University of Manchester AI ML Society - Introduction to ML

Workshop 3
6th of November 2019

By Botond Maros

Intro to ML timetable

Workshop 1 - Introduction to Machine Learning

Workshop 2 - Data preprocessing

Workshop 3 - Fundamental Algorithms I

Workshop 4 - Fundamental Algorithms II

Workshop 5 - Neural Networks Part I

Workshop 6 - Neural Networks Part II

Today’s session

- Linear regression
- Logistic regression
- Gradient descent
- Decision tree

What are we doing?

- Come up with a model that describes data
- Measure how it describes the data
- Calculate error = difference of real data vs model
- Find model with lowest error

Two main types of ML algos

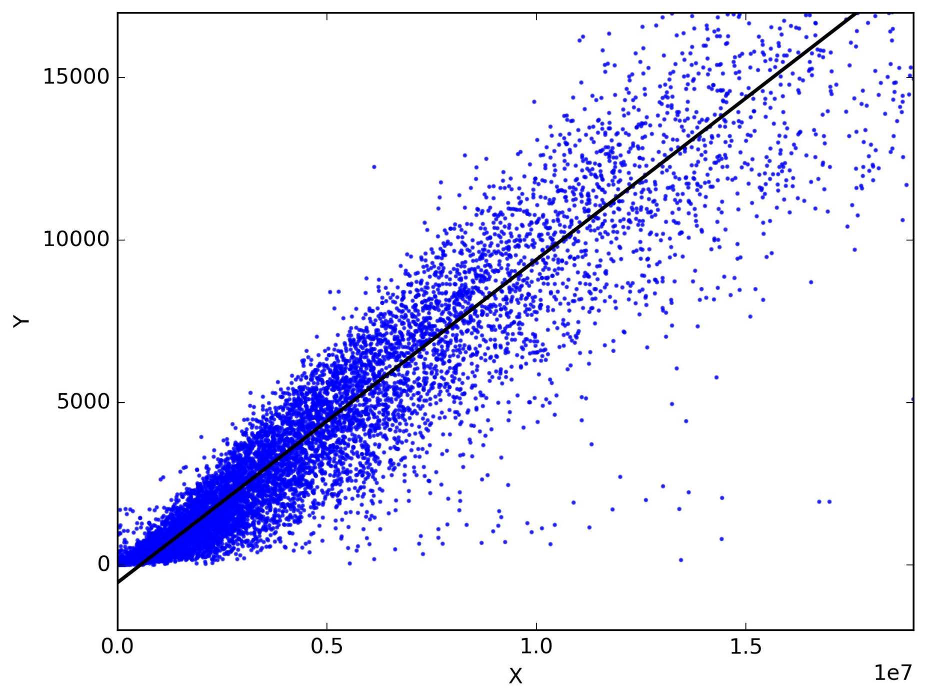
Regression :

- input maps directly to a continuous output space
- involves estimating or predicting a response

Classification:

- input maps to a class label
- identifying class membership

What is linear regression?



Linear regression

General form of the linear regression problem:

$$y = b + a * x + \epsilon$$

Where

$$\epsilon \sim N(0, \sigma^2)$$

In vector form:

$$y = b + a * x + \epsilon = \theta^T X + \epsilon$$

Probabilistic form:

$$p(y|x, \theta) = N(y|w^T x, \sigma^2)$$

Linear regression

We want to solve the optimization problem:

$$\hat{\theta} = \arg\max_{\theta} log(p(D|\theta))$$

Where

$$l(\theta) = log(p(D|\theta)) = \sum_{i=1}^{N} log(p(y_i|x_i, \theta))$$

Negative log likelihood

$$NLL(\theta) = -\sum_{i=1}^{N} log(p(y_i|x_i, \theta))$$

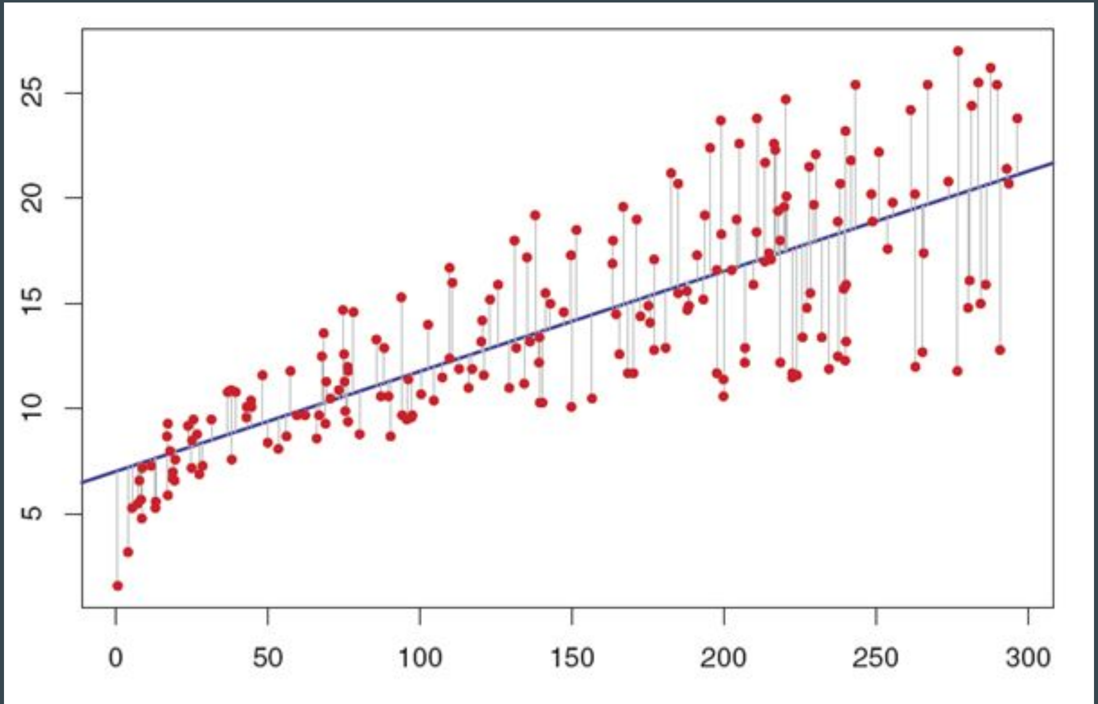
The optimization

$$l(\theta) = \sum_{i=1}^{N} log[(\frac{1}{2\pi\sigma^2})^{1/2} exp(\frac{-1}{2\sigma^2}(y_i - w^T x_i)^2)]$$
$$= \frac{-1}{2\sigma^2} RSS(w) - \frac{N}{2} log(2\pi\sigma^2)$$

