DMML, 31 Jan 2019

Supervised learning — Predict from training data

- Classification problem

Sometimes — need to predict a numeric value

- Exam marks
- Salary, mileage for a car, ...

Attributes: A_i ← numeric ✓
categorical

Mileage of a car: Weight of car, engine size, ...

Exam marks: Exam marks in practice tests, IQ score, height, weight, dob

how to rule these out?

Not today

Model $f(a_1, \ldots, a_k) = y$

Variables $x_1 \ldots x_k$ for k input attributes

Calculate a function $f(x_1 \ldots x_k)$ that approximates output well

Assume linear relationship (by inspection?)

$$y = \theta_0 + \theta_1 x_1 + \ldots + \theta x_k$$

Learning = Parameter Estimation

- Find θ_i

Find "best" set of θ_i

Measure of "goodness"

Cost function / loss function

Measure of how far the current estimate deviate from training data

Training input $(\bar{x}_j, y_j)$

Compare $f(\bar{x}_j)$ & y_j - Accumulate over all $\bar{x}_j$

Simple difference $f(x_j) - y_j$ may cancel

Should at least take $|f(x_j) - y_j|$

Theoretically, squared error is best

$$\frac{1}{2} \sum_j \left(f(x_j) - y_j \right)^2$$

Constant factor does not affect the nature of the cost

Repeatedly adjust θ_i so that cost reduces

In Statistics – can explicitly calculate best θ_i using mean, variance etc

How to iterate θ_i?

Initial estimate $E_0 = \langle \theta_0^0, \theta_1^0, \ldots, \theta_k^0 \rangle$

$E_0 \rightarrow C_0 \xrightarrow{\text{adjust}} E_1 \rightarrow C_1 \cdots$

How to go from C_j to E_{j+1}?

Gradient descent (greedy - reduce cost as much as possible)

For each parameter θ_j, compute $\frac{\partial C}{\partial \theta_j}$

Adjust θ_j to $\theta_j - \alpha \frac{\partial C}{\partial \theta_j}$

α: step factor

Cost is $\frac{1}{2} \sum_{k=1}^{N} \left(f(x_k) - y_k\right)^2$

$$f(\bar{x}_k) = \theta_0 + \theta_1 x_1^k + \theta_2 x_2^k + \cdots + \theta_m x_m^k$$

$$\frac{\partial f(\bar{x}_k)}{\partial \theta_j} = \bar{x}_j^k$$

$$\frac{\partial}{\partial \theta_j} \frac{1}{2} \left(f(\bar{x}_k) - y \right)^2$$

$$\frac{1}{2} \cdot 2 \left(f(\bar{x}_k) - y \right)_k \cdot \frac{\partial f(\bar{x}_k)}{\partial \theta_j} x_j$$

$$\theta_j^{k+1} = \theta_j^k + \alpha \sum_{i=1}^{i} (y_i - f(\bar{x}_i) \cdot \bar{x}_i$$

