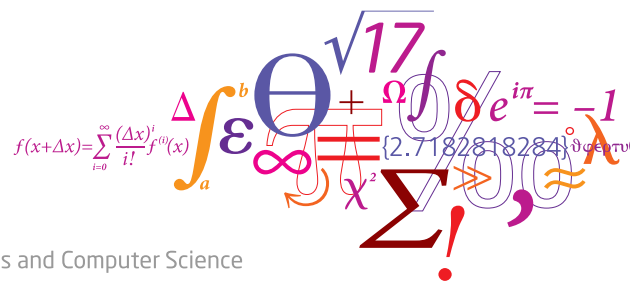
02450: Introduction to Machine Learning and Data Mining

Decision trees and linear regression

Georgios Arvanitidis

DTU Compute, Technical University of Denmark (DTU)



DTU Compute
Department of Applied Mathematics and Computer Science

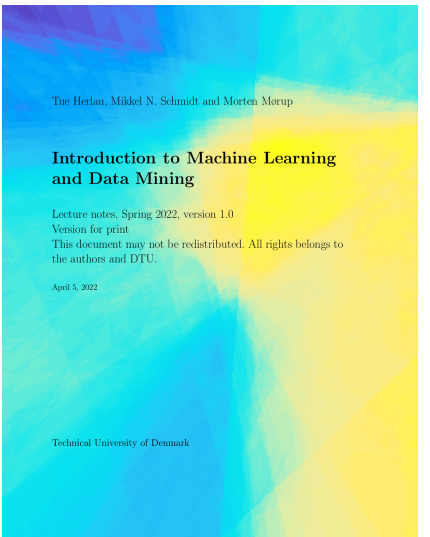
Today

Feedback Groups of the day:

RÈmi Graffin, Joachim Nordskov Grahndin, Paula Johanna Maria Granlund, Jonas Ravn Grelle, Mejse Gr¯nborg-Koch, Mikkel Romvig Gr¯ngÂrd, Niels Christian Gr¯nlykke, Peter-Aleksandr Gr¸nder, Nicholas Oliver Grundtvig, Toma Guci, Signe Friis Guldberg, Niklas Spanggaard Gundersen, Emil Johannes Gantzel Gungaard, Liming Guo, Benjamin Mirad Gurini, Jaime Guzman, Zolt · n Gyøri, Johann Haack, Eske Haack, Zeyneb Hady, Mitta Hage, Alexander Jeppsson Hagedorn, Andreas Hallb‰ck, David Elias Hammersh¯i, Ethan Hampton, Andreas Bacher Hannah, Marcus Jagd Hansen, Karoline Klan Hansen, Valdemar Hee Hansen, Christian Ong¯ Hansen, Morten Jao Hansen, Marie Vennerstr¯m Hansen, Johanne Cornelius Hansen, Stefan Smedegaard Hansen, Betina Mee Hansen, Jeppe Hansson, Johanna Munch HaraldsdÛttir, Shania Hau, Gustav Haugegaard, Patrick Guldager Heen, Thomas Seifert Hegelund, Sara Heiberg, Lasse Johs Lindenhoff Heide, Franz Heim, David Heiser, Mats Hellwig, Anton Aaby Henriksen, Benjamin Jenfort Henriksen, Moritz Samuel Herrmann, Nikolaj Severin StÊhr Hertz, David Hindahl, Benjamin Ho, Maria Hoffmann, Pierre H¯genhaug, Philip Kishimoto Hohwy

Reading material:

Chapter 8, Chapter 9



Lecture Schedule

1. Introduction
   30 August: C1

Data: Feature extraction, and visualization

2. Data, feature extraction and PCA
   6 September: C2, C3
3. Measures of similarity, summary statistics and probabilities
   13 September: C4, C5
4. Probability densities and data visualization
   20 September: C6, C7

Supervised learning: Classification and regression

5. **Decision trees and linear regression**
   27 September: C8, C9
6. Overfitting, cross-validation and Nearest Neighbor
   4 October: C10, C12 **(Project 1 due before 13:00)**
7. Performance evaluation, Bayes, and Naive Bayes
   11 October: C11, C13
8. Artificial Neural Networks and Bias/Variance
   25 October: C14, C15
9. AUC and ensemble methods
   1 November: C16, C17

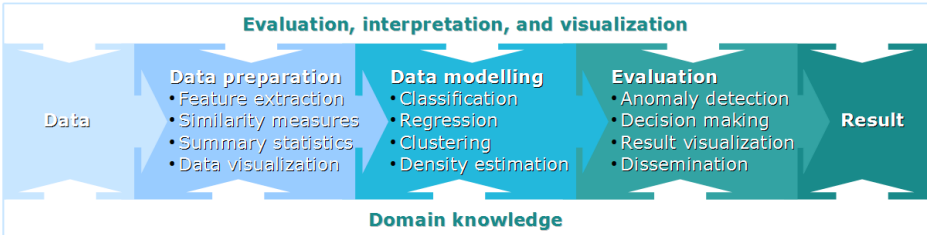
Unsupervised learning: Clustering and density estimation

10. K-means and hierarchical clustering
    8 November: C18
11. Mixture models and density estimation
    15 November: C19, C20 **(Project 2 due before 13:00)**
12. Association mining
    22 November: C21

Recap

13. Recap and discussion of the exam
    29 November: C1-C21

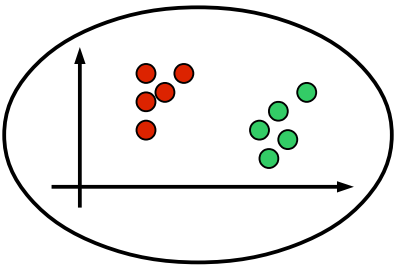
Online help: Piazza
Videos of lectures: `https://video.dtu.dk`
Streaming of lectures: Zoom (link on DTU Learn)



Learning Objectives

- Explain what supervised learning is
- Explain the difference between classification and regression
- Be able to evaluate classifiers in terms of the confusion matrix, error rate and accuracy
- Understand the principle behind decision trees and Hunt's algorithm
- Apply and interpret decision trees, linear regression and logistic regression

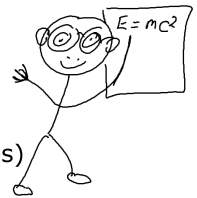
Supervised and Unsupervised learning

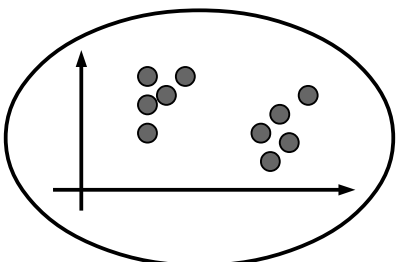


Supervised Learning

Input data $\mathbf{x}_n$ and output y_n

(Generalize from known examples)



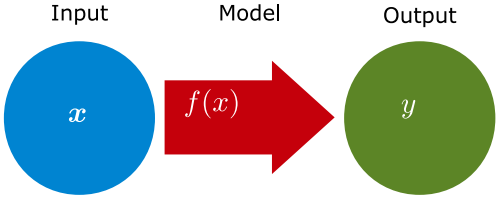


Unsupervised Learning

Input data $\mathbf{x}_n$ alone

(Exploratory analysis)

Supervised learning



- **Data**
  - Inputs and outputs *(this is what we are given)*

$$\{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$$

- **Model**
  - Function that maps inputs to outputs *(what we are trying to determine)*

$$f(\boldsymbol{x})$$

- **Cost function**
  - Dissimilarity measure between observation and prediction *(how we tell if a model is good or bad)*

$$d\big(y, f(\boldsymbol{x})\big)$$

- **Types of supervised learning**
  - Regression: Continuous output **y**
  - Classification: Discrete output **y**

Classification

- **Definition:** Learning a function that maps a data object to a discrete class

- **Why classify?**
  - Descriptive modeling
    - Explain / understand the relation between attributes and class
  - Predictive modeling
    - Predict the class of a new data object

Decision trees

- Remember the game "20 questions to the professor"? (see also www.20q.net)

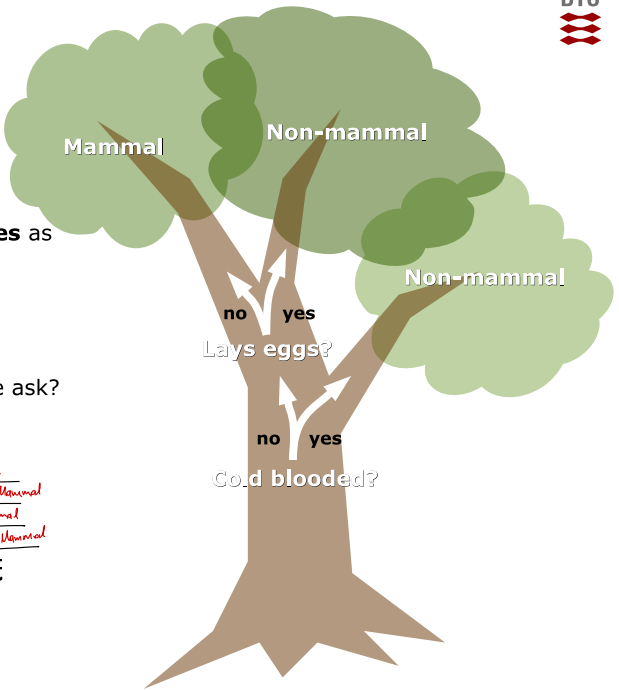
Q1.	Is it an Animal? Yes.
Q2.	Can you hold it? No.
Q3.	Does it live in groups (gregarious)? Yes.
Q4.	Are there many different sorts of it? No.
Q5.	Can it jump? Yes.
Q6.	Does it eat seeds? No.
Q7.	Is it white? Sometimes.
Q8.	Is it black and white? No.
Q9.	Does it have paws? Yes.
Q10.	Can you see it in a zoo? Yes.
Q11.	Does it roar? Yes.
Q12.	Is it worth a lot of money? Yes.
Q13.	Does it have spots? Yes.
Q14.	Is it multicoloured? Yes.
Q15.	Can you make money by selling it? Yes.
Q16.	Does it live in the jungle? Yes.
Q17.	I guessed that it was a leopard? Wrong.
Q18.	Does it like to play? Yes.
Q19.	I guessed that it was a cheetah? Wrong.
Q20.	I am guessing that it is a siberian tiger? Correct.