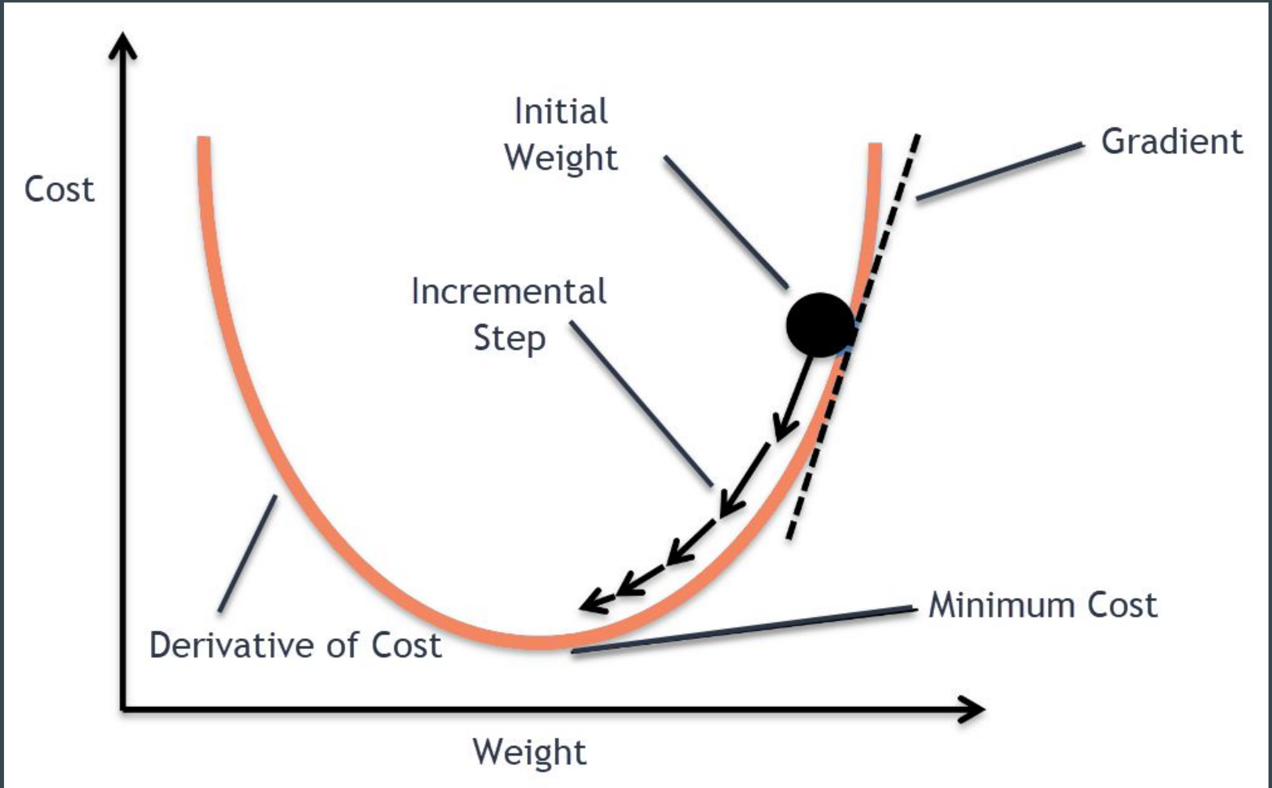
and

$$RSS(w) = \sum_{i=1}^{N} (y_i - w^T x_i)^2$$

Distances:



Gradient descent



The optimization

$$l(\theta) = \sum_{i=1}^{N} log[(\frac{1}{2\pi\sigma^2})^{1/2} exp(\frac{-1}{2\sigma^2}(y_i - w^T x_i)^2)]$$

$$= \frac{-1}{2\sigma^2} RSS(w) - \frac{N}{2} log(2\pi\sigma^2)$$

and

$$RSS(w) = \sum_{i=1}^{N} (y_i - w^T x_i)^2$$

Gradient descent

Derivative:

$$g(w) = \sum_{i=1}^{N} x_i(w^T x_i - y_i)$$

Given initial vector w_0

do for k=1, 2, ...

$$w_{k+1} = w_k - \alpha_k * g(w_k)$$

stop if :

