Decision Trees (2)

Many of these slides courtesy of Alex Thomo (thanks!)

Administrivia

- Identifying promising projects
  - Is this project easy to distribute amongst a team?
    - in groups of 5, you must minimize communication overhead
  - Does this project have a core task that seems do-able, as well as some more risky (and interesting!) extensions?
  - Does the data already exist? If not, are the resources to get/create the data available? A data mining project that gets stuck in the data phase won't be successful.

Last Time

- Building a decision tree

A criterion for attribute selection

- Which is the best attribute?
- The one which will result in the smallest tree
  - Heuristic: choose the attribute that produces the "purest" nodes
- Popular impurity criterion: entropy of nodes
  - **Lower the entropy, purer the node**.

Entropy

- H(X) = E(I(X)) Expected value of the information in X
- Expected value: $$\mathrm{E(f(X))} = \sum_i P(x_i) * f(x_i)$$
- Information: $$I(x_i) = -\log_2 P(x_i)$$
- Entropy: $$H(X) = E(I(X)) = \sum_i P(x_i) I(x_i) = -\sum_i P(x_i) \log_2 P(x_i)$$
- **Strategy: choose attribute that results in lowest entropy of the children nodes.**

Attribute "Temperature"

temperature=hot

entropy(2/4,2/4) = -2/4*log(2/4) -2/4*log(2/4) = 1

temperature=mild

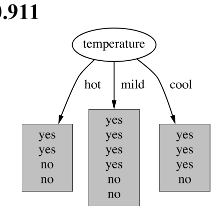
entropy(4/6,2/6) = -4/6*log(4/6) -2/6*log(2/6) = .918

temperature=cool

entropy(3/4,1/4) = -3/4*log(3/4)-1/4*log(1/4) = .811

Expected info:

AE = 1*(4/14) + .918*(6/14) + .811*(4/14) = **0.911**

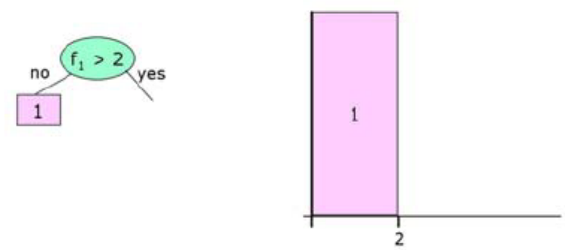


Decision Trees

- Pros/Cons?

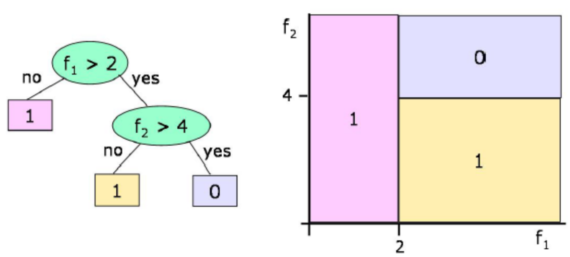
Numerical attributes revisited

- Tests in nodes are of the form f_i > constant



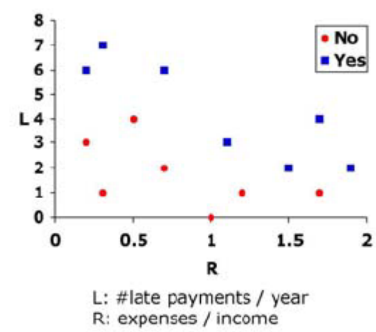
Numerical attributes

- Tests in nodes are of the form f_i > constant
- Divides the space into rectangles.



