02450 Introduction to Machine Learning and Data Mining

Week 5: Decision trees and linear regression

Bjørn Sand Jensen

4 March 2025

DTU Compute, Technical University of Denmark

Today

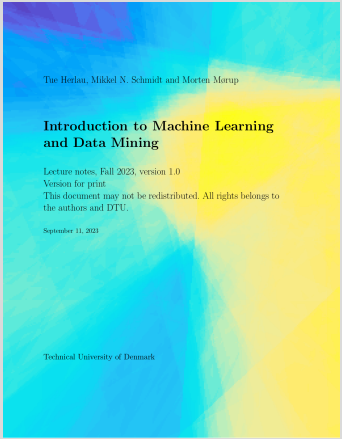
Feedback Groups of the day:

Kerem Ozemre, Mads Frederik Kielsgaard Sørensen, Lea Simonsen, Christian Divert Colberg, Francisco Pelayo, Oscar Lomborg, Nuha Al-Hajjaj, Isak Bastholm Petrin, Emil Bruhn, Alexander Obel Strauss Vestbjerg, Bálint Kostyál, Anya Helle Pritzl, Antonio Javier Cayetano Matamoros, Aleksandra Tunic, Olivier Gaufrès, Jorge Santiago Bajo, Ingeborg Margrethe Holt Nielsen, Robert Spralja, Noa Hvid Shiv, Nicholas Erup Larsen, Yuan Zhang, Kolbeinn Flóki Kristjánsson, Simon Fogh Kristiansen, Thomas Leth Jensen, Martin Mejer, Asbjørn Schroll Graae, Md Imran Hossain, Hoang Anh Do, Levente István Kucsma, Kristian Jerichau Nissen, Vebika Sam Boulos, Harshul Singhal, Valdemar Foster, Alex Jedig, Ehab Fadhil Al-Saoudi, Jonas Amtoft, Liv Cristina Alves Bresnov, Tyler Ramanata, Josephine Højland, Flavio Sarno, Ioannis Bekiaris, Francesco Balducci, Angelos Mekras

Reading/homework material:

Chapter 8,9

P9.1, P8.1, P8.2



Lecture Schedule

1. Introduction
   4 February: C1,C2

Data: Feature extraction, and visualization

2. Summary statistics, similarity and visualization
   11 February: C4,C7
3. Computational linear algebra and PCA
   18 February: C3
4. Probability and probability densities
   25 February: C5, C6

Supervised learning: Classification and regression

5. **Decision trees and linear regression**
   4 March: C8, C9 **(Project 1 due 6 March at 17:00)**
6. Overfitting, cross-validation and Nearest Neighbor
   11 March: C10, C12
7. Performance evaluation, Bayes, and Naive Bayes
   18 March: C11, C13
8. Artificial Neural Networks and Bias/Variance
   25 March: C14, C15
9. AUC and ensemble methods
   1 April: C16, C17

Unsupervised learning: Clustering and density estimation

10. K-means and hierarchical clustering
    8 April: C18 **(Project 2 due 10 April at 17:00)**
11. Mixture models and density estimation
    22 April: C19, C20
12. Association mining
    29 April: C21

Recap

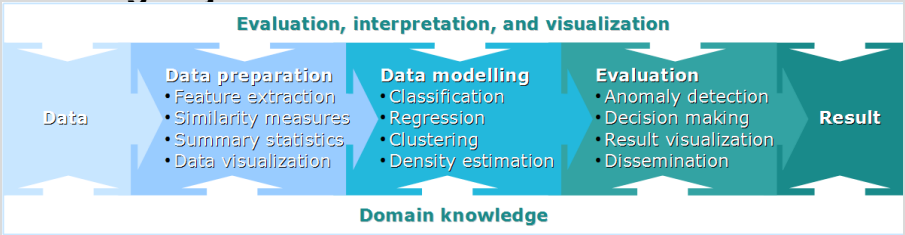
13. Recap and discussion of the exam
    6 May: C1-C21

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

Practicalities and announcements

- Reminder: Project 1 deadline **6th March** (mandatory to pass the course)
- Exam date 28th May 2025. See `https://student.dtu.dk/en/eksamen/exam-dates`
- Exam: "Non-programmable" calculator
  - We are govenered by rule 1) from `https://www.dtu.dk/english/education/examination-timetable` with the additional option of 2 sheets of notes.
  - **Our view:** You are allowed a "non-programmable" calculator, but if you feel you need one due to complicated calculations (e.g. matrix) you are likely doing something wrong. It migth be helpful for log, exp and various fractions.
  - **If in doubt whether your calculator is programmable, you must ask the central study guidance/administartion. It is not in our remit.**

Learning Objectives



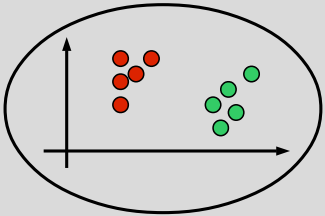
Learning Objectives

- 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

Plan for today:

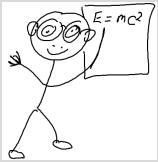
- Lecture 5 (13:00 – $\sim$15:00)
  - Supervised learning
  - Decision trees (classificaiton)
  - Regression trees
  - Linear models for regression (*linear regression*)
  - Linear models for classification (*logistic regression*)
- Exercises ($\sim$ 15:00–17:00)

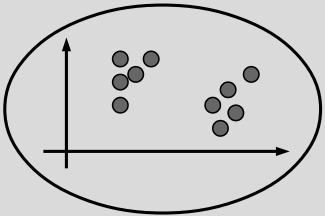
Supervised and Unsupervised learning



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

(Generalize from known examples)