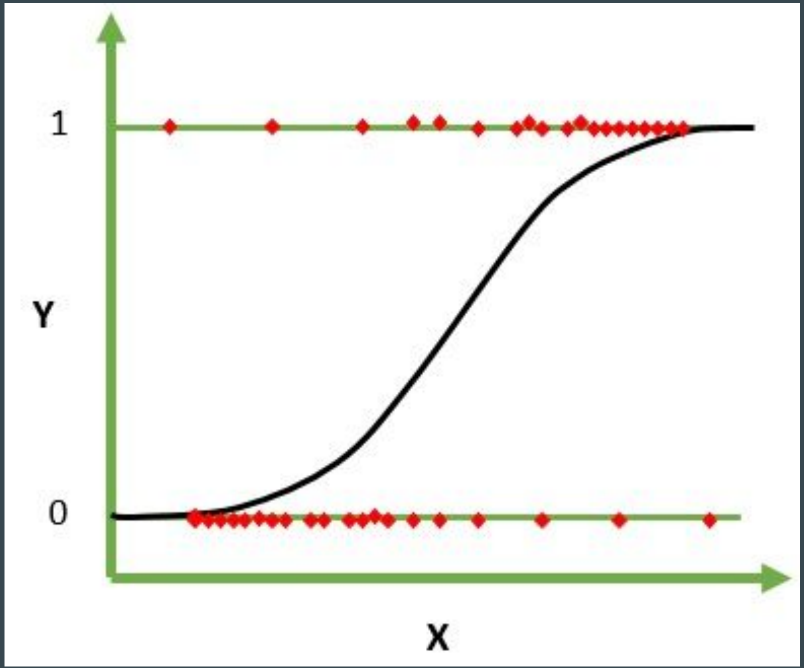
$$|w_{k+1} - w_k| < \epsilon$$

Summary

- Fit a linear hyperplane
- Calculate errors
- Iteratively minimize errors by gradient descent

Logistic regression

- linear model + threshold function
- sigmoid f(x) is in [0, 1]
- take 0.5 as threshold
- f(x) > 0.5 -> class 1
- f(x) <= 0.5 -> class 2



Logistic regression derivation

Logistic regression corresponds to this binary classification model:

$$p(y|x, w) = Ber(y|sigm(w^T x)$$

Where the negative log likelihood is:

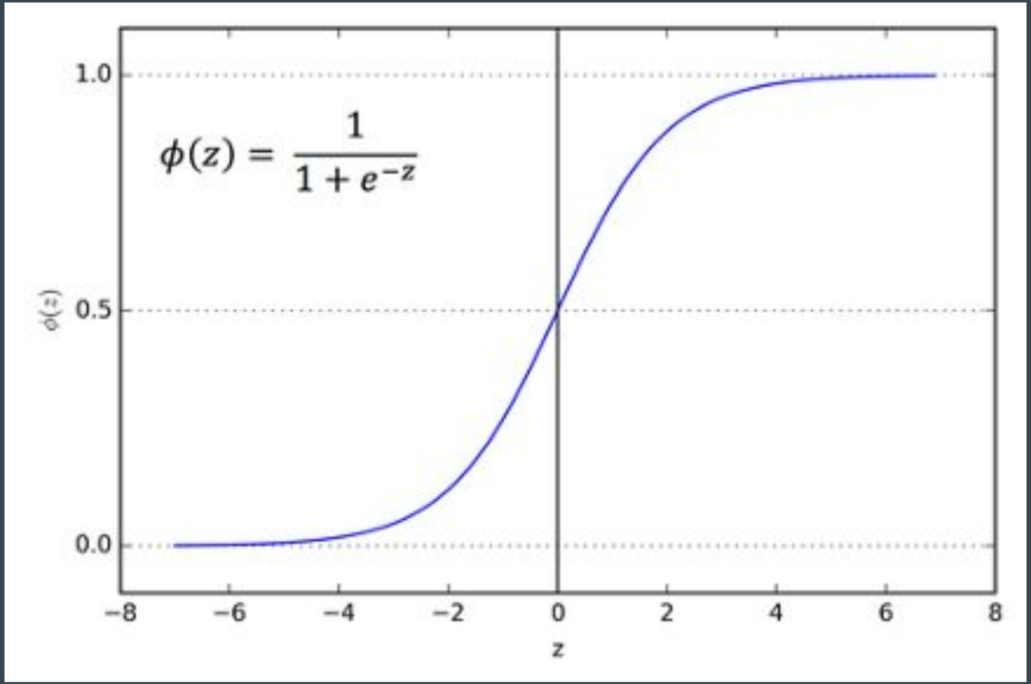
$$NLL(w) = -\sum_{i=1}^{N} log[\mu_i^{I(y_i=1)} * (1-\mu_i)^{I(y_i=0)}]$$

$$= -\sum_{i=1}^{N} [y_i log\ \mu_i + (1-y_i) log(1-\mu_i)]$$

Where

$$\mu_i = sigm(w^T x_i)$$

Sigmoid function



Logistic regression derivation

Logistic regression corresponds to this binary classification model:

$$p(y|x, w) = Ber(y|sigm(w^T x)$$

Where the negative log likelihood is:

$$NLL(w) = -\sum_{i=1}^{N} log[\mu_i^{I(y_i=1)} * (1 - \mu_i)^{I(y_i=0)}]$$

$$= -\sum_{i=1}^{N} [y_i log\ \mu_i + (1 - y_i) log(1 - \mu_i)]$$

Where

$$\mu_i = sigm(w^T x_i)$$

Gradient descent for logistic regression

$$g(w) = \frac{\partial(Err(w))}{\partial w} = -\sum_{i=1}^{N} x_i(y_i - \sigma(w^T x_i))$$

