Machine Learning 1.03: Random Forests

Tom S. F. Haines
T.S.F.Haines@bath.ac.uk

This lecture

- Last lecture:
  - Decision trees

    Input $=$ real

    Output $=$ categorical (integer)

This lecture

- Last lecture:
  - Decision trees

    Input = real

    Output = categorical (integer)

- This lecture:
  - Decision trees

    Input = real, categorical, directional.

    Output = real, categorical, directional.

  - Multiple outputs
  - Bagging
  - Random forests

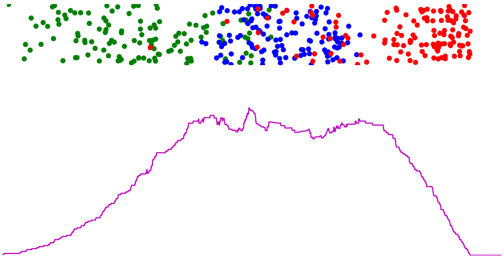
- Goal: One good algorithm for most supervised problems!

Real input

(from last time)

- Find the greatest info gain (or Gini impurity) across all axes

  (1 axis shown, vertical offset for visualisation only)

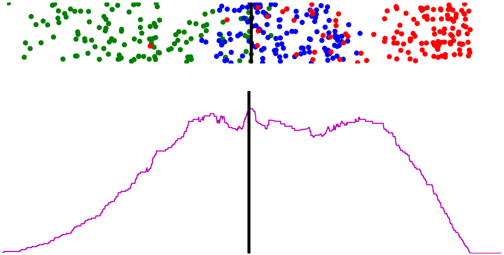


Real input

(from last time)

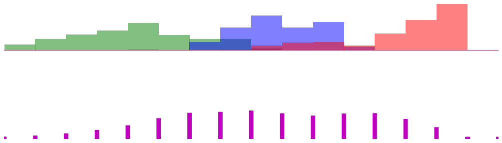
- Find the greatest info gain (or Gini impurity) across all axes

  (1 axis shown, vertical offset for visualisation only)



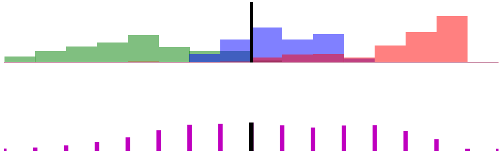
Quantised real input

- Continuous data may be collected **quantised**,
- e.g. When asking "What is your age?"
  ask which of 12–17, 18–24 etc. it falls into rather than a specific value.



Quantised real input

- Continuous data may be collected **quantised**,
- e.g. When asking "What is your age?"
  ask which of 12–17, 18–24 etc. it falls into rather than a specific value.



- Still split – just have to do it between bins.
  (Information gain shown as spikes
  as only defined at bin transitions)

Categorical input

- Similar to quantised (histogram of classes)...
  ...but unordered.
- e.g. "What is your favourite cheese?"
- Spliting no longer makes sense!

Categorical input

- Similar to quantised (histogram of classes)...
  ...but unordered.
- e.g. "What is your favourite cheese?"
- Spliting no longer makes sense!

- Two choices:
  - Try every assignment of class to the left/right side and choose best.
    (with one classes always going left to account for symmetry)
  - One class goes down left branch, rest go right.

Categorical input

- Similar to quantised (histogram of classes)...

  ...but unordered.
- e.g. "What is your favourite cheese?"
- Spliting no longer makes sense!

- Two choices:
  - Try every assignment of class to the left/right side and choose best.

    (with one classes always going left to account for symmetry)
  - One class goes down left branch, rest go right.