Decision trees

- Ask a series of questions until a conclusion is reached
- **Example**: Classify **vertebrates** as
  - **Mammal** or
  - **Non-mammal**
- **Learning task**
  - Which questions should we ask?

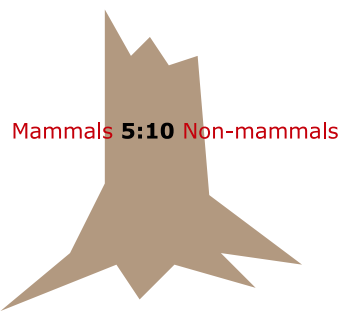
Example of data matrix

	Cold blood	Lay eggs	Has fur/hair	Label
data point 1	Yes	No	No	Non-Mammal
data point 2	No	No	Yes	Mammal
data point 3	Yes	Yes	No	Non-Mammal
⋮	⋮	⋮	⋮	⋮



Hunts algorithm

- Assign all data objects to the root

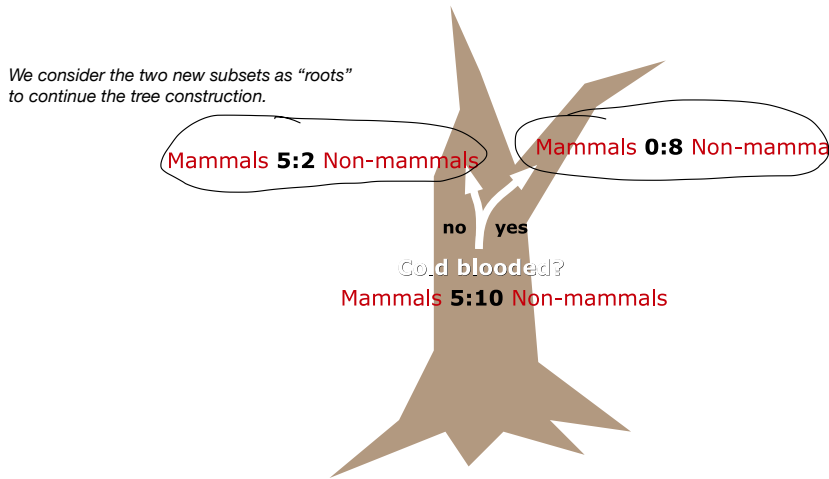


Hunts algorithm

- Select an attribute test condition
  - Find a good question to ask

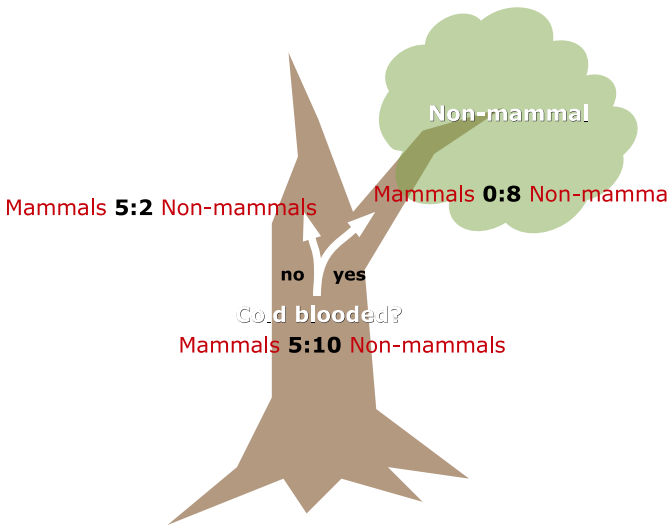
Hunt's Algorithm

- Partition the data objects into subsets according to the test condition



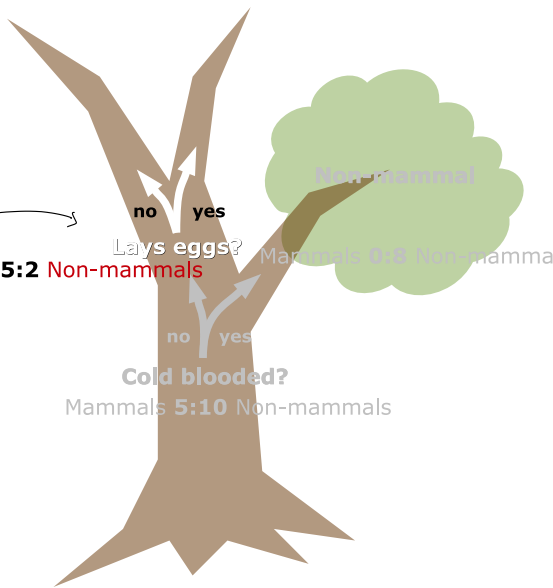
Hunts algorithm

- If all data objects belong to the same class
  - Create a leaf node



Hunts algorithm

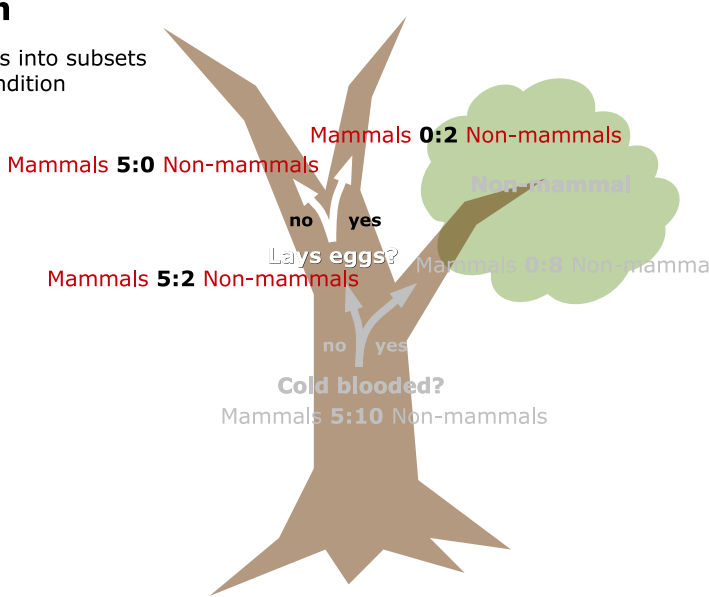
- Repeat for each non-leave node:
  - Select an attribute test condition



We consider this subset as a "root" and we continue the tree construction i.e. we try to find a good question to participate the data into two subsets.

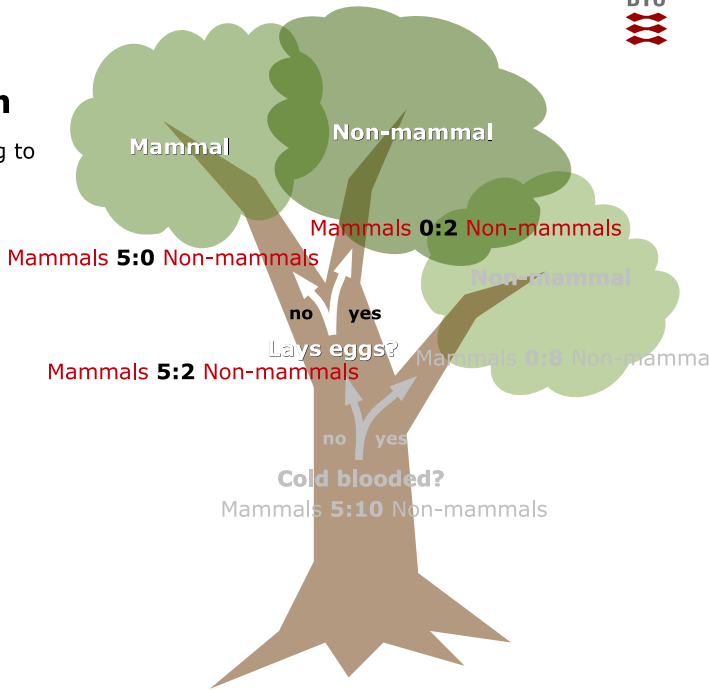
Hunts algorithm

- Partition the data objects into subsets according to the test condition



Hunts algorithm

- If all data objects belong to the same class
  - Create a leaf node

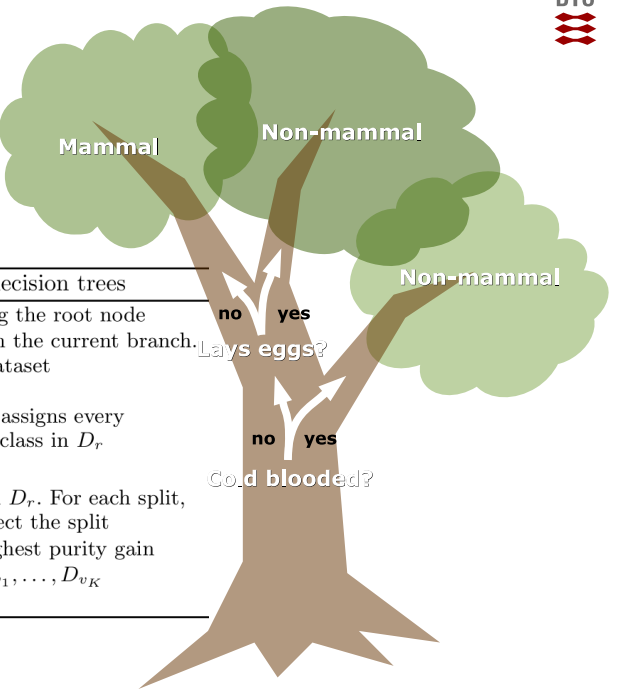


Hunts algorithm

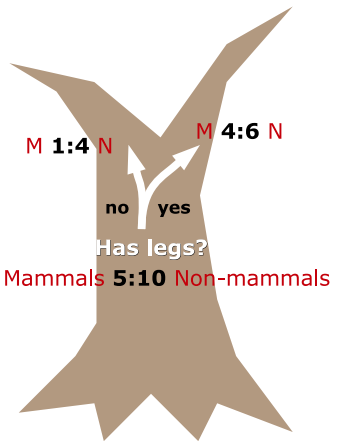
- But how do we find the **best question** at each step?