Given initial vector w_0

do for k=1, 2, ...

$$w_{k+1} = w_k - \alpha_k * g(w_k)$$

stop if :

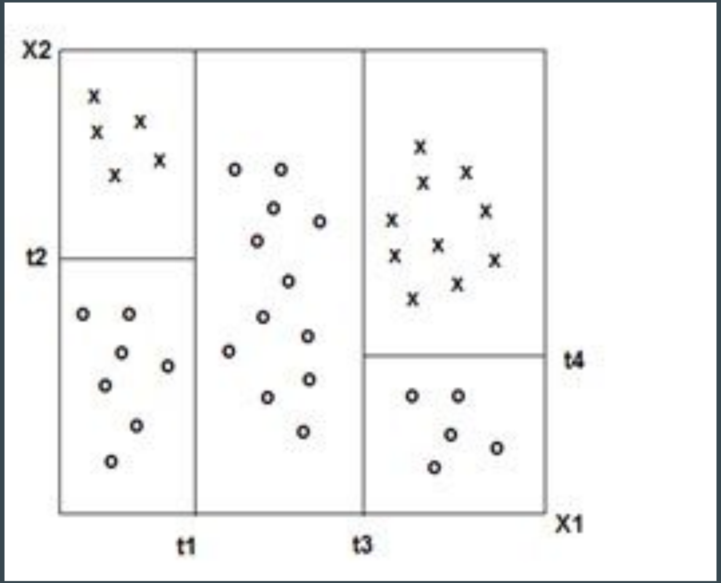
$$|w_{k+1} - w_k| < \epsilon$$

Notes

- There are other optimization algorithms that might be better
- Minimum of errors != accuracy
- Classification accuracy: misclassified/all

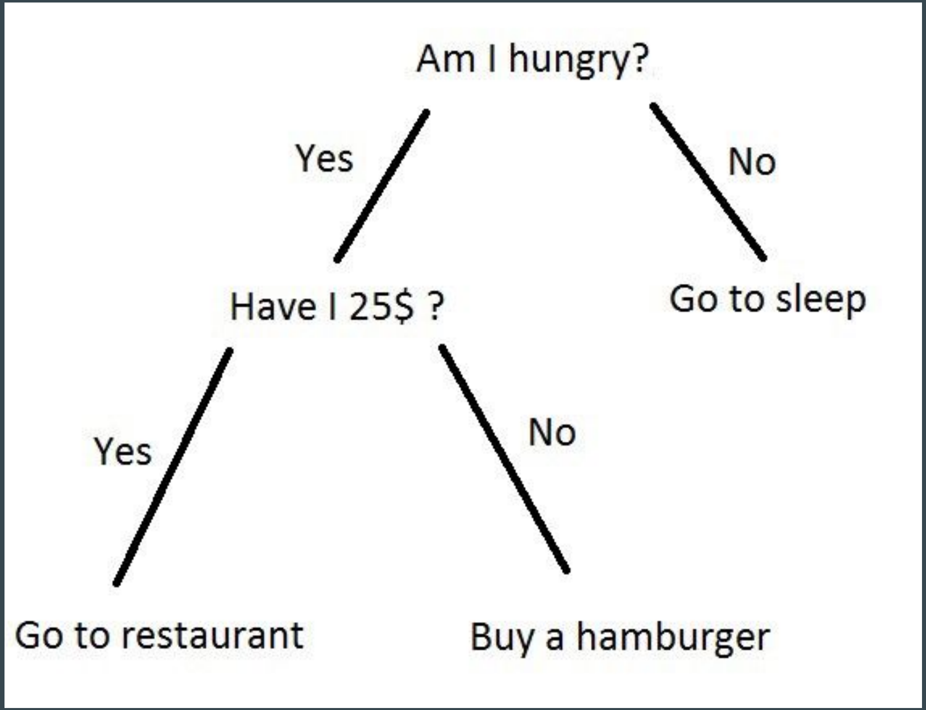
CART - Classification and regression trees

What if we want a richer partitioning?