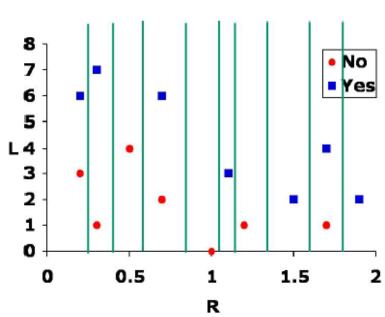
Predicting Bankruptcy

L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1.0	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



Considering splits

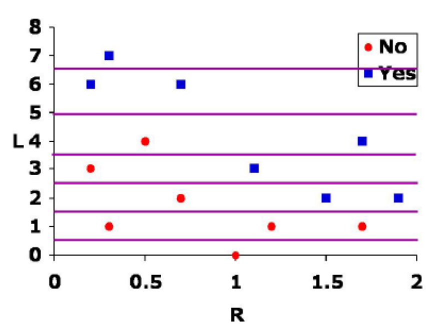
- Consider splitting between each data point in each dimension.



- So, here we'd consider 9 different splits in the R dimension

Considering splits II

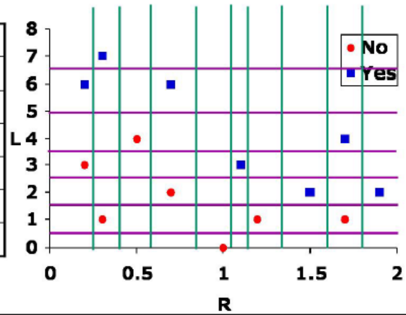
- And there are another 6 possible splits in the L dimension



Bankruptcy Example

L<y	NL	PL	NR	PR	AE
6.5	7	6	0	1	0.93
5.0	7	4	0	3	0.74
3.5	6	3	1	4	0.85
2.5	5	2	2	5	0.86
1.5	4	0	3	7	0.63
0.5	1	0	6	7	0.93

neg to left
pos to left
neg to right
pos to right

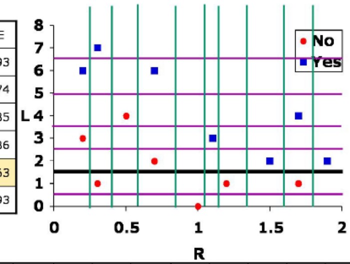


AE	1.00	1.00	0.98	0.98	0.94	0.98	0.92	0.98	0.92
R<x	0.25	0.40	0.60	0.85	1.05	1.15	1.35	1.60	1.80

Bankruptcy Example

- We consider all the possible splits in each dimension, and compute the average entropies of the children.

L<y	NL	PL	NR	PR	AE
6.5	7	6	0	1	0.93
5.0	7	4	0	3	0.74
3.5	6	3	1	4	0.85
2.5	5	2	2	5	0.86
1.5	4	0	3	7	0.63
0.5	1	0	6	7	0.93



AE	1.00	1.00	0.98	0.98	0.94	0.98	0.92	0.98	0.92
R<x	0.25	0.40	0.60	0.85	1.05	1.15	1.35	1.60	1.80

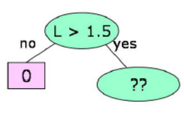
- And we see that, conveniently, all the points with L not greater than 1.5 are of class 0, so we can make a leaf there.

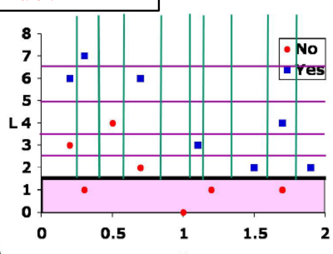
Bankruptcy Example

- Now, we consider all the splits of the remaining part of space.

Can we re-use the "R" table from the previous slide?

L<y	NL	PL	NR	PR	AE
6.5	6	3	0	1	0.83
5.0	4	3	0	3	0.69
3.5	3	2	4	1	0.85
2.5	2	1	5	2	0.88



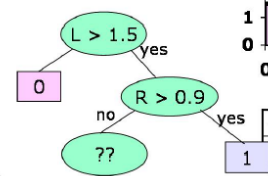


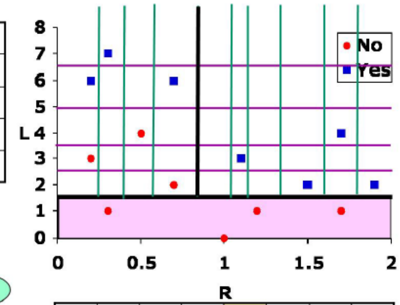
AE	0.85	0.88	0.79	0.60	0.69	0.76	0.83
R<x	0.25	0.40	0.60	0.90	1.30	1.60	1.80

Bankruptcy Example

- Now the best split is at R > 0.9.