Finding optimal θ_i — regression

Start with some initial estimate E_0

Calculate $f(\bar{x}_j)$ for all j & compute gradient for each coefficient

Update each θ_j as above

Batch gradient descent

Update θ_j's after a full batch of predictions

Adjustment of θ_j is proportional to error $f(x_j) - y_j)$

Another option

Update θ_j after <u>each</u> input

Stochastic Gradient Descent

- More unstable
- Overall converges faster

Overfitting

Regularization — keep coefficients simple

Add to cost, an extra penalty proportional to size of coefficients

Decision trees suffer from variance

Small input perturbation → big change in output

Regression suffers from bias

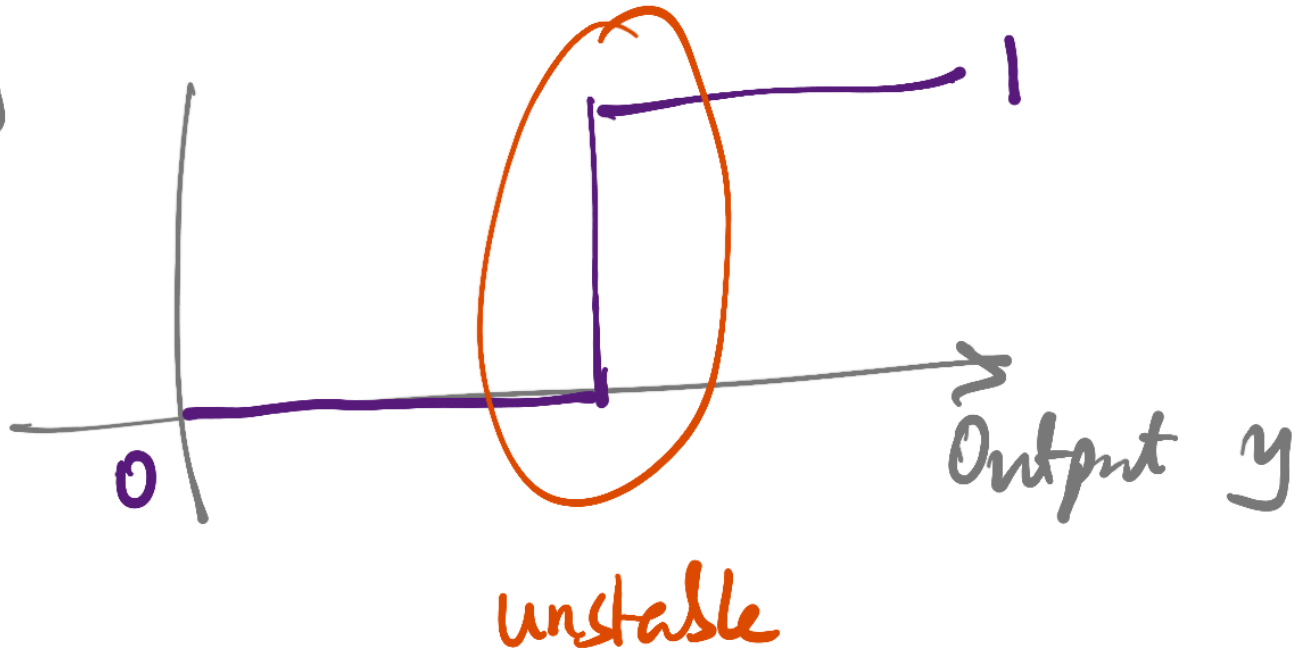
Intentionally limiting the nature of $f(x)$