- No difference for 2 or 3 classes!
- One left, rest right is preferred if more:
  - Fixed storage requirement/simpler code
  - Number of combinations grows linearly

    (n, not 2^{n-1} where n is the numbers of classes)

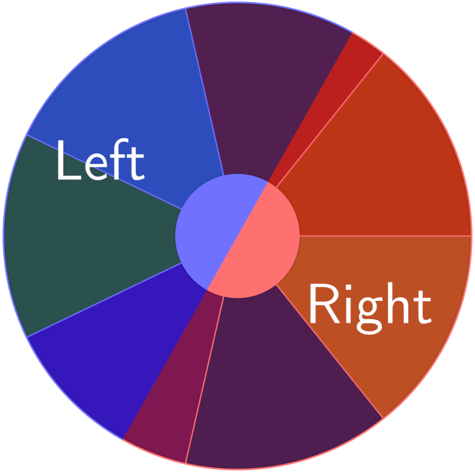
UNIVERSITY OF BATH

Directional input

- Direction = Real number(s) that wrap around (modular arithmetic)
- Examples:
  - Angles
  - Time of day, day of week, month of year...
  - Octaves (music)

Directional input

- Direction $=$ Real number(s) that wrap around (modular arithmetic)
- Examples:
  - Angles
  - Time of day, day of week, month of year...
  - Octaves (music)
- Single split makes no sense...
  ...chop in half instead!
- Use unit length vectors instead of angles,
  $\theta \rightarrow [\cos(\theta), \sin(\theta)]^T$
  Dot product feature with "left angle" and send left if positive.



Regression

- Classification:
    - Split to minimise Gini impurity or maximise information gain.
    - Leaf gives answer as most common class to reach it.

Regression

- Classification:
  - Split to minimise Gini impurity or maximise information gain.
  - Leaf gives answer as most common class to reach it.
- Regression:
  - Split to minimise variance or maximise information gain.
  - Leaf gives answer as mean value to reach it.

    (median/medoid only rarely confers an advantage)
- Otherwise identical!

Variance reduction

- Variance of output measures how consistent a node is...
  ...so choose splits that minimise it.

Variance reduction

- Variance of output measures how consistent a node is...
  ...so choose splits that minimise it.

- Variance of left node:

$$\sigma_l^2 = \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])^2\right]$$

similarly for right, σ_r^2 (expectations estimated using data that reaches node).

Variance reduction

- Variance of output measures how consistent a node is...

  ...so choose splits that minimise it.

- Variance of left node:

$$\sigma_l^2 = \mathbb{E}\left[\left(f(x) - \mathbb{E}\left[f(x)\right]\right)^2\right]$$

  similarly for right, σ_r^2 (expectations estimated using data that reaches node).

- Minimise weighted combination:

$$L(\texttt{split}) = \frac{n_l}{n}\sigma_l^2 + \frac{n_r}{n}\sigma_r^2$$

  $n =$ total exemplar count,
  $n_l =$ exemplars traveling down left branch,
  $n_r =$ exemplars traveling down right branch.

Information gain

- Not used very often – variance reduction both good and faster.
- Including to show generalisation.

Information gain

- Not used very often – variance reduction both good and faster.
- Including to show generalisation.

- Fit Gaussian to output variable, entropy is

$$\frac{1}{2}\log\left(2\pi e\sigma^2\right)$$

- Hence information gain of a split:

$$I(\texttt{split}) = \frac{1}{2}\log\left(2\pi e\sigma_p^2\right) - \frac{n_l}{2n}\log\left(2\pi e\sigma_l^2\right) - \frac{n_r}{2n}\log\left(2\pi e\sigma_r^2\right)$$

(reusing variables from previous slide, $+p$ subscript for parent)

Directional output

- Regression with different rules.
- Good example of unusual data type[1].

- Represent angles with unit length vectors, $\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n\}$, $\hat{x}_i = [\cos\theta_i, \sin\theta_i]^T$
- "Mean" angle $= \frac{\sum \hat{x}}{|\sum \hat{x}|}$
  (use for leaf nodes)

- Residual, $r = \frac{|\sum \hat{x}|}{n}$
- Variance $= 1 - r$
  (use to select splits with variance reduction)

[1] This is assuming the data is distributed with the von-Mises distribution

Multiple outputs I

- Predicting multiple outputs from a single tree.

- Information gain $\rightarrow$ bits are added (assuming independent)
- e.g. 3D positions can use the entropy of a multivariate Gaussian.

- Everything else requires some kind of hack!

