Lecture 8: 6 February, 2025

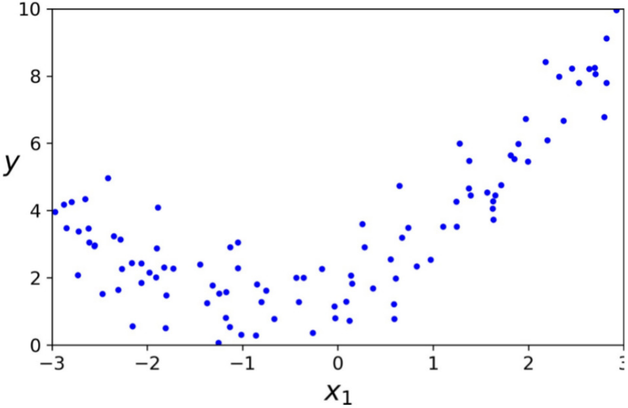
Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Data Mining and Machine Learning
January–April 2025

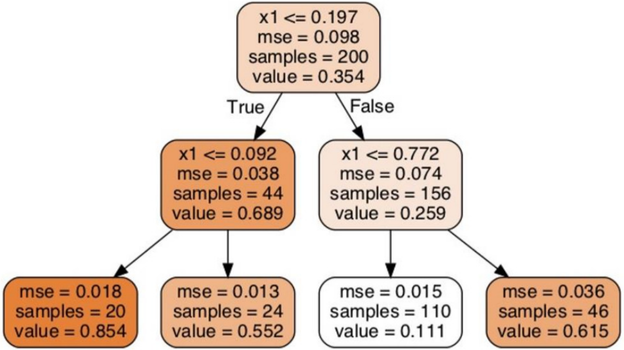
Decision trees for regression

- Can we use decision trees for regression?
- Partition the input into intervals
- For each interval, predict mean value of output, instead of majority class
- Regression tree



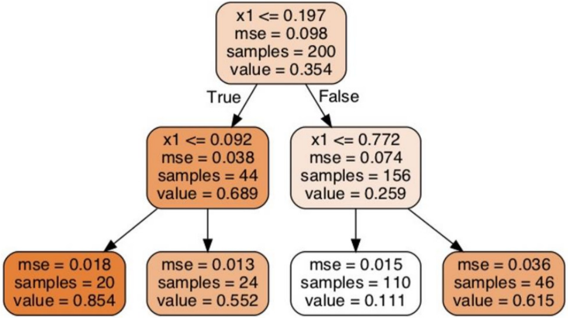
Decision trees for regression

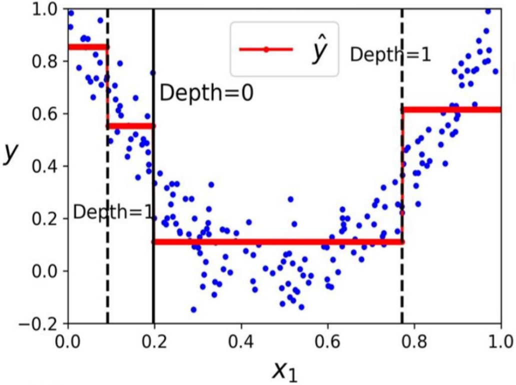
- Regression tree for noisy quadratic centered around $x_1 = 0.5$
- For each node, the output is the mean y value for the current set of points
- Instead of impurity, use mean squared error (MSE) as cost function
- Choose a split that minimizes MSE