L<y	NL	PL	NR	PR	AE
6.5	6	3	0	1	0.83
5.0	4	3	0	3	0.69
3.5	3	2	4	1	0.85
2.5	2	1	5	2	0.88

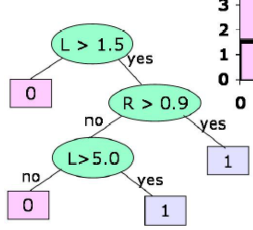


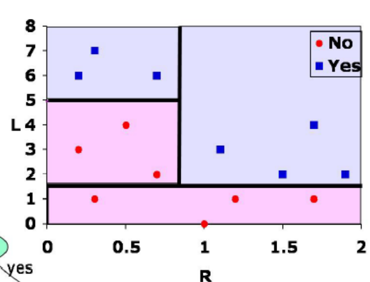


AD	0.85	0.88	0.79	0.60	0.69	0.76	0.83
R<x	0.25	0.40	0.60	0.90	1.30	1.60	1.80

Bankruptcy Example

- Continuing in this way, we finally obtain:





Alternative Splitting Criteria based on GINI

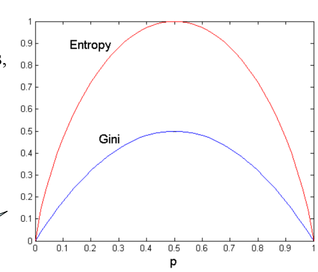
- We have used so far the entropy.
- GINI is an alternative.

$$Entropy(p_1,\dots,p_n) = -\sum_{i=1}^{n} p_i \log p_i$$

$$GINI(p_1,\dots,p_n) = 1 - \sum_{i=1}^{n} p_i^2$$

- Both, have:
  - Maximum when records are equally distributed among all classes, implying **least** purity
  - Minimum attained at p equal to 0 or 1, i.e. when all records belong to one class, implying most purity

For a 2-class problem:



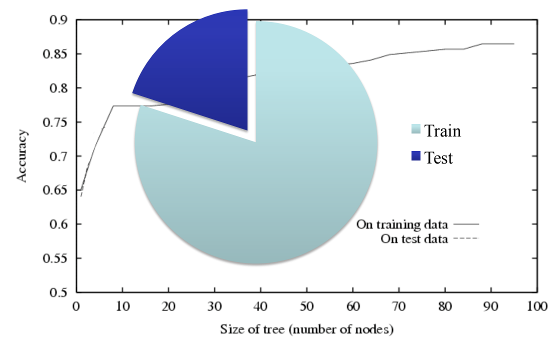
Pruning

- How can we reduce overfitting in our decision trees?
  - make the trees smaller (less complex)

Overfitting

- What is overfitting?

- What does overfitting look like?



Why is it overfitting?

- Where does overfitting come from?
  - noise in labels
  - noise in features
  - too little data (lack of representative data)
  - model-data mismatch

Back to Decision Trees…

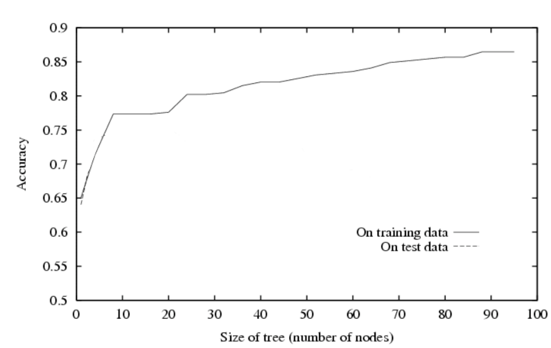
- How can we avoid overfitting?

Early Stopping (Pre-prune)

- Stop making splits when
  - average entropy doesn't change much
  - Predefined # of training instances reach leaf
  - Predefined depth
  - …? Others?