Multiple outputs I

- Predicting multiple outputs from a single tree.

- Information gain $\rightarrow$ bits are added (assuming independent)
- e.g. 3D positions can use the entropy of a multivariate Gaussian.

- Everything else requires some kind of hack!

- Could train n models instead. How to decide?
  (one per output)
  - Outputs uncorrelated: Separate models

    $\vdots$

  - Outputs correlated: One model
- This means you never actually add bits!

Random Forests

Ensemble learning I

- Random forest $=$ decision trees $+$ ensemble learning

Ensemble learning I

- Random forest = decision trees + ensemble learning
- Ensemble learning = combining multiple estimators
  - Different models (e.g. linear regression, decision tree, SVM, neural network).
    or
  - Same model, randomised training so each is different

(using estimator to distinguish from model)

Ensemble learning I

- Random forest = decision trees + ensemble learning
- Ensemble learning = combining multiple estimators
  - Different models (e.g. linear regression, decision tree, SVM, neural network).

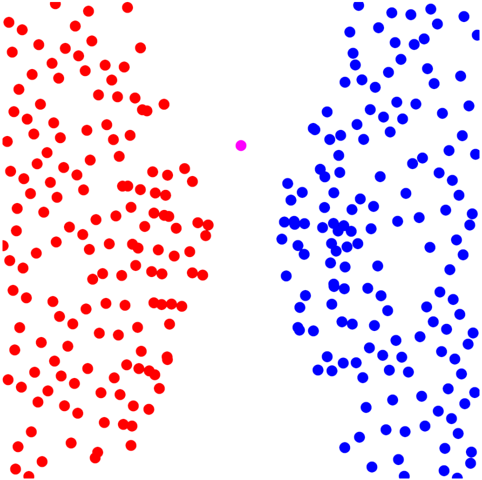
    or
  - Same model, randomised training so each is different

  (using estimator to distinguish from model)
- Random forest:
  - Many decision trees (hence name).
  - Randomised training using **bagging**,

    a specific ensemble learning technique.

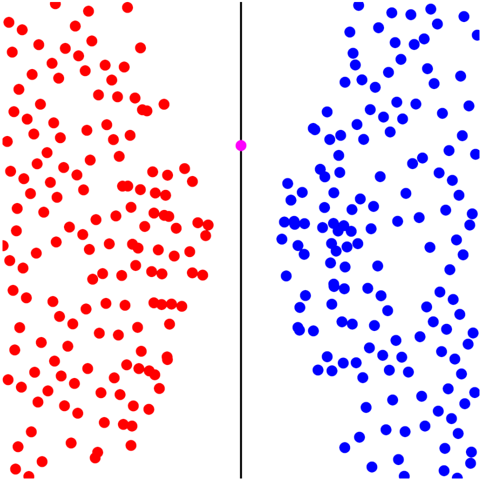
Ensemble learning II

- Which class (red or blue) should the magneta dot be?



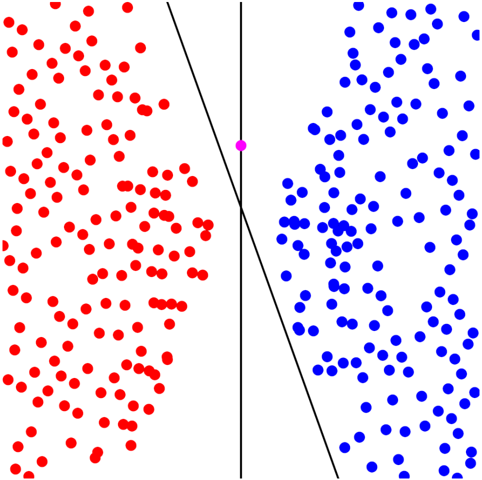
Ensemble learning II

- Which class (red or blue) should the magneta dot be?
- The obvious classification boundary (draw?)...



