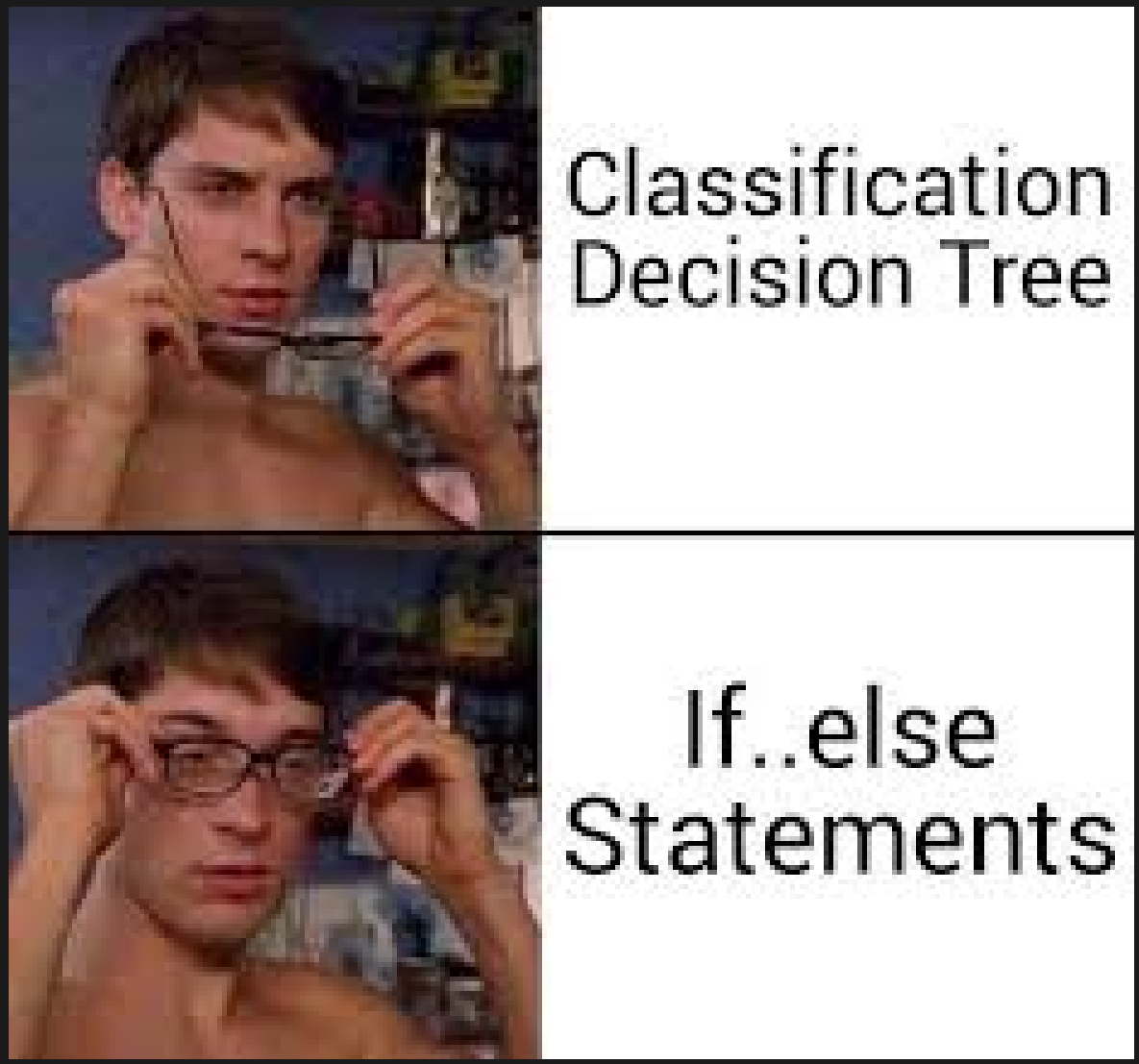
Minimising Losses for Decision Trees

Daniel Lawson University of Bristol

Lecture 03.1 (v4.0.0)



neuralnetmemes on IG

Signposting

- We'd like to make predictions about data
- Based on **decisions** for "which model to use"
- **Model** starts as a distribution over classes
- choosing the "best" model
- "Best" meaning: the one that minimises some **loss**

Trees, Forests, Decisions

- **Decision trees**: are extremely flexible and can fit highly non-linear spaces.
  - They can capture arbitrary complexity in the training data
  - They tend to overfit
  - They have overly-regular shapes, aligned to features
- **Random Forests**: Combining many random, decision trees
  - Randomization fights overfitting
  - Averaging creates smoother decision boundaries
  - Remarkable predictive performance.

Some data, before we start

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    features_pd, labels, train_size=0.8, test_size=0.2)
```

Decision Tree

- A decision tree is a sequence of conditionally evaluated tests:
- If c_1 then:
  - If c_{11} then:
    - ...
  - Else $!c_{11}$ so:
    - ...
- Else $!c_1$ so:
  - If c_{21} then:
    - ...
  - Else $!c_{21}$ so:
    - ...
- The conditions need to be chosen appropriately. How to decide?

Decision Tree Algorithm: CART

- **Classification and Regression Trees** (CART)[1]
- Consider a decision tree for C classes.
- For each **feature** $j \in [1, \cdots, J]$, we evaluate the **best-split** location in the feature space...
  - i.e. the one that **Minimises** the "Gini impurity" loss:

$$G_j = 1 - \sum_{c=1}^{C} p_{jc}^2 = \sum_{c=1}^{C} p_{jc}(1 - p_{jc}).$$

  - The assignment probability p_{jc} is the probability that class c is found at this leaf of the tree, if we use feature j.
  - We choose the "best" (induces many $p \to 0$ or $p \to 1$) feature as the next split.

---

[1] CART = Classification and Regression Trees. Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). Classification and regression trees. See Wei-Yin Loh's Review

Gini index example