Model chosen is not expressive enough

Regression for classification

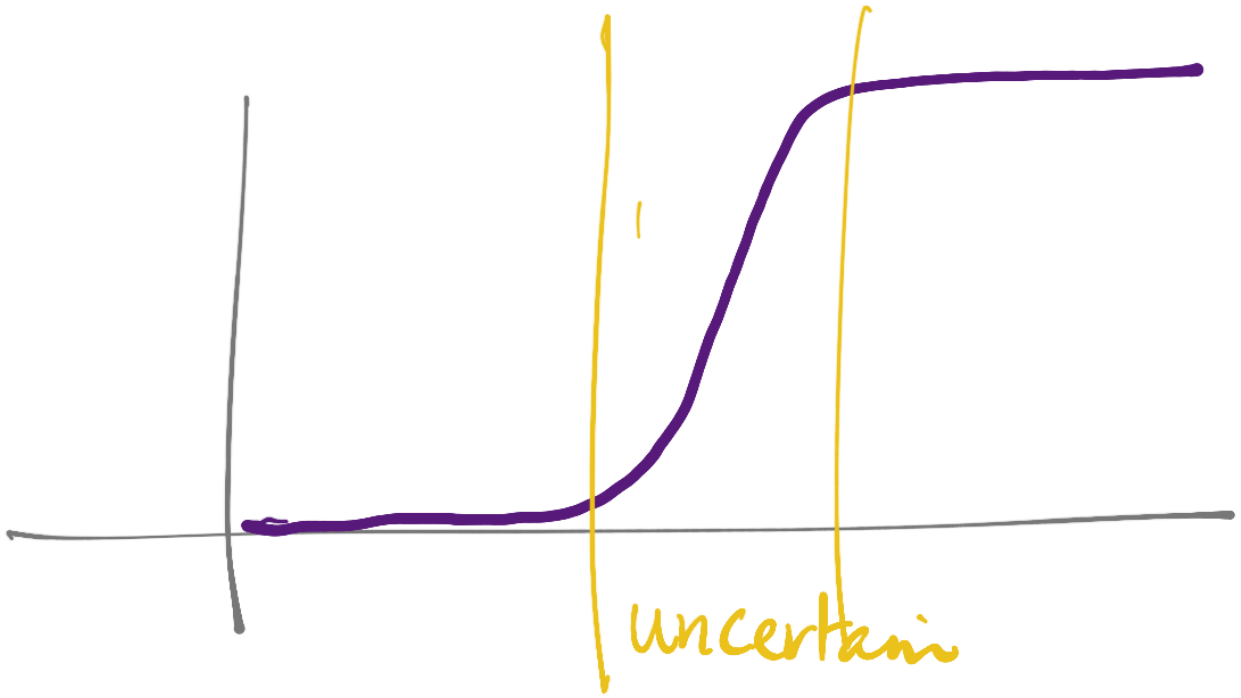
Classification → predict 0 or 1

Typically



unstable

Instead, a smooth approximation of step



Sigmoid →

$$\sigma(x) = \frac{1}{1+e^{-\bar{\theta}^T \bar{x}}}$$

$\vec{\theta}^T$ $\vec{x}$

$$\begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_m \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \\ \\ x_m \end{bmatrix}$$

Still use square-error for cost

$$\frac{\partial C}{\partial \theta_j} = (f(x_i) - y_i) \cdot \frac{\partial f}{\partial \theta_j}$$