Regression trees

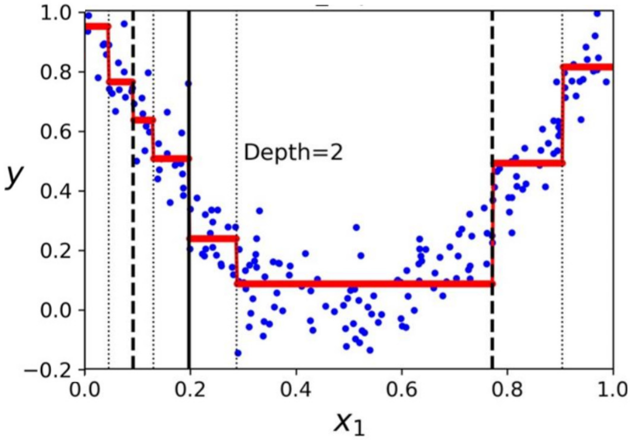
- Approximation using regression tree





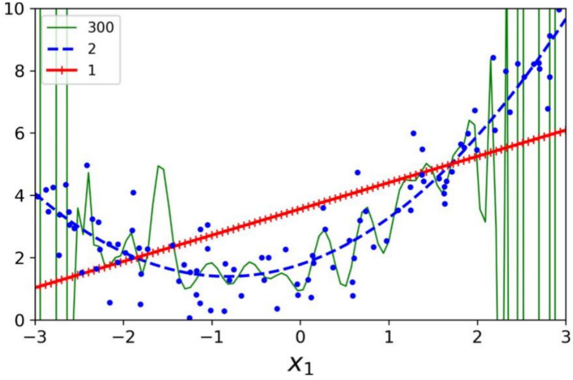
Regression trees

- Extend the regression tree one more level to get a finer approximation
- Set a threshold on MSE to decide when to stop
- Classification and Regression Trees (CART)
  - Combined algorithm for both use cases
- Programming libraries typically provide CART implementation



Overfitting

- Overfitting: model too specific to training data, does not generalize well
- Regression — use regularization to penalize model complexity
- What about decision trees?
- Deep, complex trees ask too many questions
- Prefer shallow, simple trees

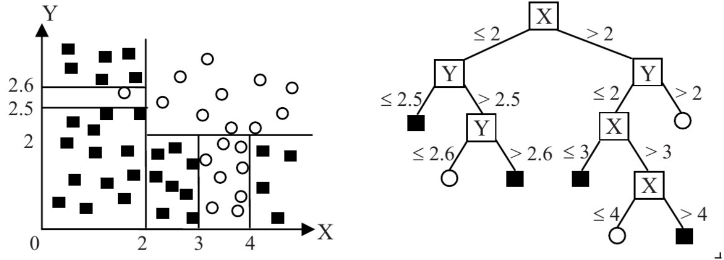


Tree pruning

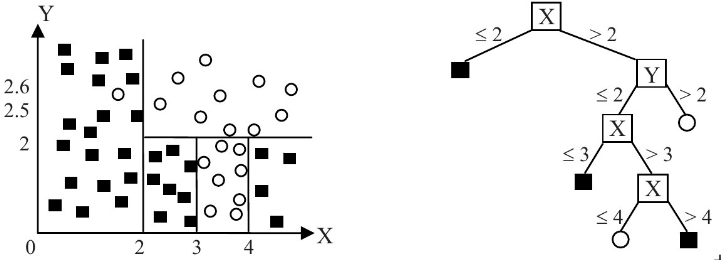
- Remove leaves to improve generalization
- Top-down pruning
  - Fix a maximum depth when building the tree
  - How to decide the depth in advance?
- Bottom-up pruning
  - Build the full tree
  - Remove a leaf if the reduced tree generalizes better
  - How do we measure this?

Tree pruning

Overfitted tree



Pruned tree



Bottom up tree pruning

- Build the full tree, remove leaf if the reduced tree generalizes better
- How do we measure this?
- Check performance on a test set
- Use sampling theory [Quinlan]
- Given n coin tosses with h heads, estimate probability of heads as h/n
  - Estimate comes with a confidence interval: $h/n \pm \delta$
  - As n increases, δ reduces: 7 heads out of 10 vs 70 out of 100 vs 700 out of 1000
- Impure node, majority prediction, compute confidence interval
- Pruning leaves creates a larger impure sample one level above
- Does the confidence interval decrease (improve)?

Example: Predict party from voting pattern [Quinlan]

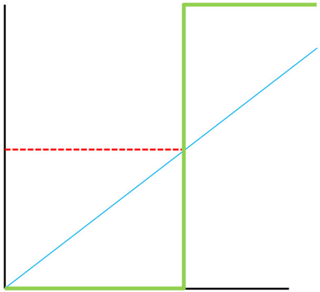
- Predict party affiliation of US legislators based on voting pattern
    - Read the tree from left to right
- After pruning, drastically simplified tree
- Quinlan's comment on his use of sampling theory for post-pruning