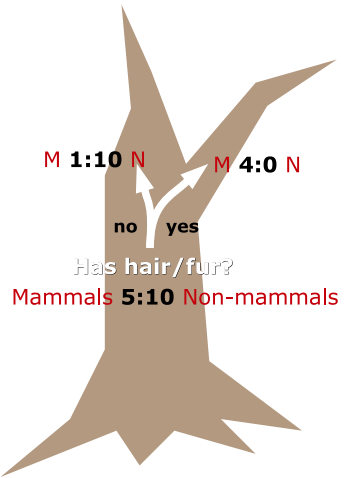
Algorithm 2: Hunt's algorithm for decision trees

Require: Initial tree T only containing the root node
Require: D_r : Dataset associated with the current branch. Initially just the full dataset
if The **stop criterion** is met **then**
  Add a leaf node to the tree which assigns every observation to the most prevalent class in D_r
else
  Try a number of different splits on D_r. For each split, compute the **purity gain** and select the split $D_r = \{D_{v_1}, \ldots, D_{v_K}\}$ with the highest purity gain
  Recursively call the method on $D_{v_1}, \ldots, D_{v_K}$
end if



Which split is best?

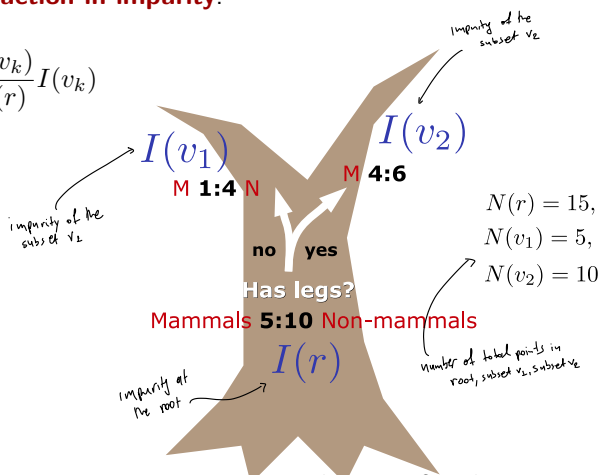




Which split is best?

- Create a measure Δ (**the purity gain**) of how good a split is
- A binary split creates 3 partitions: the root r and the right/left branches v_1, v_2.
- For each partition, we compute $I(r), I(v_1), I(v_2)$ (the **impurity**)
- Purity gain is the **weighted reduction in impurity**:

$$\Delta = I(r) - \sum_{k=1}^{K=2} \frac{N(v_k)}{N(r)} I(v_k)$$



$p(c|v)$: the prob. of class c for $v = r, v_1, v_2$

Which split is best?

- Create a measure Δ (**the purity gain**) of how good a split is
- A binary split creates 3 partitions: the root r and the right/left branches v_1, v_2.
- For each partition, we compute $I(r), I(v_1), I(v_2)$ (the **impurity**) of each partition
- Purity gain is the **weighted reduction in impurity**:

$$\Delta = I(r) - \sum_{k=1}^{K=2} \frac{N(v_k)}{N(r)} I(v_k)$$

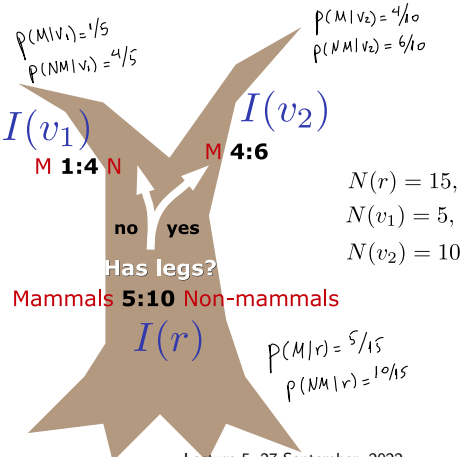
The impurity measure $I()$ can be one of the following

$$\text{Entropy}(v) = -\sum_{c=1}^{C} p(c|v) \log_2 p(c|v),$$

$$\text{Gini}(v) = 1 - \sum_{c=1}^{C} p(c|v)^2,$$

$$\text{ClassError}(v) = 1 - \max_c p(c|v).$$

$$p(c|v) = \frac{\{\text{Nr. in class } c \text{ in branch } v\}}{N(v)}$$



Quiz 1, Impurity gain

If we use the Gini index as impurity measure I, what is the purity gain Δ for the split indicated by the tree?

$$\Delta = I(r) - \sum_{k=1}^{K=2} \frac{N(v_k)}{N(r)} I(v_k)$$

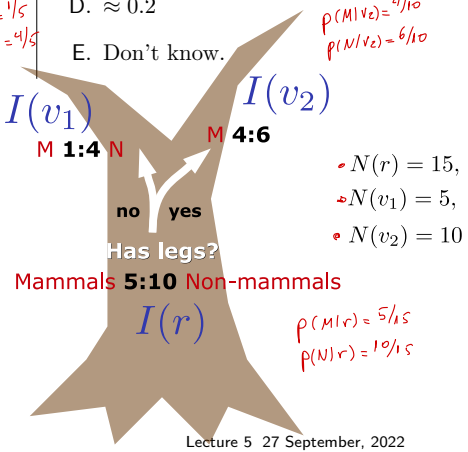
A. ≈ 0.0177

B. ≈ 0.104

C. ≈ 0.129

D. ≈ 0.2

E. Don't know.



The impurity measure $I()$ can be one of the following

$$\text{Entropy}(v) = -\sum_{c=1}^{C} p(c|v) \log_2 p(c|v),$$

$$\text{Gini}(v) = 1 - \sum_{c=1}^{C} p(c|v)^2,$$

$$\text{ClassError}(v) = 1 - \max_c p(c|v).$$

$$p(c|v) = \frac{\{\text{Nr. in class } c \text{ in branch } v\}}{N(v)}$$

Using Gini impurity we get:

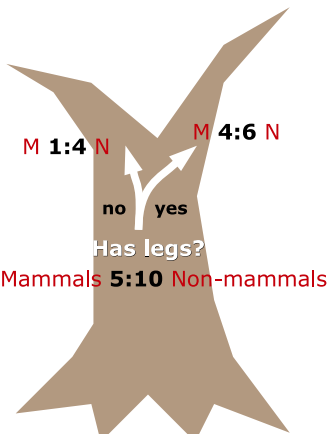
$$I(r) = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2, I(v_1) = 1 - \left(\frac{1}{5}\right)^2 - \left(\frac{4}{5}\right)^2, I(v_2) = 1 - \left(\frac{4}{10}\right)^2 - \left(\frac{6}{10}\right)^2,$$

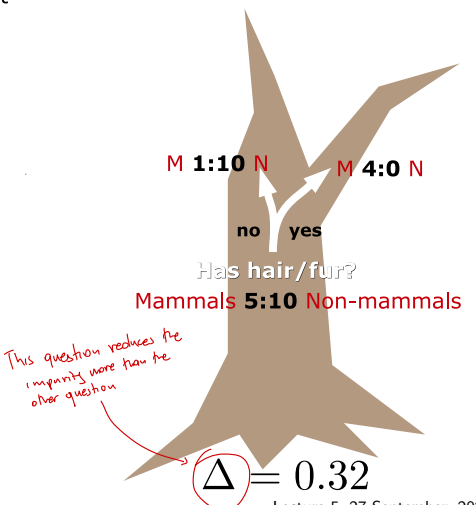
and finally

$$\Delta = I(r) - \frac{5}{15} I(v_1) - \frac{10}{15} I(v_2) \approx 0.0177$$

Selecting the best split

- Consider a large number of possible splits
- Compute a measure of impurity after the proposed split
  - For each new branch of the tree
  - Compute weighted average impurity
- Choose split that reduces impurity most