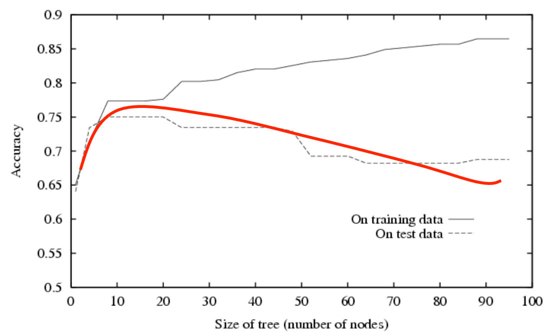
Post pruning

- Build a complex tree, then simplify
- You don't get to see the test set accuracy during training



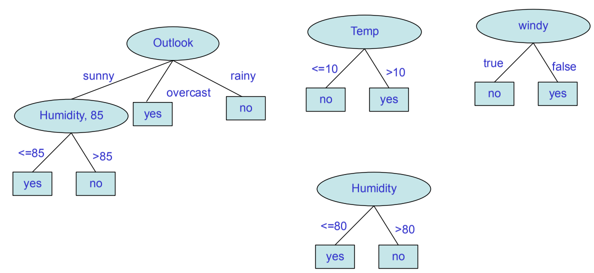
Validation Set

- Select some part of your training data for validation (tuning)
  - Choose depth based on validation performance peak
    - (hopefully similar to test data peak)



Random Forests

- Use multiple *simple* trees (i.e. a forest)
  - Consider only a *subset* of the data, *subset* of attributes
- Predict based on the majority prediction of all trees in forest



Random Forest Construction

Each tree is constructed using the following algorithm:

Input

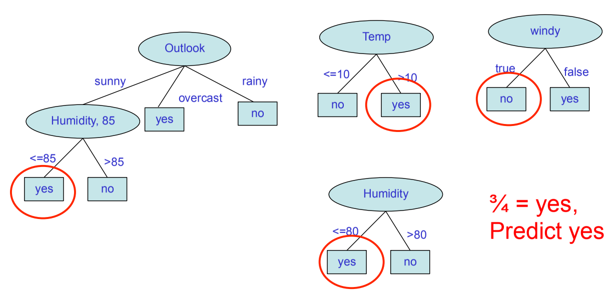
- N training cases with M attributes each.
- Number m (<M) of attributes to be used to determine the decision at a node of the tree
- Number n (<N) of training cases to be used for one tree.

Algorithm

- Choose a training set for this tree by choosing n times with replacement from all N available training cases.
- **For each node of the tree**, randomly choose m attributes on which to base the decision at that node. Calculate the best split based on these m attributes.
- Fully grow the tree.

Random Forest Prediction

- The new sample is pushed down a tree.
- It is assigned the label of the terminal node it ends up in.
- This procedure is iterated over all trees in the ensemble (forest), and the majority vote of all trees is reported as random forest prediction.
- Sunny, 15 degrees, windy = true, humidity = 75



Random Forests

- More robust to overfitting
  - why?
    - hint: trees are simple
  - The average of multiple simple (not overfit) classifiers tend not to combine to make an overfit classifier

  - more info http://kappa.math.buffalo.edu/aos.pdf

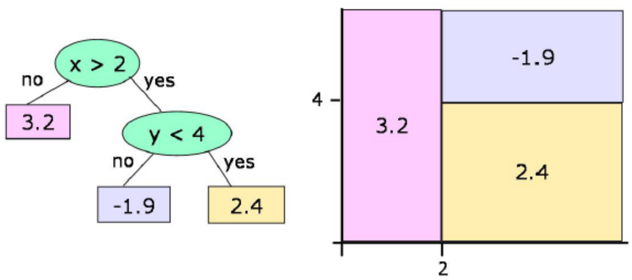
Classification vs Regression

- So far we have described classification
  - predicting one of a discrete set of labels
    - play tennis? yes/no
    - Neighbor’s behavior: walk/drive
    - Car type: luxury, mini, sports, van

- Sometime we want to predict a number in a range
  - E.g. age, forecast temperature, etc…
  - That’s regression (predicting a real number)

Regression Trees

- Like decision trees, but with real-valued constant outputs at the leaves.