  Now, this description does violence to statistical notions of sampling and confidence limits, so the reasoning should be taken with a large grain of salt. Like many heuristics with questionable underpinnings, however, the estimates it produces seem frequently to yield acceptable results.

```
physician fee freeze = n:
|   adoption of the budget resolution = y: democrat (151)
|   adoption of the budget resolution = u: democrat (1)
|   adoption of the budget resolution = n:
|   |   education spending = n: democrat (6)
|   |   education spending = y: democrat (9)
|   |   education spending = u: republican (1)
physician fee freeze = y:
|   synfuels corporation cutback = n: republican (97/3)
|   synfuels corporation cutback = u: republican (4)
|   synfuels corporation cutback = y:
|   |   duty free exports = y: democrat (2)
|   |   duty free exports = u: republican (1)
|   |   duty free exports = n:
|   |   |   education spending = n: democrat (5/2)
|   |   |   education spending = y: republican (13/2)
|   |   |   education spending = u: democrat (1)
physician fee freeze = u:
|   water project cost sharing = n: democrat (0)
|   water project cost sharing = y: democrat (4)
|   water project cost sharing = u:
|   |   mx missile = n: republican (0)
|   |   mx missile = y: democrat (3/1)
|   |   mx missile = u: republican (2)
```

Regression for classification

- Regression line
- Set a threshold
- Classifier
  - Output below threshold : 0 (No)
  - Output above threshold : 1 (Yes)
- Classifier output is a step function



Smoothen the step

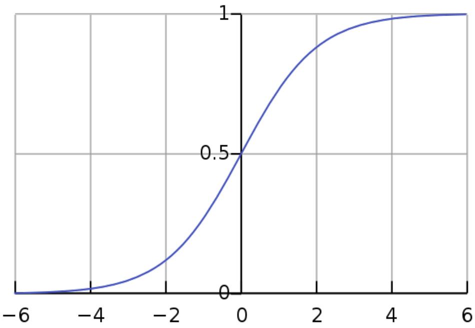
- Sigmoid function

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

- Input z is output of our regression

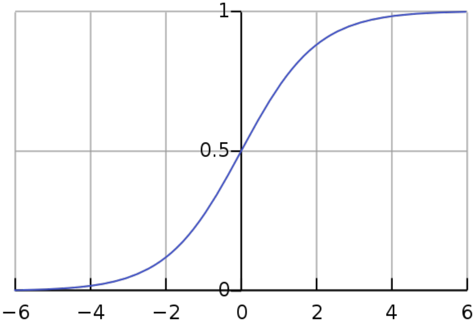
$$\sigma(z) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \cdots + \theta_k x_k)}}$$

- Adjust parameters to fix horizontal position and steepness of step



Logistic regression

- Compute the coefficients?
- Solve by gradient descent
- Need derivatives to exist
  - Hence smooth sigmoid, not step function
  - $\sigma'(z) = \sigma(z)(1 - \sigma(z))$
- Need a cost function to minimize



Loss function for logistic regression

- Goal is to maximize log likelihood
- Let $h_\theta(x_i) = \sigma(z_i)$. So, $P(y_i = 1 \mid x_i; \theta) = h_\theta(x_i)$,
  $P(y_i = 0 \mid x_i; \theta) = 1 - h_\theta(x_i)$
- Combine as $P(y_i \mid x_i; \theta) = h_\theta(x_i)^{y_i} \cdot (1 - h_\theta(x_i))^{1-y_i}$
- Likelihood: $\mathcal{L}(\theta) = \prod_{i=1}^{n} h_\theta(x_i)^{y_i} \cdot (1 - h_\theta(x_i))^{1-y_i}$
- Log-likelihood: $\ell(\theta) = \sum_{i=1}^{n} y_i \log h_\theta(x_i) + (1 - y_i) \log(1 - h_\theta(x_i))$
- Minimize cross entropy: $-\sum_{i=1}^{n} y_i \log h_\theta(x_i) + (1 - y_i) \log(1 - h_\theta(x_i))$

MSE for logistic regression and gradient descent

- Suppose we take mean sum-squared error as the loss function.
- Consider two inputs $x = (x_1, x_2)$

$$C = \frac{1}{n}\sum_{i=1}^{n}(y_i - \sigma(z_i))^2, \text{ where } z_i = \theta_0 + \theta_1 x_{i_1} + \theta_2 x_{i_2}$$

- For gradient descent, we compute $\frac{\partial C}{\partial \theta_1}, \frac{\partial C}{\partial \theta_2}, \frac{\partial C}{\partial \theta_0}$
  - For $j = 1, 2$,