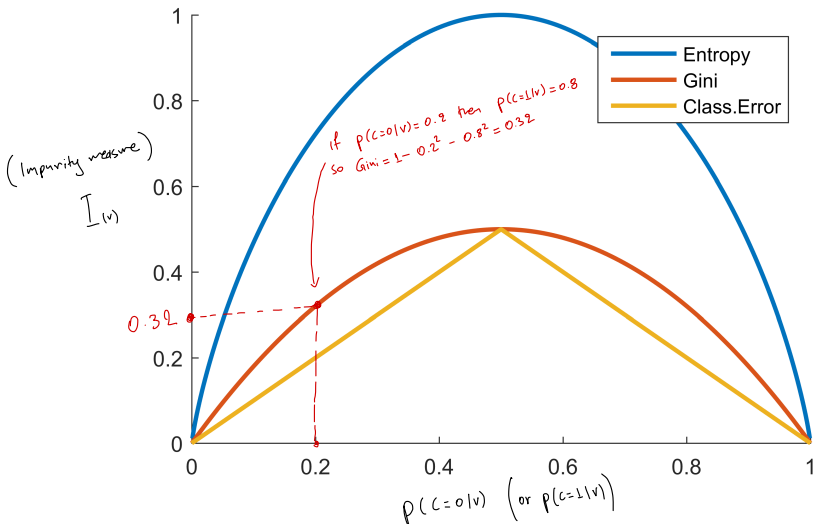




$$\Delta = I(r) - \sum_{k=1}^{2} \frac{N(v_k)}{N(r)} I(v_k)$$

Example: Gini, $I(r) = 1 - p(c=0|v)^2 - p(c=1|v)^2$

For a two class problem



Which splits to consider

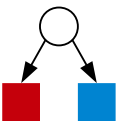
- **Binary**

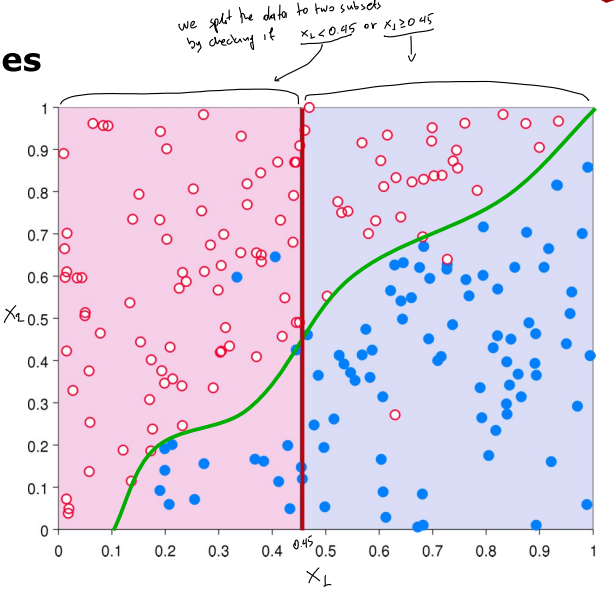
- **Nominal**

- **Ordinal**

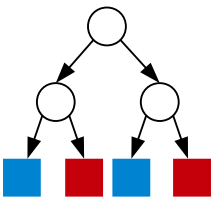
- **Continuous**

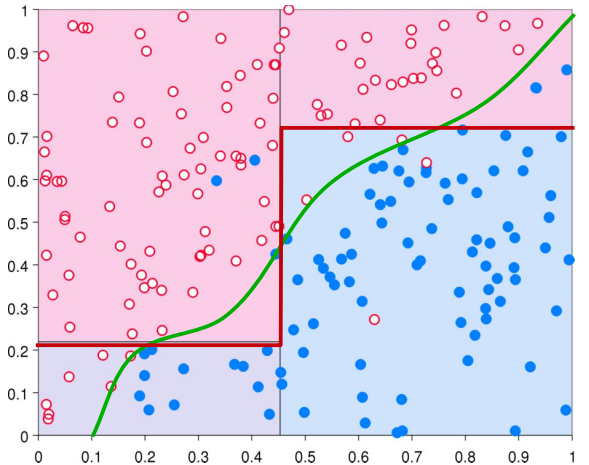
Classification Trees





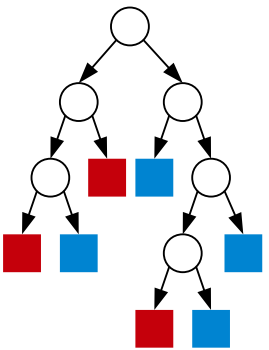
Classification trees

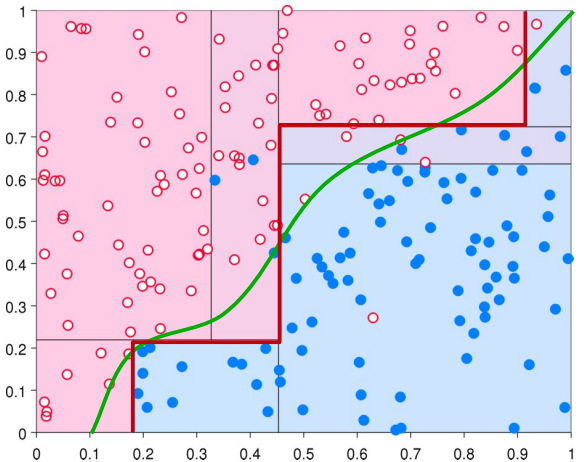




We give a label to a leaf based on the majority of the points in it. For example: if we have 10 blue points and 2 red points in a leaf, we will assign the label blue to this leaf.

Classification trees





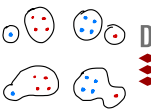
Classification trees

Common stopping criteria:

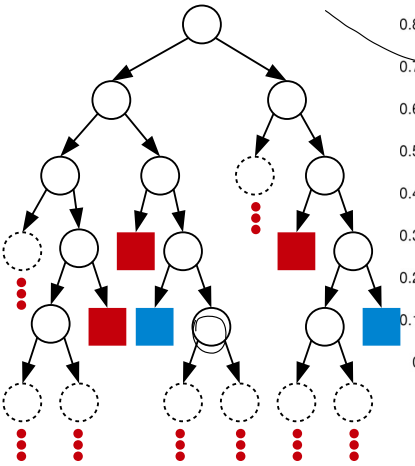
1. All records have the same class label
2. The number of observations have fallen below some minimum treshold

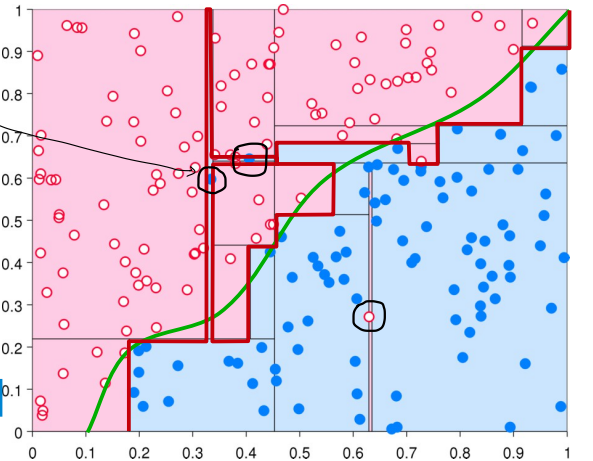
Example:
These are the final leaves when using criterion (1).

We can change the final leaves, by using criterion (2) and we stop splitting when we have at least 5 points in a leaf.



Probably these are noisy points, so we should prevent our model from classifying them and consequently the regions around them "correctly". In this example, it seems to be an unnatural behaviour to have these small regions classified with another label. This was the result of using the (1) as the stopping criterion when constructing the tree.





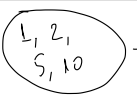
If we want all the final leaves to be pure (have points only from one class), we might end up to this "overfitted" solution, which does not generalise well.

Regression trees

Algorithm 4: Hunt's algorithm for regression trees

Require: Initial tree T only containing the root node
Require: D_r : Dataset associated with the current branch. Initially just the full dataset
if The **stop criterion** is met **then**
Add a leaf node to the tree which assigns every observation the mean value of the nodes in D_r: $y(r) = \frac{1}{N(r)} \sum_{i \in r} y_i$
else
Try a number of different splits on D_r. For each split, compute the **purity gain** using the sum-of-squares impurity measure and select the split $D_r = \{D_{v_1}, \ldots, D_{v_K}\}$ with the highest purity gain
Recursively call the method on $D_{v_1}, \ldots, D_{v_K}$
end if

Example: the y_i of the points x_i in this leaf



→ 4.5 average value in this leaf.

Use mean square error as purity gain

$$I(v) = \frac{1}{N(v)} \sum_{i \in v} (y_i - \hat{y}_v)^2, \quad \hat{y}_v = \frac{1}{N(v)} \sum_{i \in v} y_i$$

For the impurity of a leaf, we compute the error between the average value of the leaf and the y_i of the points x_i in it.

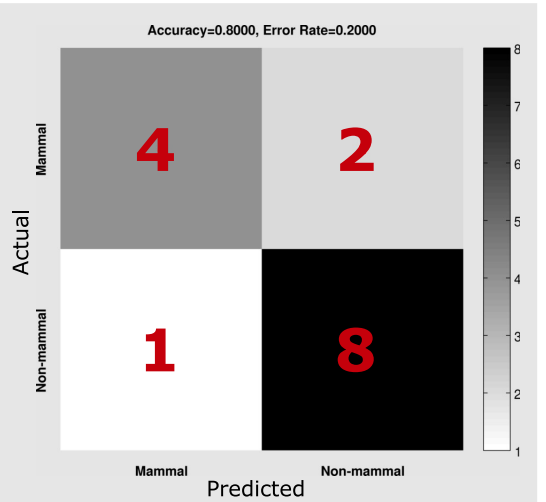
When we construct a regression tree the value of a leaf is the average value of the points in this leaf.