can be calculated

$$\sigma(z) = \frac{1}{1+e^{-\theta z}}$$

$$\sigma'(z) = \sigma(z)\,(1 - \sigma(z))$$

Logistic regression

Sigmoid is not only option for smooth step

Square is not only option for cost

Regression using decision trees

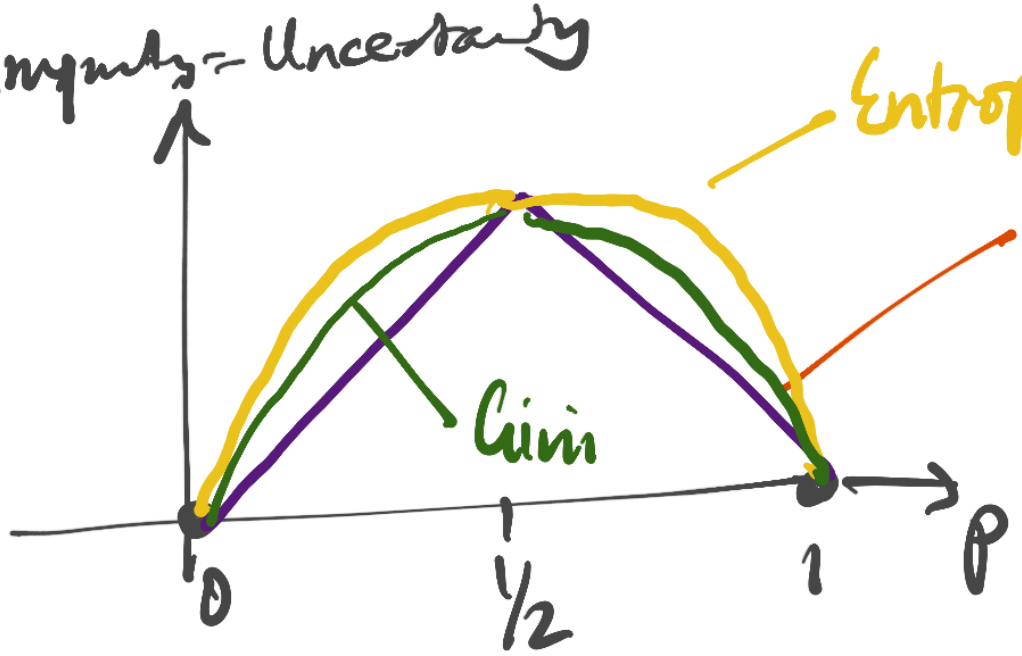
R, Python — CART

Classification & Regression Tree

CART

- Classification - same as a decision tree

$1 - \sum p_i^2$ - Gini index instead of entropy



Measuring Impurity

= misclassification rate

- Restrict to binary branching

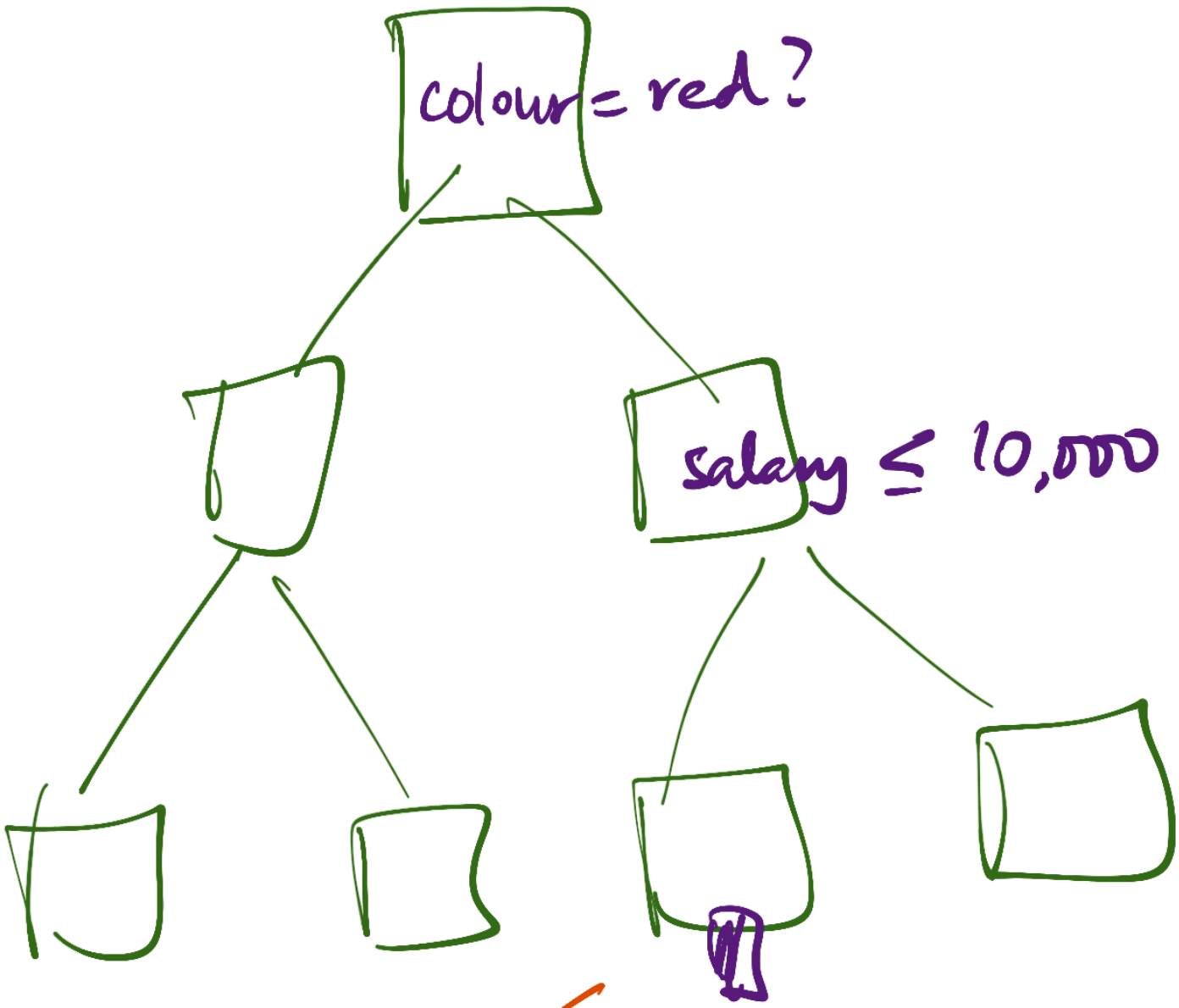
Regression trees

① - Which attribute/question to ask next

② - How to evaluate cost/impurity

③ - What value to assign at a leaf

3



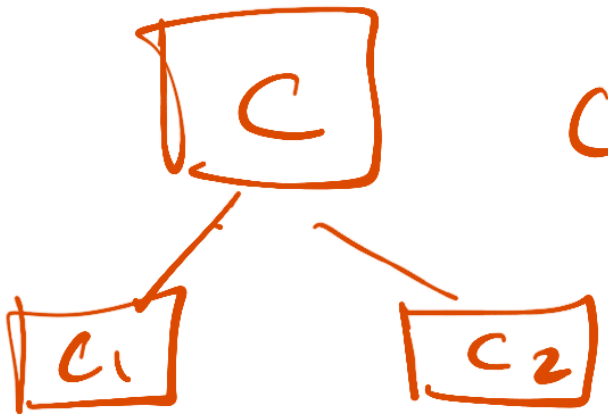
Take mean of all y_i's in this node

(2) & (1) Use the usual cost function

$f(x)$ is the average for that node

$$\sum (y_{mean} - y_i)^2$$

Choose question to reduce cost



$C - (C_1 + C_2)$ to be maximized