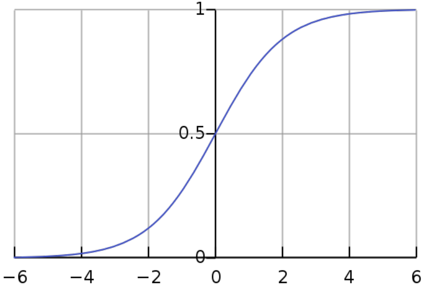
$$\frac{\partial C}{\partial \theta_j} = \frac{2}{n}\sum_{i=1}^{n}(y_i - \sigma(z_i)) \cdot -\frac{\partial \sigma(z_i)}{\partial \theta_j} = \frac{2}{n}\sum_{i=1}^{n} (\sigma(z_i) - y_i)\ \frac{\partial \sigma(z_i)}{\partial z_i}\frac{\partial z_i}{\partial \theta_j}$$

$$= \frac{2}{n}\sum_{i=1}^{n}(\sigma(z_i) - y_i)\sigma'(z_i)x_{i_j}$$

  - $\frac{\partial C}{\partial \theta_0} = \frac{2}{n}\sum_{i=1}^{n}(\sigma(z_i) - y_i)\frac{\partial \sigma(z_i)}{\partial z_i}\frac{\partial z_i}{\partial \theta_0} = \frac{2}{n}\sum_{i=1}^{n}(\sigma(z_i) - y_i)\sigma'(z_i)$

MSE for logistic regression and gradient descent . . .

- For $j = 1, 2$, $\dfrac{\partial C}{\partial \theta_j} = \dfrac{2}{n}\displaystyle\sum_{i=1}^{n}(\sigma(z_i) - y_i)\sigma'(z_i)x_j^i$, and $\dfrac{\partial C}{\partial \theta_0} = \dfrac{2}{n}\displaystyle\sum_{i=1}^{n}(\sigma(z_i) - y_i)\sigma'(z_i)$
- Each term in $\dfrac{\partial C}{\partial \theta_1}, \dfrac{\partial C}{\partial \theta_2}, \dfrac{\partial C}{\partial \theta_0}$ is proportional to $\sigma'(z_i)$
- Ideally, gradient descent should take large steps when $\sigma(z) - y$ is large
- $\sigma(z)$ is flat at both extremes
- If $\sigma(z)$ is completely wrong, $\sigma(z) \approx (1 - y)$, we still have $\sigma'(z) \approx 0$
- Learning is slow even when current model is far from optimal



Cross entropy and gradient descent

- $C = -[y \ln(\sigma(z)) + (1-y)\ln(1-\sigma(z))]$

- $$\frac{\partial C}{\partial \theta_j} = \frac{\partial C}{\partial \sigma}\frac{\partial \sigma}{\partial \theta_j} = -\left[\frac{y}{\sigma(z)} - \frac{1-y}{1-\sigma(z)}\right]\frac{\partial \sigma}{\partial \theta_j}$$
$$= -\left[\frac{y}{\sigma(z)} - \frac{1-y}{1-\sigma(z)}\right]\frac{\partial \sigma}{\partial z}\frac{\partial z}{\partial \theta_j}$$
$$= -\left[\frac{y}{\sigma(z)} - \frac{1-y}{1-\sigma(z)}\right]\sigma'(z)x_j$$
$$= -\left[\frac{y(1-\sigma(z)) - (1-y)\sigma(z)}{\sigma(z)(1-\sigma(z))}\right]\sigma'(z)x_j$$

Cross entropy and gradient descent . . .

- $\dfrac{\partial C}{\partial \theta_j} = -\left[\dfrac{y(1-\sigma(z)) - (1-y)\sigma(z)}{\sigma(z)(1-\sigma(z))}\right]\sigma'(z)x_j$
- Recall that $\sigma'(z) = \sigma(z)(1-\sigma(z))$
- Therefore, $$\begin{aligned}\frac{\partial C}{\partial \theta_j} &= -[y(1-\sigma(z)) - (1-y)\sigma(z)]x_j \\ &= -[y - y\sigma(z) - \sigma(z) + y\sigma(z)]x_j \\ &= (\sigma(z) - y)x_j\end{aligned}$$
- Similarly, $\dfrac{\partial C}{\partial \theta_0} = (\sigma(z) - y)$
- Thus, as we wanted, the gradient is proportional to $\sigma(z) - y$
- The greater the error, the faster the learning rate