Confusion matrix

- Visualization of actual versus predicted class labels

- **Accuracy**
  (Number of correctly predicted observations divided by the total number of observations)
  $$\frac{4+8}{4+2+1+8} = 80\%$$

- **Error rate**
  (Number of in-correctly predicted observations divided by the total number of observations)
  $$\frac{2+1}{4+2+1+8} = 20\%$$



Evaluating a regression model

Compute average loss per observation:

$$E = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(\boldsymbol{x}_i))$$

Where we either use L_1 or L_2 (Euclidean) loss

$$L_1(y_i, \hat{y}_i) = |y_i - \hat{y}_i|, \quad L_2(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$$

(Compare these to p-norms)

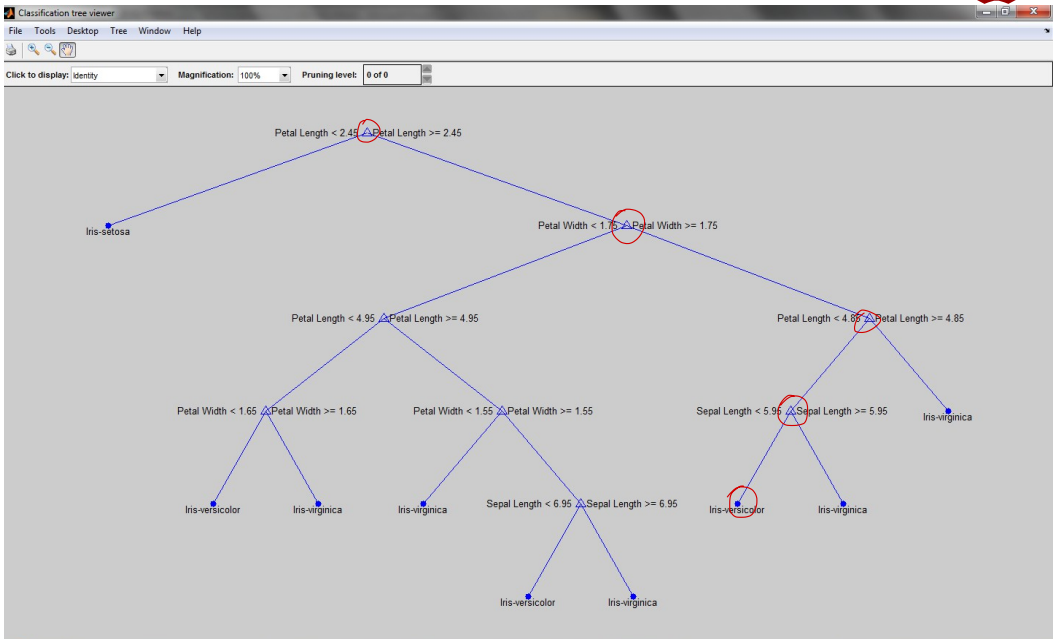
The iris data set

- **Three flowers**
  - 50 instances of each class, 150 in total

- **Attributes**
  - Sepal (outermost leaves)
    - length in cm
    - width in cm
  - Petal (innermost leaves)
    - length in cm
    - width in cm
  - Class of flower
    - Iris Setosa
    - Iris Versicolour
    - Iris Virginica

Fower ID	Attribute			
	Sepal Length	Sepal Width	Petal Length	Petal Width
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
.	.	.	.	.
.	.	.	.	.
150	5.9	3.0	5.1	1.8

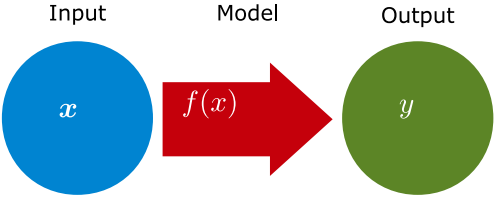
X Observation x Attribute



What would the following iris flower be classsified as?

Sepal Length	Sepal Width	Petal Length	Petal Width
4.0	3.5	3.0	2.0

Supervised learning



- **Mapping between domains**
  - Classification: Discrete (nominal) output
  - Regression: Continuous output

Supervised learning

- **Data**
  - Inputs and outputs $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$
- **Model**
  - Function that maps inputs to outputs
    $$f(\boldsymbol{x})$$
- **Cost function**
  - Dissimilarity measure between data and model
    $$d\big(y, f(\boldsymbol{x})\big)$$

Regression

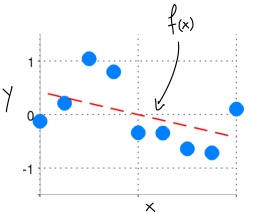
- **Definition:** Learning a function that maps a data object to a continuous-valued output

- **Why Regression?**
  - Descriptive modeling
    - Explain / understand the relation between attributes and continous-valued output
  - Predictive modeling
    - Predict the output value of a new data object

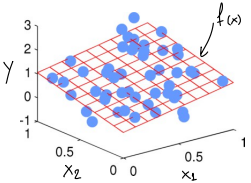
We want to learn the parameters $w_0, w_1, \ldots$ from the given dataset.

Linear regression

- 1-dimensional inputs
  $f(x) = w_0 + w_1 x$



- 2-dimensional inputs
  $f(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2$



- K-dimensional inputs
  $f(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_K x_K$

Linear regression

- K-dimensional inputs

$$f(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_K x_K = w_0 + \sum_{i=1}^{K} w_i \cdot x_i$$

w_i parameters
x_i attributes of the point $\times$ (vector)

- Non-linearly transformed inputs

$$f(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_K x^K$$

new features = non-linear transformations of the given attributes

$$f(x) = w_0 + w_1 sin(x) + w_2 cos(x)$$

Remark:
We call it linear regression because we have a linear combination of the attributes/features with the parameters. If we include non-linear transformations of the attributes, then the learned function is non-linear, but it is again a linear combination of the features with the parameters.

Example: Observed data $\{x_i, y_i\}_{i=1}^{N}$, $x_i \in \mathbb{R}$ (1-dim input)

We include additional features, which are non-linear transformations of the 1-dim input x.

The observed data.

	y	x	x^2	sin(x)
point 1	y_1	2	4	sin(4)
point 2	y_2	5	25	sin(5)
⋮	⋮	⋮	⋮	⋮

The regression model we want to learn:

$y = f(x) = w_0 + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot \sin(x)$

This is a function $f: \mathbb{R} \to \mathbb{R}$

Linear regression

- K-dimensional inputs

$$f(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_K x_K$$

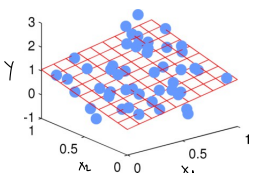
- Non-linearly transformed inputs

$$f(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_K x^K$$

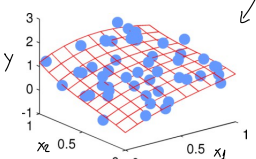
$$f(x) = w_0 + w_1 sin(x) + w_2 cos(x)$$

- **Example**

$$f(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2$$



We construct non-linear features based on the attributes x_1, x_2 of the given data points. The resulting nonlinear function is a linear combination of the non-linear features.



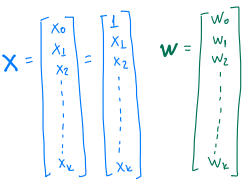
$$f(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + w_6 x_1^3 + w_7 x_1^2 x_2 + w_8 x_1 x_2^2 + w_9 x_2^3$$

Vector notation

- The linear model can be written compactly using vector notation

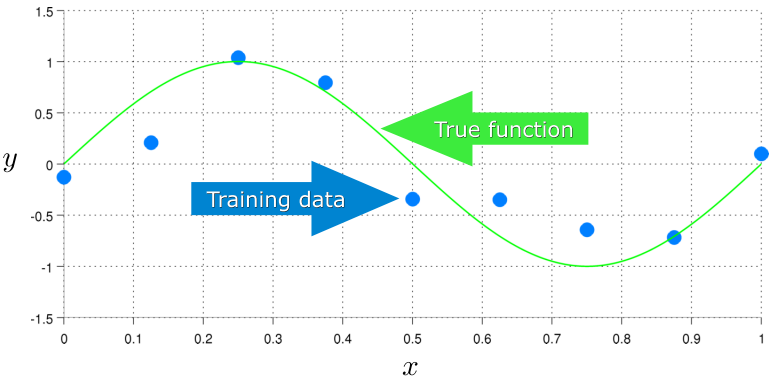
$$f(\boldsymbol{x}) = \overset{(w_0 \cdot 1 = w_0 \cdot x_0)}{w_0} + w_1 x_1 + w_2 x_2 + \cdots + w_K x_K$$