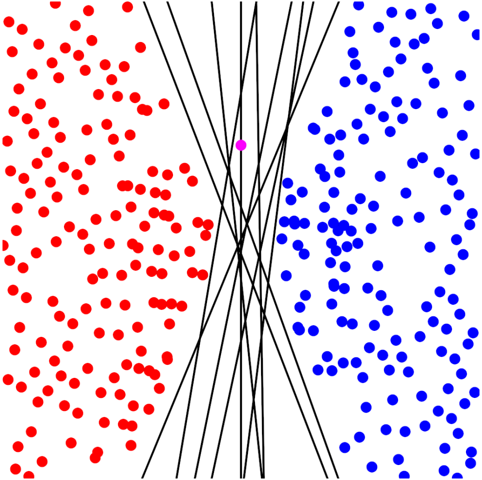
Ensemble learning II

- Which class (red or blue) should the magneta dot be?
- The obvious classification boundary (draw?)...
- But this is just as good (suggesting blue)



Ensemble learning II

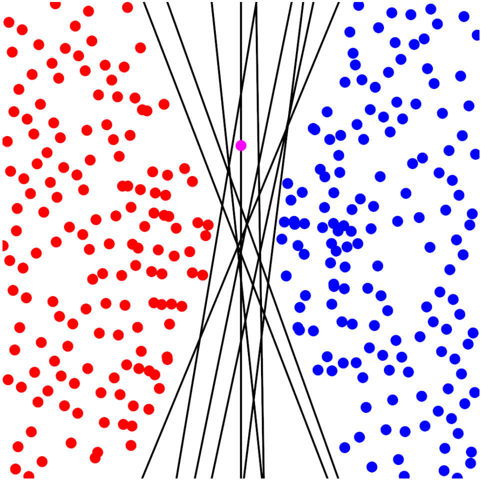
- Which class (red or blue) should the magneta dot be?
- The obvious classification boundary (draw?)...
- But this is just as good (suggesting blue)
- As are all of these!



Ensemble learning II

- Which class (red or blue) should the magneta dot be?
- The obvious classification boundary (draw?)...
- But this is just as good (suggesting blue)
- As are all of these!

- Models can be fit in many ways due to:
  - Insufficient data
  - Noisy data
  - Model not complex enough
    (curved boundaries can also separate this data!)

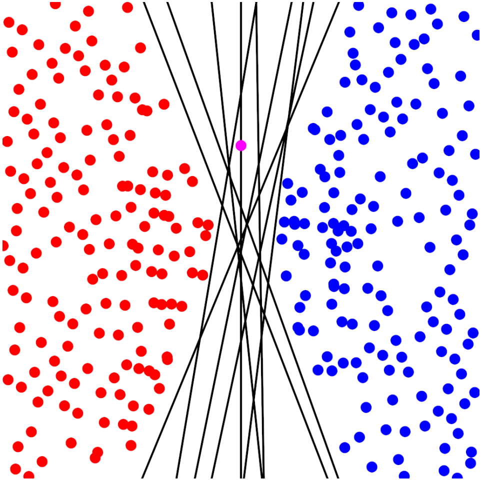


Ensemble learning II

- Which class (red or blue) should the magneta dot be?
- The obvious classification boundary (draw?)...
- But this is just as good (suggesting blue)
- As are all of these!

- Models can be fit in many ways due to:
  - Insufficient data
  - Noisy data
  - Model not complex enough
    (curved boundaries can also separate this data!)

- An ensemble generates many estimators...
  ...and hence captures this ambiguity.



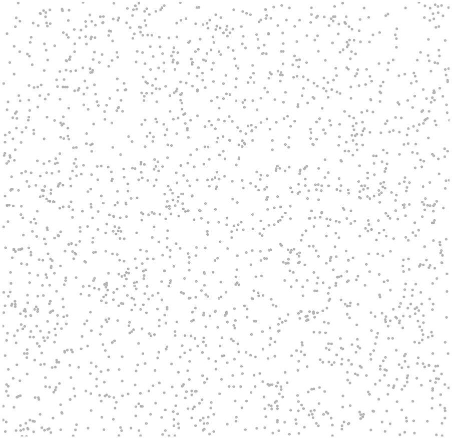
Ensemble learning III

- Ensembles need **diversity**, which is not well defined!
- Estimators should make different mistakes, i.e. if they all make the same mistake the ensemble will repeat it.
- Increasing estimator diversity at the expense of their individual performance often makes the ensemble better! (up to a limit)