Things to consider

- Prediction is a real number
  - Thus training labels are real numbers
  - Instead of {yes, yes, yes, no} at a leaf, we will have something like {0.1, 0.5, 1.3, 0.8}

- How to evaluate a candidate split?
- When to stop splitting?
- What to predict based on the instances in a leaf node?

Leaf values

- What to predict?
- Assume that multiple training points are in the leaf and we have decided, for whatever reason, to stop splitting.
  - In the boolean case, we used the **majority** output value
    - {yes, yes, yes, no} ➔ yes
  - In the numeric case?
    - {0.1, 0.5, 1.3, 0.8} ➔ ?
    - we'll use the **average** output value (here 0.675)

Splitting

- How to evaluate splits?
- We can't use entropy. Why?
- What could we use?
  - What is entropy measuring?
  - What's a similar measure for continuous values?

Variance

- Mean:

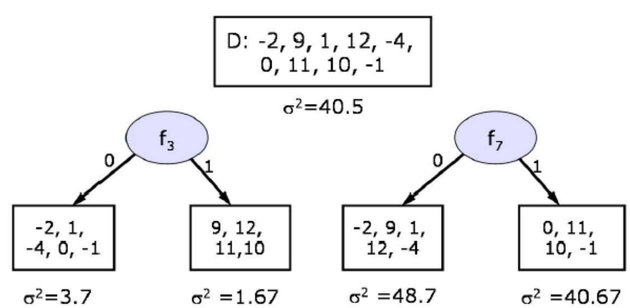
$$\mu = \frac{1}{m}\sum_{k=1}^{m} z_k$$

- Variance (unbiased estimator):

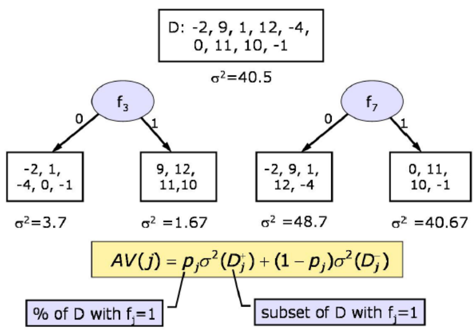
$$\sigma^2 = \frac{1}{m-1}\sum_{k=1}^{m}(z_k - \mu)^2$$

- We will use now the variance instead of entropy.

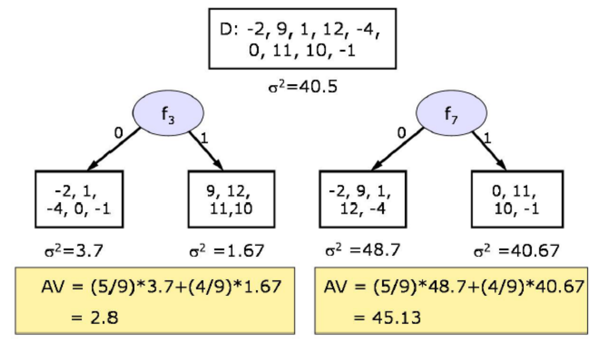
Splitting



Splitting



Splitting



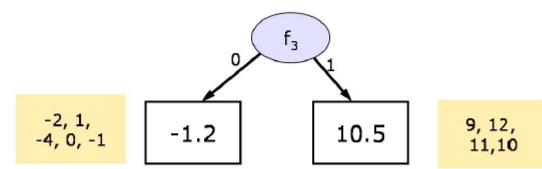
Leaf values

- When to stop splitting?
- So, if we're going to use the average value at a leaf as its output, we'd like to split up the data so that the leaf averages are not too far away from the actual items in the leaf.

- Again we will use variance

Stopping

- When to stop splitting?
- Stop when the variance at the leaf is “small enough”.
- Stop when the variance doesn’t change by much
- Then, set the value at the leaf to be the mean of the y values of the elements.



Scikit Learn

http://scikit-learn.org/stable/modules/tree.html