$$= \sum_{k=0}^{K} w_k x_k = \boldsymbol{x}^\top \boldsymbol{w}$$

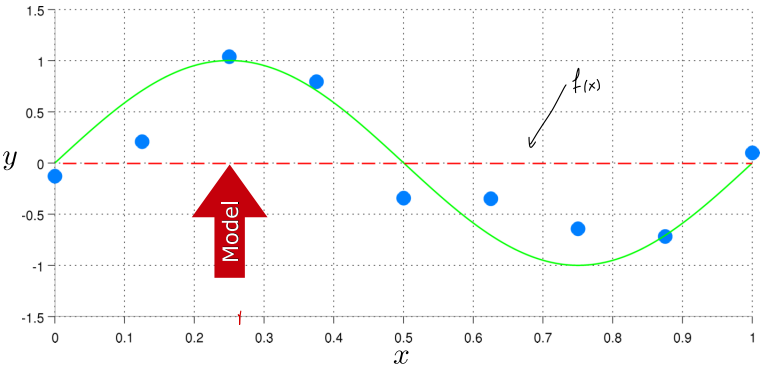


– where $x_0 = 1$

Linear regresssion



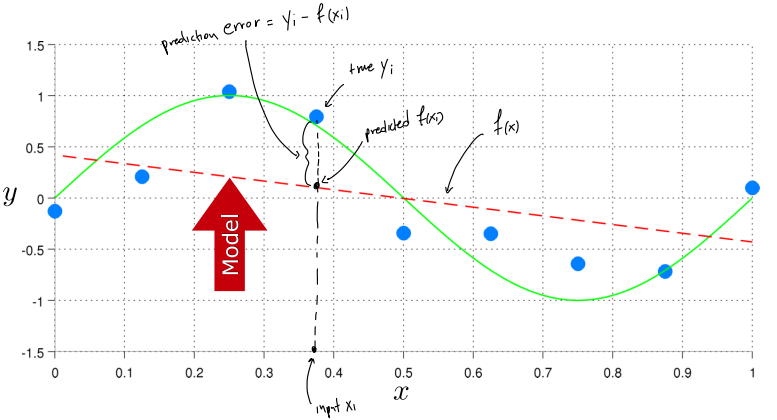
Linear regresssion



Model

$$f(x) = w_0$$

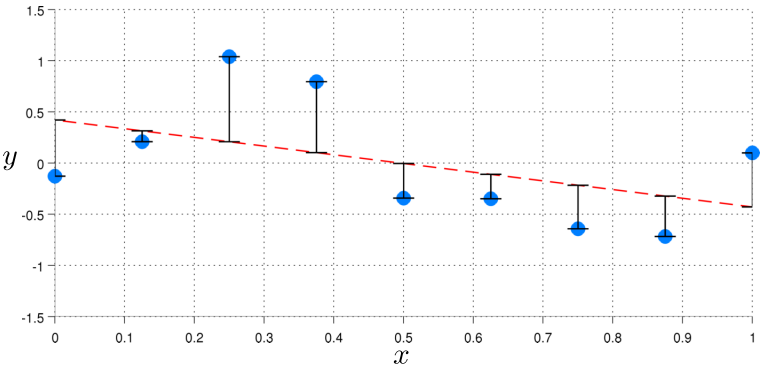
Linear regresssion



Model

$$f(x) = w_0 + w_1 x$$

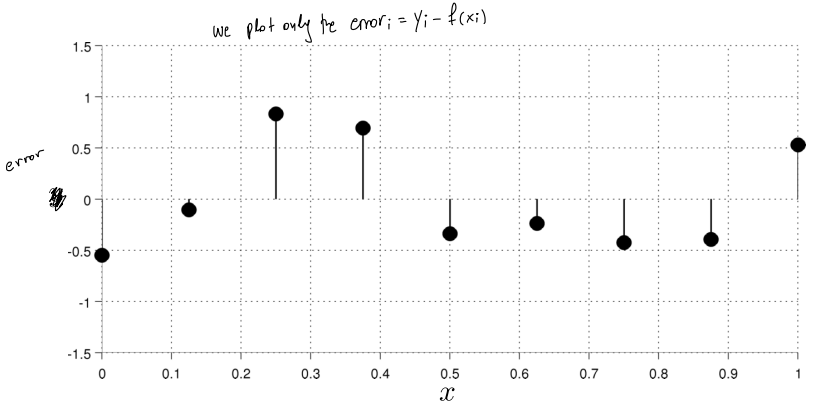
Residual error



Model

$$f(x) = w_0 + w_1 x$$

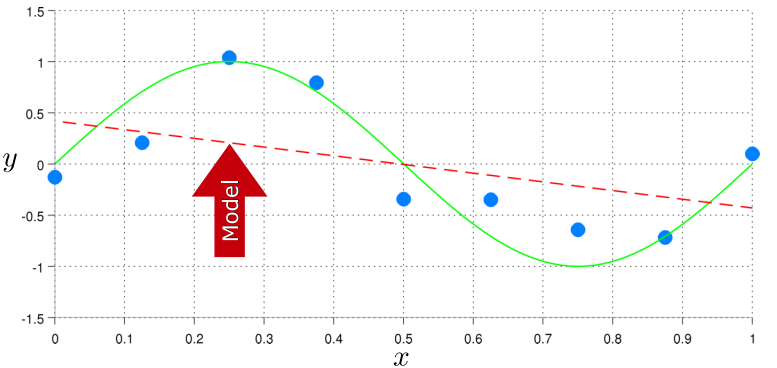
Residual error



Model

$$f(x) = w_0 + w_1 x$$

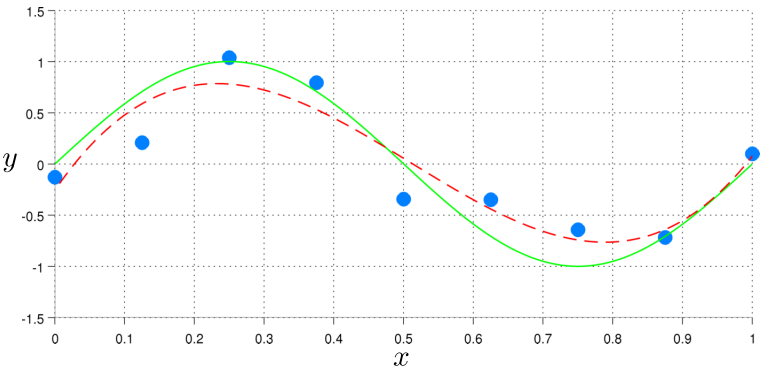
Linear regression



Model

$$f(x) = w_0 + w_1 x$$

Linear regression

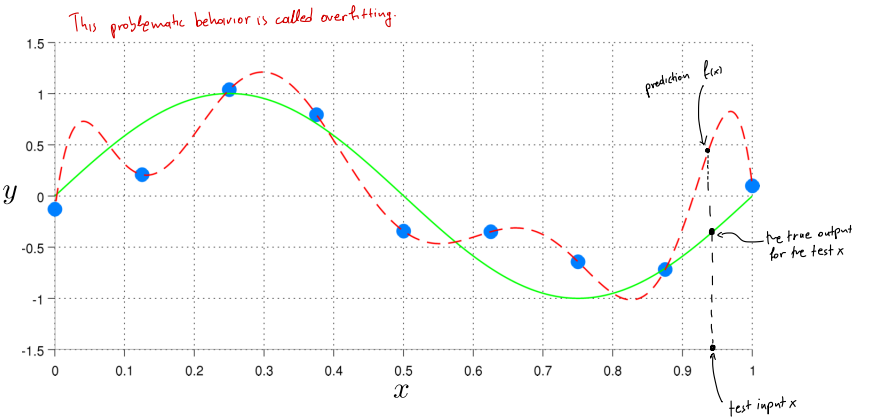


Model

$$f(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

(This function is a polynomial of degree 3. We combine linearly the parameters with the non-linear features.)

Linear regression



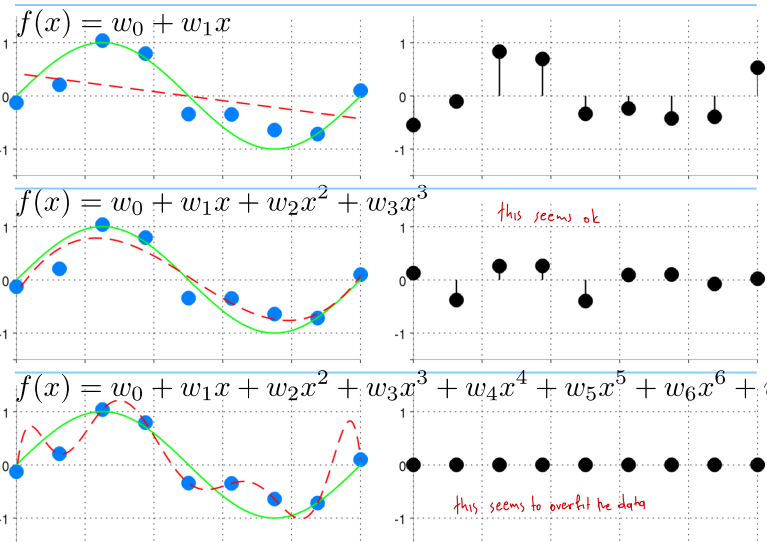
Model

$$f(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + w_5 x^5 + w_6 x^6 + w_7 x^7 + w_8 x^8$$

(This function is a polynomial of degree 8. This model has very high complexity which is unnecessary for the given data, and in fact, does not give good predictions for test (unseen) inputs.)

Model order

- Which model order
  - Gives the best fit?
  - Do you think is most "correct"?



Learning

Setup: We got some data $(\boldsymbol{y}, \boldsymbol{X})$. We have a way to define a function

$$f(\boldsymbol{x}; \boldsymbol{w}) = \tilde{\boldsymbol{x}}^T \boldsymbol{w} = w_0 + \sum_{k=1}^{M} x_k w_k$$

Learning

Setup: We got some data $(\boldsymbol{y}, \boldsymbol{X})$. We have a way to define a function

$$f(\boldsymbol{x}; \boldsymbol{w}) = \tilde{\boldsymbol{x}}^T \boldsymbol{w} = w_0 + \sum_{k=1}^{M} x_k w_k$$

- **Question:** How do we learn the correct $\boldsymbol{w}$?

Learning

Setup: We got some data $(\boldsymbol{y}, \boldsymbol{X})$. We have a way to define a function

$$f(\boldsymbol{x}; \boldsymbol{w}) = \tilde{\boldsymbol{x}}^T \boldsymbol{w} = w_0 + \sum_{k=1}^{M} x_k w_k$$

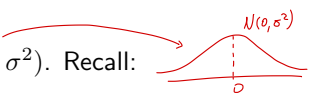
- **Question:** How do we learn the correct $\boldsymbol{w}$?
- Answer: For each observation, assume:

We assume that the functional relationship of the input to the output is $y_i = f(x_i; w)$ plus some random noise ϵ_i .

