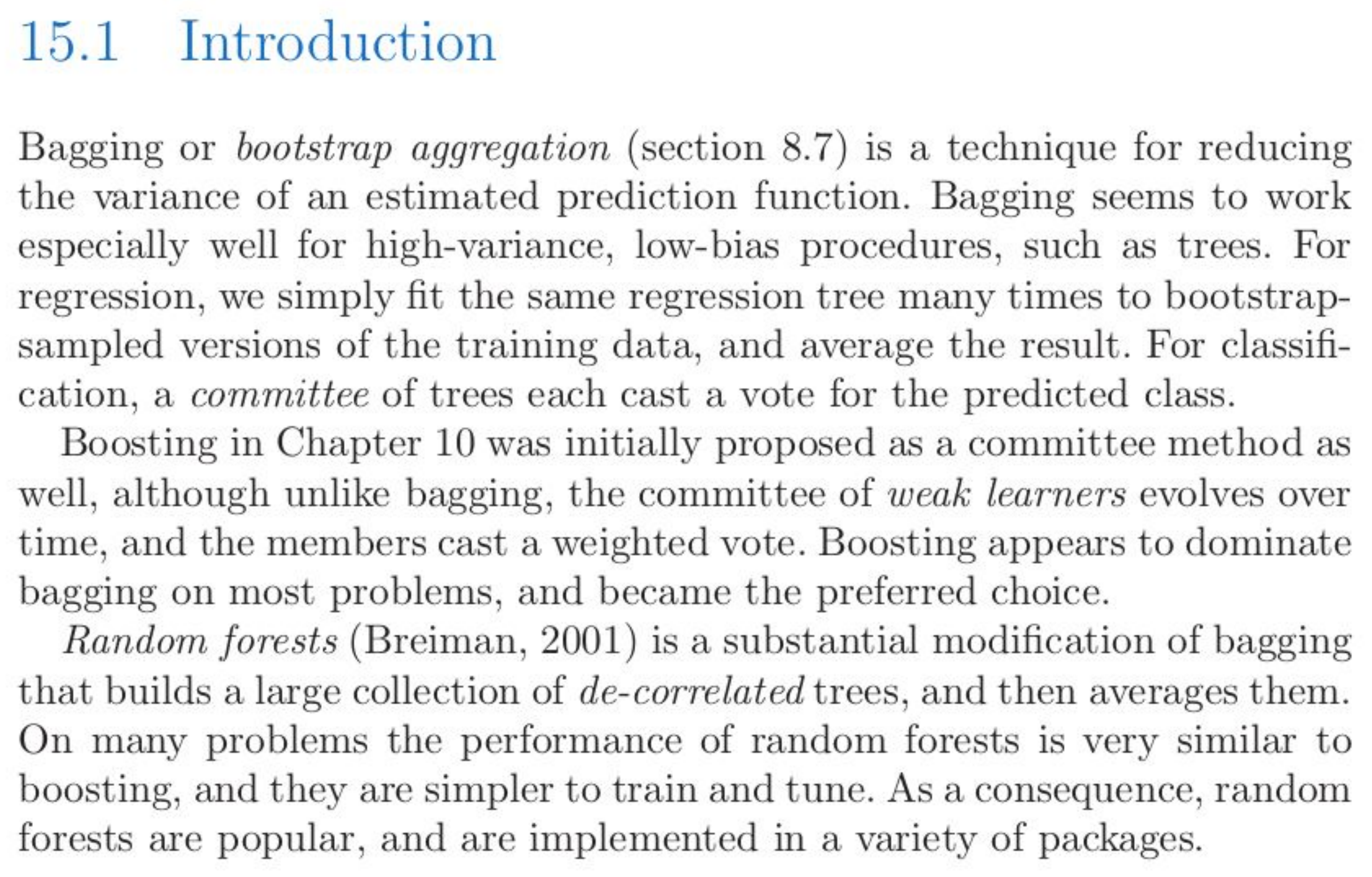
B

Bc

er

pe

Ba



15.1 Introduction

Bagging or *bootstrap aggregation* (section 8.7) is a technique for reducing the variance of an estimated prediction function. Bagging seems to work especially well for high-variance, low-bias procedures, such as trees. For regression, we simply fit the same regression tree many times to bootstrap-sampled versions of the training data, and average the result. For classification, a *committee* of trees each cast a vote for the predicted class.

Boosting in Chapter 10 was initially proposed as a committee method as well, although unlike bagging, the committee of *weak learners* evolves over time, and the members cast a weighted vote. Boosting appears to dominate bagging on most problems, and became the preferred choice.

Random forests (Breiman, 2001) is a substantial modification of bagging that builds a large collection of *de-correlated* trees, and then averages them. On many problems the performance of random forests is very similar to boosting, and they are simpler to train and tune. As a consequence, random forests are popular, and are implemented in a variety of packages.