An aside: Accuracy of an estimate

- Estimate a statistic of a population

  e.g. mean love of cheese within UK



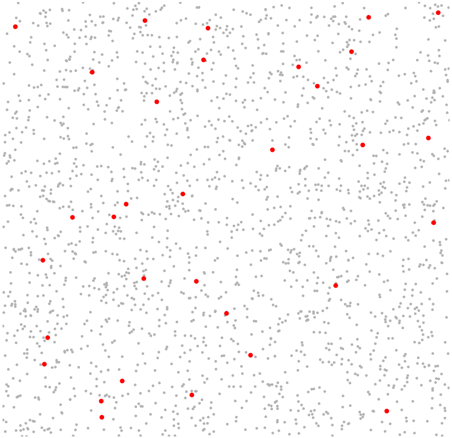
An aside: Accuracy of an estimate

- Estimate a statistic of a population

  e.g. mean love of cheese within UK

- Can't ask everyone so ask a subset...

  e.g. ask 32 people, get $\mu = 0.868$



An aside: Accuracy of an estimate

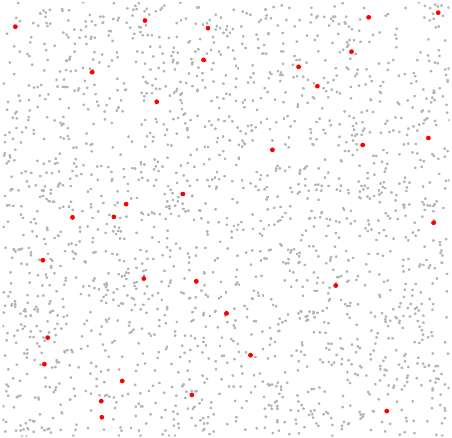
- Estimate a statistic of a population

  e.g. mean love of cheese within UK

- Can't ask everyone so ask a subset...

  e.g. ask 32 people, get $\mu = 0.868$

- But how accurate is this estimate?



An aside: Accuracy of an estimate

- Estimate a statistic of a population

  e.g. mean love of cheese within UK

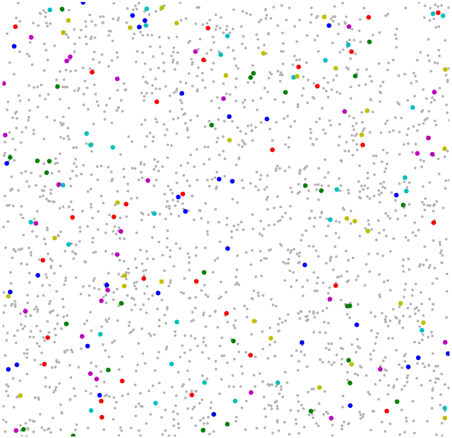
- Can't ask everyone so ask a subset...

  e.g. ask 32 people, get $\mu = 0.868$

- But how accurate is this estimate?

- Estimate as standard deviation of many repetitions of the experiment...
  - $\mu = \{0.868, 0.870, 0.885, 0.904, 0.891, 0.893\}$
  - $\mu^* = 0.885$, $\sigma^* = 0.013$

  (Numbers simulated/made up; true $\mu = 0.883$)



An aside: Accuracy of an estimate

- Estimate a statistic of a population

  e.g. mean love of cheese within UK

- Can't ask everyone so ask a subset...

  e.g. ask 32 people, get $\mu = 0.868$

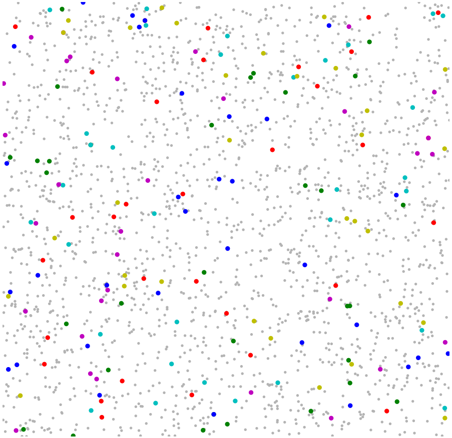
- But how accurate is this estimate?