The noise might be due to the way we record the data e.g. our sensor is not super accurate, similar houses might have a small difference in price, etc.

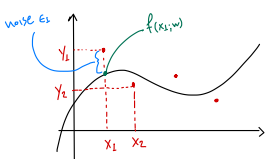
$$y_i = f(\boldsymbol{x}_i; \boldsymbol{w}) + \varepsilon_i$$

where ε_i is a normally distributed noise term $\mathcal{N}(\varepsilon_i | \mu = 0, \sigma^2)$. Recall:



$$\mathcal{N}(\varepsilon_i | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\varepsilon_i - \mu)^2}{2\sigma^2}}$$

Given data: $\{x_i, y_i\}_{i=1}^N$



$0 \Rightarrow \varepsilon_i = y_i - f(x_i; w)$

Learning

Setup: We got some data $(\boldsymbol{y}, \boldsymbol{X})$. We have a way to define a function

$$f(\boldsymbol{x}; \boldsymbol{w}) = \tilde{\boldsymbol{x}}^T \boldsymbol{w} = w_0 + \sum_{k=1}^{M} x_k w_k$$

- **Question:** How do we learn the correct $\boldsymbol{w}$?
- Answer: For each observation, assume:

$$y_i = f(\boldsymbol{x}_i; \boldsymbol{w}) + \varepsilon_i$$

where ε_i is a normally distributed noise term $\mathcal{N}(\varepsilon_i | \mu = 0, \sigma^2)$. Recall:

$$\mathcal{N}(\varepsilon_i | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\varepsilon_i - \mu)^2}{2\sigma^2}}$$

- This means that

$$p(y_i | \boldsymbol{x}_i, \boldsymbol{w}) =$$

Given an input x_i we want to estimate the distribution for the associated output y_i. This distribution depends on the function $f(x_i; w)$ that we want to learn.

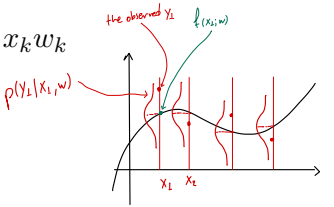
We want to find the parameters w so that the densities $p(y_i|x_i,w)$ "explain well" the given data y_i.

DTU

Learning

Setup: We got some data $(\boldsymbol{y}, \boldsymbol{X})$. We have a way to define a function

$$f(\boldsymbol{x}; \boldsymbol{w}) = \tilde{\boldsymbol{x}}^T \boldsymbol{w} = w_0 + \sum_{k=1}^{M} x_k w_k$$



- **Question:** How do we learn the correct $\boldsymbol{w}$?
- Answer: For each observation, assume:

$$y_i = f(\boldsymbol{x}_i; \boldsymbol{w}) + \varepsilon_i \Rightarrow \varepsilon_i = y_i - f(x_i; w) \quad (1)$$

where ε_i is a normally distributed noise term $\mathcal{N}(\varepsilon_i|\mu = 0, \sigma^2)$. Recall:

$$\mathcal{N}(\varepsilon|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\varepsilon-\mu)^2}{2\sigma^2}} \quad (2)$$

- This means that (since $\varepsilon_i = y_i - f(\tilde{\boldsymbol{x}}_i, \boldsymbol{w})$)

This is the $p(y_i|x_i,w)$ — the prob. density function of y_i given the input x_i and the parameters w

$$p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) = p(\varepsilon_i|\tilde{\boldsymbol{x}}_i, \boldsymbol{w}) \overset{(1)}{\underset{(2)}{=}} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{((y_i - f(\tilde{\boldsymbol{x}}_i, \boldsymbol{w})) - 0)^2}{2\sigma^2}} = \mathcal{N}(y_i|\mu = f(\tilde{\boldsymbol{x}}_i, \boldsymbol{w}), \sigma^2)$$

(ε_i, $\mu = 0$)

Recall from last time: Maximum A Posteriori (MAP) learning

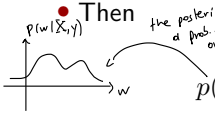
- Consider some data $\boldsymbol{X} = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ and $\boldsymbol{y} = y_1, \ldots, y_N$
- Suppose $\boldsymbol{x}_i$ relates to y_i by some parameters $\boldsymbol{w}$
- **Assume**

"Explain well" implies that this likelihood is maximised.

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w}) = \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i, \boldsymbol{w}), \quad p(\boldsymbol{w}|\boldsymbol{X}) = p(\boldsymbol{w})$$

our prior belief over the parameters

- Then



the posterior is a pmf/density function over the parameters given the data:

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{X})}{p(\boldsymbol{y}|\boldsymbol{X})} = \frac{\prod_{i=1}^{N} p(\boldsymbol{y}_i|\boldsymbol{w}, \boldsymbol{x}_i)p(\boldsymbol{w})}{p(\boldsymbol{y}|\boldsymbol{X})}$$

$p(y|X)$

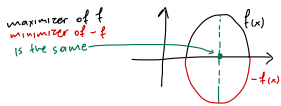
- And maximizing: $\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})$ is equivalent to

(1)(2) ⇒ *Minimize:* $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w})$

$$E(\boldsymbol{w}) = \frac{1}{N}\left[-\sum_{i=1}^{N} \log p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) - \log p(\boldsymbol{w})\right]$$

(1) $\arg\max_x f(x) = \arg\min_x -f(x)$

maximizer of f is the same as minimizer of $-f$



(2) $\arg\max_x f(x) = \arg\max_x \log f(x)$

Because the log(.) is a monotonically increasing function.

Back to the linear model

$$p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \tilde{\boldsymbol{x}}_i^\top \boldsymbol{w})^2}{2\sigma^2}}$$

Optimal $\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})$ found as $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w})$

$$E(\boldsymbol{w}) = \frac{1}{N}\left[-\sum_{i=1}^{N} \log p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) - \log p(\boldsymbol{w})\right]$$

Back to the linear model

$$p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) = \mathcal{N}(y_i|f(\boldsymbol{x}_i, \boldsymbol{w}), \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - f(\boldsymbol{x}_i;\boldsymbol{w}))^2}{2\sigma^2}}$$

Optimal $\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})$ found as $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w})$

$$E(\boldsymbol{w}) = \frac{1}{N}\left[-\sum_{i=1}^{N} \log p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) - \log p(\boldsymbol{w})\right]$$

By assuming a constant (flat prior) we can ignore we obtain (Máximum Likelihood learning)

$\tilde{\boldsymbol{x}}_i^\top = [1, x_{i1}, x_{i2}, \ldots, x_{ik}]$

(Convex optimization problem)

$$E(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} \frac{(y_i - \tilde{\boldsymbol{x}}_i^\top \boldsymbol{w})^2}{2\sigma^2} = \frac{1}{N}\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \tilde{\boldsymbol{x}}_i^\top \boldsymbol{w})^2 \propto \|\boldsymbol{y} - \tilde{\boldsymbol{X}}\boldsymbol{w}\|^2$$

Because E(w) convex we get the minimizer by setting the gradient to zero

$$\frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}} = 2\tilde{\boldsymbol{X}}^\top(\boldsymbol{y} - \tilde{\boldsymbol{X}}\boldsymbol{w}) = 0$$

$$\Rightarrow \boxed{\boldsymbol{w}^* = (\tilde{\boldsymbol{X}}^\top \tilde{\boldsymbol{X}})^{-1}\tilde{\boldsymbol{X}}^\top \boldsymbol{y}}$$

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \quad \tilde{\boldsymbol{X}} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1k} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ & & & & \\ 1 & x_{N1} & x_{N2} & \cdots & x_{Nk} \end{bmatrix} \quad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_k \end{bmatrix}$$

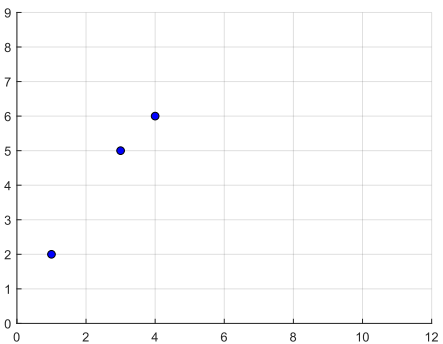
Quiz 2, The linear model

Suppose you observe three points:

$$(x, y) = (1, 2), (3, 5), (4, 6)$$

Knowing what you have learned so far, you first bring these points to the standard format:

$$\boldsymbol{X} = \begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} 2 \\ 5 \\ 6 \end{bmatrix}$$



You wish to train a linear model of the form $y = ax + b$ on this dataset. What is $\boldsymbol{w} = \begin{bmatrix} b \\ a \end{bmatrix}$ Then, compute the prediction of the model at $x = 5$? (the prediction is given as: $y = \tilde{\boldsymbol{x}}^\top \boldsymbol{w}^*$)

A. 6.5

B. 7

C. 7.5

D. 8

E. Don't know.

- first we construct the matrix: $\tilde{\boldsymbol{X}} = \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}$
- then we compute w^* using (1)
- the $w^* = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} b \\ a \end{bmatrix}$
- We can compute the $y = w_1 \cdot x + w_0$ for $x = 5$.

Recall (1) $\boldsymbol{w}^* = (\tilde{\boldsymbol{X}}^\top \tilde{\boldsymbol{X}})^{-1} \tilde{\boldsymbol{X}}^\top \boldsymbol{y}$

The solution is found by first computing $\boldsymbol{w}$ using the standard formula, and remembering to add a column of ones to $\boldsymbol{X}$ to account for the offset. We get:

$$\boldsymbol{w} \approx \begin{bmatrix} 0.7143 \\ 1.3571 \end{bmatrix}$$

Evaluating the model gives $f(5) = y = 7.5$.