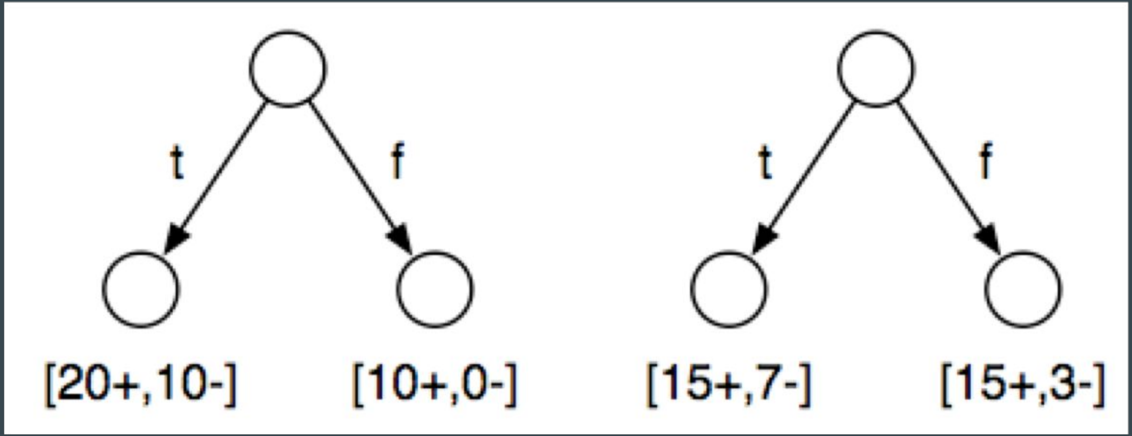
Decision trees

- Nodes partition the data
- Internal node: test on values
- Leaf node: training example that satisfy each branch



How to grow a tree

- At each node, we want to maximise the information gain
- Example - 40 examples: 30 positive, 10 negative
- Which test is better? One with more information



Information

Dataset entropy (information): -p*log2(p) - q*log2(q)

- p = positive/positive+negative
- q = negative/positive+negative

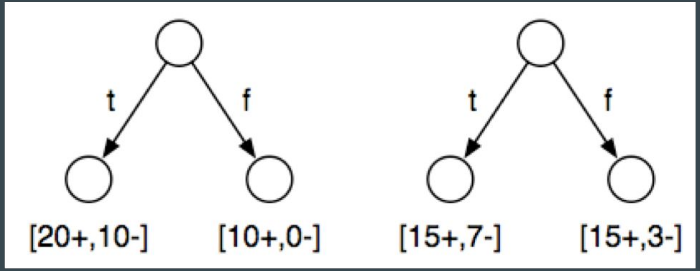
Conditional entropy: probability of arriving to a node * entropy

Which is best?

Dataset entrop:

30+ and 10- => p=3/4 , q = 1/4

H(D) = -(3/4)log2 (3/4)-(1/4)log2 (1/4) = 0.811



Test 1:

H(D|T1) = (30/40)[-(20/30)log2 (20/30)-(10/30)log2 (10/30)]+(10/40)[0] = 0.688

Test 2:

H(D|T2) = (22/40)[-(15/22)log2 (15/22)-(7/22)log2 (7/22)]

+ (18/40)[-(15/18)log2 (15/18)-(3/18)log2 (3/18)] = 0.788

Growing a tree

1. Pick best test to split the data
2. Split the data
3. Repeat for each subset

Stop if:

4. If all the training instances have the same class

Summary

Linear regression

Gradient descent

Logistic regression = linear regression + sigmoid

Decision trees

Information gain