- Estimate as standard deviation of many repetitions of the experiment...
  - $\mu = \{0.868, 0.870, 0.885, 0.904, 0.891, 0.893\}$
  - $\mu^* = 0.885$, $\sigma^* = 0.013$

  (Numbers simulated/made up; true $\mu = 0.883$)

- This is hardly practical!

  (prefer to add extra data to original experiment)



Bootstrapping

- Instead of collecting more data…

  …fake it from available data.

Bootstrapping

- Instead of collecting more data…
  …fake it from available data.
- Given data set of size n:
  Create new data set by drawing, with replacement, n times
  (will be repetitions)
- Can calculate statistic on each new data set, and estimate accuracy as variance.
  (or any other measure, e.g. a confidence interval)

Bagging

- An ensemble technique!
- Short for "bootstrap aggregating"
- Bagging = Bootstrapping applied to estimator output
  (via optimised parameters)

Bagging

- An ensemble technique!

- Short for "bootstrap aggregating"
- Bagging = Bootstrapping applied to estimator output
  (via optimised parameters)

- Algorithm:
  1. Select S, size of ensemble.
  2. Create S bootstrap draws of original data set.
  3. Train estimator on each.
  4. Combine outputs of all estimators for each query.

Random subspace method

- Another ensemble technique!
- Bootstrap applied to features
  i.e. fit each estimator with a random subset of features
- For decision tree: New bootstrap for each split.
- Remove the duplicates (makes no difference, saves computation)

Random forests

- Random forest = decision trees + bagging + random subspace method

- Algorithm:
  1. Select S, number of trees (more is better, up to a limit).
  2. Create S bootstrap draws of original data set.
  3. Train decision tree on each, with random subspace method
  4. Combine outputs of all trees for each query.

Combine output?

- Classification:
  - The decision trees vote – ensemble outputs winner.
- Regression:
  - Take the mean/median of all of the estimates.

Combine output?

- Classification:
  - The decision trees vote – ensemble outputs winner.
- Regression:
  - Take the mean/median of all of the estimates.
- Can generate probabilities!
  - Classification: Create histogram
  - Regression: Fit Gaussian distribution

A tweak

- Random subspace method selects around 63% of features.
- Can typically get away with much, much less (faster and performs better!).

- Instead: Select $\lceil \sqrt{f} \rceil$, where f is the number of features.
- Above good default; can optimise as hyperparameter.

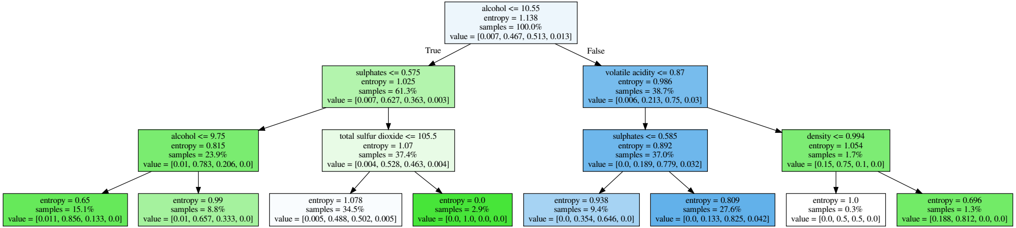
Example: Red wine I

- 1599 exemplars, split: 1199 train, 400 test.
- Input: 11 measurable features:
  - fixed acidity
  - volatile acidity
  - citric acid
  - residual sugar
  - chlorides
  - free sulfur dioxide
  - total sulfur dioxide
  - density
  - pH
  - sulphates
  - alcohol
- Output: 1–10 human rating

  (reduced to 1–5 here, to fit on screen)
- Can be used for classification or regression!