Logistic regression

- Assume we are given $(\boldsymbol{X}, \boldsymbol{y})$, but assume y is *binary*: $y_i = 0, 1$
- An idea is to use the Bernoulli distribution

For every input x_i and using the parameters w we compute the θ_i that represents the probability the associated y_i to be equal to 1.

$$p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) = \text{Bernouilli}(y_i|\theta_i) = \theta_i^{y_i}(1-\theta_i)^{1-y_i} \quad (1)$$

Where θ_i depends on $\boldsymbol{w}$ and $\boldsymbol{x}_i$.

- **Problem:** θ_i must belong to the unit interval, but $f(\boldsymbol{x}_i, \boldsymbol{w}) = \tilde{\boldsymbol{x}}_i^\top \boldsymbol{w}$ won't
- **Solution:** Assume

$$\theta_i = \sigma(f(\boldsymbol{x}, \boldsymbol{w})), \quad \text{where } \sigma(z) = \frac{1}{1+e^{-z}} \text{ is the } \textbf{logistic sigmoid}$$

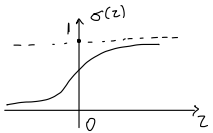
Then

$$(1) \quad -\log p(y_i|\boldsymbol{x}_i, \boldsymbol{w})) =$$

$\sigma : \mathbb{R} \to (0, 1)$

The sigmoid function returns a value in the interval (0,1).

We "squeeze" the output of the function f(x; w) in the interval (0, 1).



Logistic regression

This is a modeling decision because our output y is a binary variable {0, 1}. We want to train the parameters so the $\theta_i = \sigma(f(x_i, w))$ to "explain well" the data i.e. when $y_i = 1$ we want the associated θ_i to be near 1.

- Assume we are given $(\boldsymbol{X}, \boldsymbol{y})$, but assume y is *binary*: $y_i = 0, 1$
- An idea is to use the Bernoulli distribution

$$p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) = \text{Bernouilli}(y_i|\theta_i) = \theta_i^{y_i}(1-\theta_i)^{1-y_i}$$

Where θ_i depends on $\boldsymbol{w}$ and $\boldsymbol{x}_i$.

- **Problem:** θ_i must belong to the unit interval, but $f(\boldsymbol{x}_i; \boldsymbol{w}) = \tilde{\boldsymbol{x}}_i^\top \boldsymbol{w}$ won't
- **Solution:** Assume

$$\theta_i = \sigma(f(\boldsymbol{x}, \boldsymbol{w})), \qquad \text{where } \sigma(z) = \frac{1}{1+e^{-z}} \text{ is the } \textbf{logistic sigmoid}$$

Then

$$\begin{aligned} -\log p(y_i|\boldsymbol{x}_i, \boldsymbol{w})) &= -\log\left[\text{Bern}(y_i|\theta_i = \sigma(\tilde{\boldsymbol{x}}_i^\top \boldsymbol{w}))\right] = -\log\left[\theta_i^{y_i}(1-\theta_i)^{1-y_i}\right] \\ &= -y_i \log(\theta_i) - (1-y_i)\log(1-\theta_i) \end{aligned}$$

Recall from 10 minutes ago: Maximum A Posteriori (MAP) learning

- Consider some data $\boldsymbol{X} = \boldsymbol{x}_1, \ldots, x_N$ and $\boldsymbol{y} = y_1, \ldots, y_N$ $\quad y_i \in \{0, 1\}$?
- **Assume**

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w}) = \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i, \boldsymbol{w}), \quad p(\boldsymbol{w}|\boldsymbol{X}) = p(\boldsymbol{w})$$

- Then

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) = \frac{\prod_{i=1}^{N} p(\boldsymbol{y}_i|\boldsymbol{w}, \boldsymbol{x}_i) p(\boldsymbol{w})}{p(\boldsymbol{y}|\boldsymbol{X})}$$

- Maximizing: $\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})$ is equivalent to $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w})$

In this case we do not have a closed form solution for the parameters $\boldsymbol{w}$

$$E(\boldsymbol{w}) = \frac{1}{N}\left[-\sum_{i=1}^{N} \log p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) - \log p(\boldsymbol{w})\right]$$

- By assuming a constant (flat prior) we can ignore we obtain (Máximum Likelihood learning)

$$E(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} \left[-y_i \log(\theta_i) - (1 - y_i)\log(1 - \theta_i)\right], \quad \theta_i = \sigma(\tilde{\boldsymbol{x}}_i^\top \boldsymbol{w}) = \frac{1}{1 + e^{-\tilde{\boldsymbol{x}}_i^\top \boldsymbol{w}}}$$

Quiz 3, Logistic regression

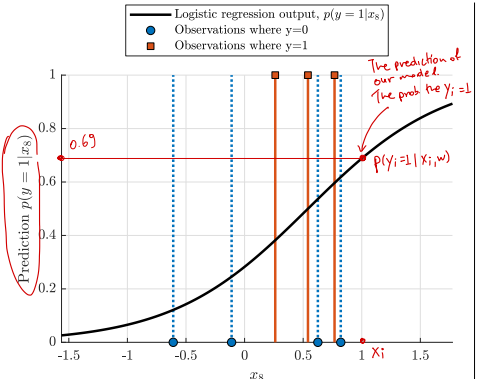


Figure 1: Output of a logistic regression classifier trained on 7 observations from the dataset.

Consider the Avila Bible dataset. We are particularly interested in predicting whether a bible copy was written by copyist 1, and we therefore wish to train a logistic regression classifier to distinguish between copyist one vs. copyist two and three.

To simplify the setup further, we select just 7 observations and train a logistic regression classifier using only the feature x_8 as input (as usual, we apply a simple feature transformation to the inputs to add a constant feature in the first coordinate to handle the intercept term). To be consistent with the lecture notes, we label the output as $y = 0$ (corresponding to copyist one) and $y = 1$ (corresponding to copyist two and three).

In Figure 1 is shown the predicted output probability an observation belongs to the positive class, $p(y = 1|x_8)$. What are the weights?

A. $\begin{bmatrix} -0.93 \\ 1.72 \end{bmatrix}$

B. $\begin{bmatrix} -2.82 \\ 0.0 \end{bmatrix}$

C. $\begin{bmatrix} 1.36 \\ 0.4 \end{bmatrix}$

D. $\begin{bmatrix} -0.65 \\ 0.0 \end{bmatrix}$

E. Don't know.

- We do not have a closed form solution for w.
- We pick a point e.g $x_i = 1$ and we examine the result for every possible set of weights $w = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$
- Bernoulli likelihood: $p(y_i|x_i,w) = \theta_i^{y_i}(1-\theta_i)^{1-y_i}$
- Here we are interested in $p(y_i=1|x_i,w) = \theta_i$
- Where $\theta_i = \frac{1}{1+\exp(-(w_0+w_1x_i))}$
- for A the $p(y_i=1|x_i=1,w) = 0.69$
- This agrees with the figure

The solution is easily found by simply computing the predicted $\hat{y} = p(y = 1|x_8)$-value for an appropriate choice of x_8. Notice that

$$p(y = 1|x_8) = \sigma(\tilde{\boldsymbol{x}}_8^T \boldsymbol{w})$$

If we select $x_8 = 1$ and select the weights as in option A we find $p(y = 1|x_8) = 0.69$, in good agreement with the figure. On the other hand, for the weights in option C we obtain $\hat{y} = 0.85$, for D that $\hat{y} = 0.34$ and finally for B that $\hat{y} = 0.06$. We can therefore conclude that A is correct.

General linear model

$$\text{Linear regr.: } E(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} \|y_i - \tilde{\boldsymbol{x}}_i^\top \boldsymbol{w}\|^2$$

$$\text{Logistic regr.: } E(\boldsymbol{w}) = \frac{-1}{N}\sum_{i=1}^{N} \left[y_i \log(\theta_i) + (1-y_i)\log(1-\theta_i)\right], \ \theta_i = \sigma(\tilde{\boldsymbol{x}}_i^\top \boldsymbol{w})$$

$$\text{GLM } E(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} d(y_i, g(\tilde{\boldsymbol{x}}_i^\top \boldsymbol{w}))$$

General linear model

$$\text{Linear regr.: } E(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} \|y_i - \tilde{\boldsymbol{x}}_i^\top \boldsymbol{w}\|^2$$

$$\text{Logistic regr.: } E(\boldsymbol{w}) = \frac{-1}{N}\sum_{i=1}^{N} \left[y_i \log(\theta_i) + (1-y_i)\log(1-\theta_i)\right], \ \theta_i = \sigma(\tilde{\boldsymbol{x}}_i^\top \boldsymbol{w})$$

$$\text{GLM } E(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} d(y_i, g(\tilde{\boldsymbol{x}}_i^\top \boldsymbol{w}))$$

We call d the cost function and g the link function. In our examples:

$$\text{Lin.reg. : } d(y,z) = \|y-z\|^2, \qquad z = g(\tilde{\boldsymbol{x}}_i^\top \boldsymbol{w}) = \tilde{\boldsymbol{x}}_i^\top \boldsymbol{w}$$

$$\text{Log.reg. : } d(y,z) = -y\log z - (1-y)\log(1-z), \quad z = g(\tilde{\boldsymbol{x}}_i^\top \boldsymbol{w}) = \sigma(\tilde{\boldsymbol{x}}_i^\top \boldsymbol{w})$$

Resources

http://www2.imm.dtu.dk Our interactive regression demo

(`http://www2.imm.dtu.dk/courses/02450/DemoComplexityRegression.html`)