```
def sq(y):
    return y * y
sq = np.vectorize(sq)
def gini(x):
    return 1-sq(x/x.sum()).sum()
```

Gini index example

```
test=pd.DataFrame()
test["size0"]=np.array([100,100,100])
test["size1.1"]=np.array([50,50,50])
test["size1.2"]=np.array([50,50,50])
test["size2.1"]=np.array([100,0,0])
test["size2.2"]=np.array([0,100,100])
```

Gini index example

```
test=pd.DataFrame()
test["size0"]=np.array([100,100,100])
test["size1.1"]=np.array([50,50,50])
test["size1.2"]=np.array([50,50,50])
test["size2.1"]=np.array([100,0,0])
test["size2.2"]=np.array([0,100,100])

print("Gini index initial value    =",gini(test["size0"]))
print("Gini reduction from split 1 =",gini(test["size0"]) -
  (gini(test["size1.1"])/2+gini(test["size1.2"])/2))
print("Gini reduction from split 2 =",gini(test["size0"]) -
  (gini(test["size2.1"])*1/3+gini(test["size2.2"])*2/3))
```

Gini index example

```
test=pd.DataFrame()
test["size0"]=np.array([100,100,100])
test["size1.1"]=np.array([50,50,50])
test["size1.2"]=np.array([50,50,50])
test["size2.1"]=np.array([100,0,0])
test["size2.2"]=np.array([0,100,100])

print("Gini index initial value    =",gini(test["size0"]))
print("Gini reduction from split 1 =",gini(test["size0"]) -
  (gini(test["size1.1"])/2+gini(test["size1.2"])/2))
print("Gini reduction from split 2 =",gini(test["size0"]) -
  (gini(test["size2.1"])*1/3+gini(test["size2.2"])*2/3))
```

```
Gini index initial value    = 0.666666666667
Gini reduction from split 1 = 0.0
Gini reduction from split 2 = 0.333333333334
```

Decision Tree Algorithm: ID3

- But is Gini Index the *right* loss?
- ID3[2] (Iterative Dichotomiser 3) **Maximises** the "information gain", by **Minimising** the loss:

$$H_j = -\sum_{c=1}^{C} p_{jc} \log(p_{jc})$$

  - The difference is how each probability is weighted.
  - We still define "best" feature as one that makes many $p \to 0$ and $p \to 1$.
  - Gini punishes large absolute-value mistakes whilst information punishes large log-scale mistakes.
  - The difference is important for a tree but usually unimportant for the classifier.

[2] ID = Iterative Dichotomiser. Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81-106.

ID3

```
def ylogy(y): # CAREFUL!
    ty=[max(1e-10,x) for x in y]
    return y * np.log(ty)
def id3(x):
return -ylogy(x/x.sum()).sum()
```

ID3

```
def ylogy(y): # CAREFUL!
    ty=[max(1e-10,x) for x in y]
    return y * np.log(ty)
def id3(x):
return -ylogy(x/x.sum()).sum()

print("ID3 index initial value    =",id3(test["size0"]))
print("ID3 reduction from split 1 =",id3(test["size0"]) -
  (id3(test["size1.1"])/2+id3(test["size1.2"])/2))
print("ID3 reduction from split 2 =",id3(test["size0"]) -
  (id3(test["size2.1"])*1/3+id3(test["size2.2"])*2/3))
```

ID3

```
def ylogy(y): # CAREFUL!
    ty=[max(1e-10,x) for x in y]
    return y * np.log(ty)
def id3(x):
return -ylogy(x/x.sum()).sum()

print("ID3 index initial value    =",id3(test["size0"]))
print("ID3 reduction from split 1 =",id3(test["size0"]) -
  (id3(test["size1.1"])/2+id3(test["size1.2"])/2))
print("ID3 reduction from split 2 =",id3(test["size0"]) -
  (id3(test["size2.1"])*1/3+id3(test["size2.2"])*2/3))

ID3 index initial value    = 1.0986122886681096
ID3 reduction from split 1 = 0.0
ID3 reduction from split 2 = 0.6365141682948128
```

Signposting

- **References**:
- Tree methods:
  - Chapter 9.2 of The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Friedman, Hastie and Tibshirani).
  - Penn State U Applied Data Mining and Statistical Learning How to prune trees
  - Decision Tree Algorithms: Deep Math ML
- Regression Trees:
  - Karalic A, "Employing Linear Regression in Regression Tree Leaves" (1992) ECAI-92

Signposting

- **References (2)**:
- Boosted Decision Trees:
  - J. Elith, J. Leathwick, and T. Hastie "A working guide to boosted regression trees" (2008). British Ecological Society.
- CART:
  - CART = Classification and Regression Trees. Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). Classification and regression trees.
  - Wei-Yin Loh's 2011 Review is popular.
- ID3: Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81-106.

Signposting

- In the practical we'll implement these models in R and Python; compare implementations, and to previous results.
- Next semester we'll start with the "other" LDA (Latent Dirichlet Allocation), Topic Modelling, and Modelling Documents.
- **References**:
  - Chapter 15 of The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Friedman, Hastie and Tibshirani).
  - Implement a Random Forest From Scratch in Python
  - A Gentle Introduction to Random Forests at CitizenNet
  - DataDive on Selecting good features
  - Cosma Shalizi on Regression Trees
  - Gilles Louppe PhD Thesis: Understanding Random Forests