Example: Red wine II

Classification tree, max depth 3:

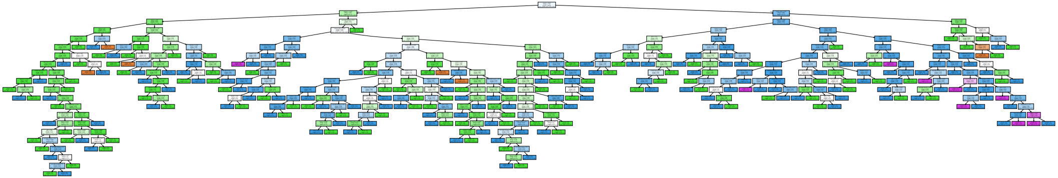


Accura

= 69% Explainable?

Example: Red wine III

Classification tree, unlimited depth:

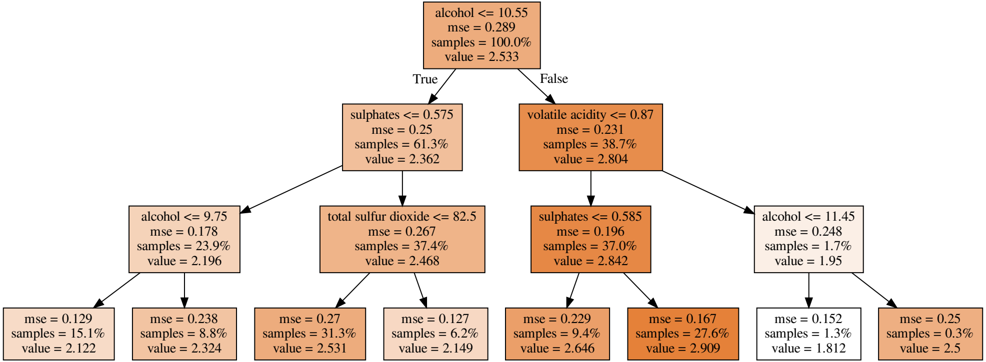


Accura

= 71% Explainable?

Example: Red wine IV

Regression tree, max depth 3:

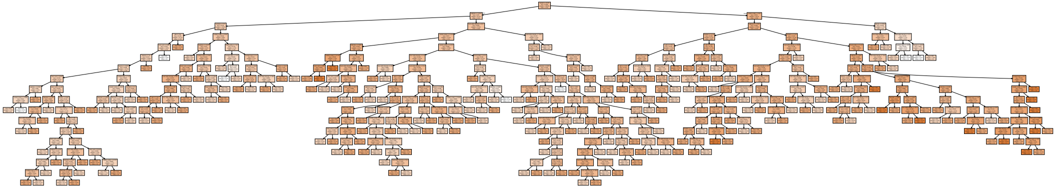


Mean

error $= 0.36$

Example: Red wine V

Regression tree, unlimited depth:



Mean

error = 0.33

Example: Red wine VI

- Random forests with 32 trees.
- Classification: Accuracy $= 79\%$ (better than 71%)
- Regression: Mean error $= 0.30$ (better than 0.33)
- Basically impossible to visualise/understand!

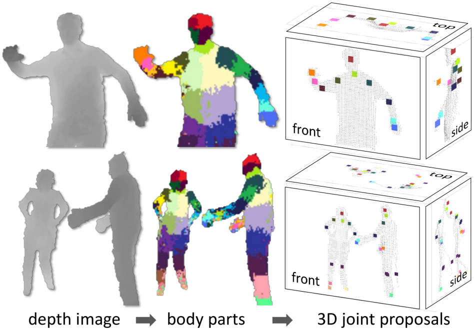
Example: Kinect I

- A depth sensor for gaming
- Tracks you as you move.
- Uses random forests!

Example: Kinect II

- Steps:
  1. Depth using structured light (infrared)
  2. Label body parts
  3. Fit skeleton

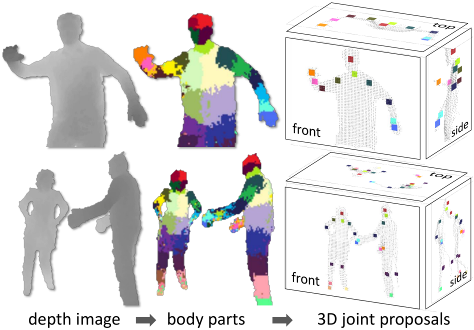
- Body part labelling:
  Classification forest run on every input pixel



Example: Kinect II

- Steps:
  1. Depth using structured light (infrared)
  2. Label body parts
  3. Fit skeleton

- Body part labelling:
  Classification forest run on every input pixel



Example: Kinect III

- Early example of **simulating** training data
  (now used extensively with deep learning)

- Collected:
  - 15 full body scans, labelled with body parts
  - 100K body poses from motion capture

- Used to generate one million random Kinect inputs. . .
  . . . with ground truth labels.
  (also randomised Kinect placement)
  (took 24 hours using 1000 node cluster)

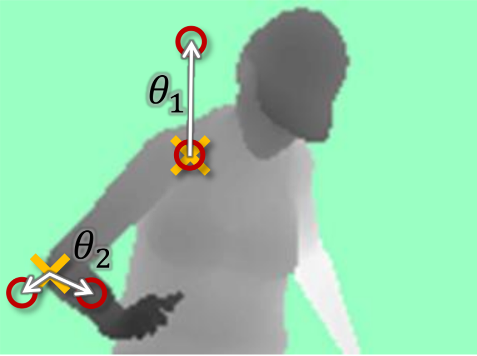
- Used real data for testing (8808 images)

Synthetic data for training:

Real data for testing:

Example: Kinect III

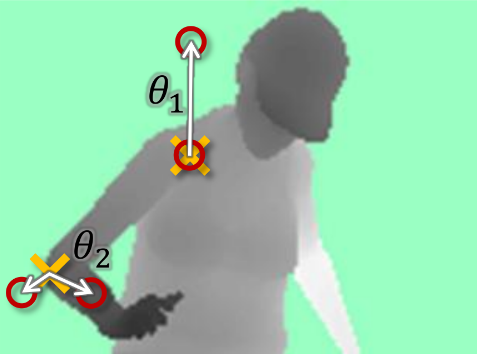
- Random forest:
  - 3 trees
  - 20 deep
  - 300K images per tree, each providing 2000 pixels.
  - Images unique for each tree – not bagging.
  - Considered 2000 features per split, each with 50 split points.



Example: Kinect III

- Random forest:
  - 3 trees
  - 20 deep
  - 300K images per tree, each providing 2000 pixels.
  - Images unique for each tree – not bagging.
  - Considered 2000 features per split, each with 50 split points.

- Features:
  - Depth differences between pixels
  - Two randomly generated offsets to make comparison between
  - This is done on the fly – implicit feature vector length is millions!



Summary

- “Upgraded” decision trees to
  - Handle all typical inputs
  - Handle all typical outputs
  - Random forests! (still one of the best)
  - Output probabilities
- Next lecture:

  Making sure a ML system is working!

Further reading

- The Kinect paper:
  “Real-Time Human Pose Recognition in Parts from Single Depth Images”
  by **Shotton, Fitzgibbon, Cook, Sharp and Finocchio**
  `https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/BodyPartRecognition.pdf`

- For more kinds of random forest:
  “Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning“ by **Criminisi, Shotton and Konukoglu**
  `http://research.microsoft.com/apps/pubs/default.aspx?id=155552`

- For more on ensemble methods:
  “Diversity creation methods: a survey and categorisation”
  by **Brown, Wyatt, Harris and Yao**
  `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.421.349&rep=rep1&type=pdf`

Legal

Kinect image from Wikipedia, in public domain, https://en.wikipedia.org/wiki/Kinect#/media/File:Xbox-360-Kinect-Standalone.png

Kinect algorithm images from "Real-Time Human Pose Recognition in Parts from Single Depth Images"; see previous slide, fair use.

Wine data set from https://archive.ics.uci.edu/ml/datasets/wine+quality