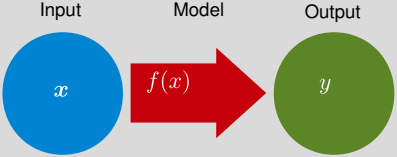


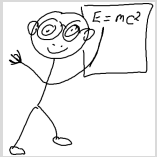


Unsupervised Learning
Input data $\mathbf{x}_n$ alone

(Exploratory analysis)

Supervised learning





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

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

- **Model**
  - Function with parameters $\boldsymbol{w}$ that maps inputs to outputs *(what we are trying to determine)*

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

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

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

- **Types** of supervised learning (in this course)
  - Regression: Continuous output $\boldsymbol{y}$
  - Classification: Discrete output $\boldsymbol{y}$

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

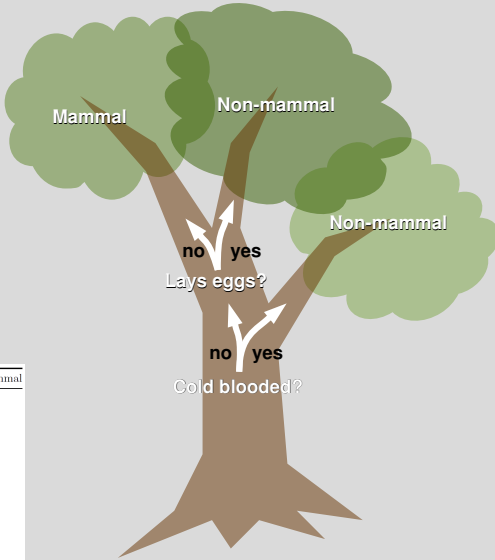
Decision trees

- Do you know the game “20 questions to the professor”?

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

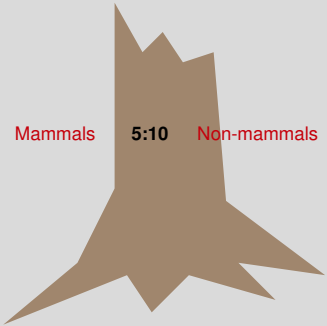
- Ask a serie of questions until a conclusion is reached
- **Example:** Classify **vertebrates** as
  - **Mammal** or
  - **Non-mammal**
- **Learning task:**
  - Which questions should we ask?



Name	x_1: Cold Blooded	x_2: Has Legs	x_3: Lay Eggs	x_4:Has Fur	y: Mammal
Snake	yes	-	yes	-	-
Starfish	yes	-	yes	-	-
Bluebird	-	yes	yes	-	-
Blackbird	-	yes	yes	-	-
Earthworm	yes	-	yes	-	-
Chameleon	yes	-	yes	-	-
Ant	yes	-	yes	-	-
Jellyfish	yes	-	yes	-	-
Snail	yes	-	yes	-	-
Sea Urchin	yes	-	yes	-	-
Dolphin	-	-	-	-	yes
Rat	-	yes	-	yes	yes
Dog	-	yes	-	yes	yes
Monkey	-	yes	-	yes	yes
Lion	-	yes	-	yes	yes

Hunt's algorithm

- Assign all data objects to the root

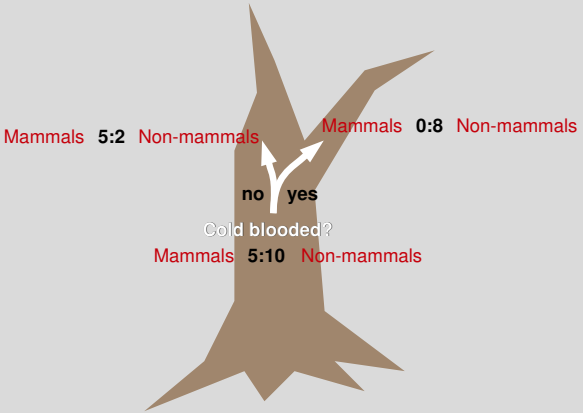


Hunt's algorithm

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

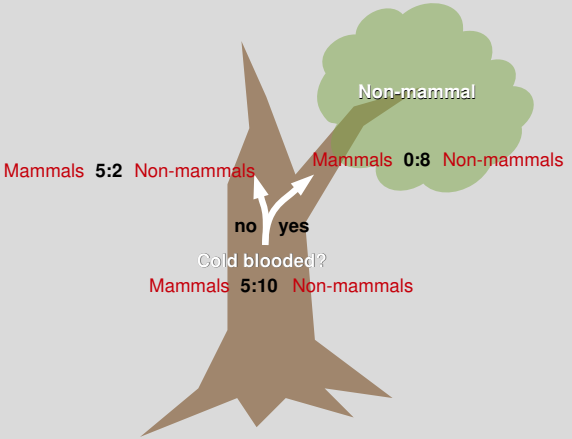
Hunt's algorithm

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



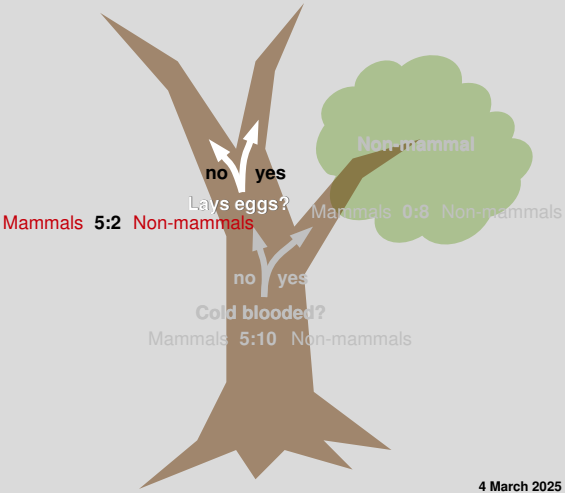
Hunt's algorithm

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



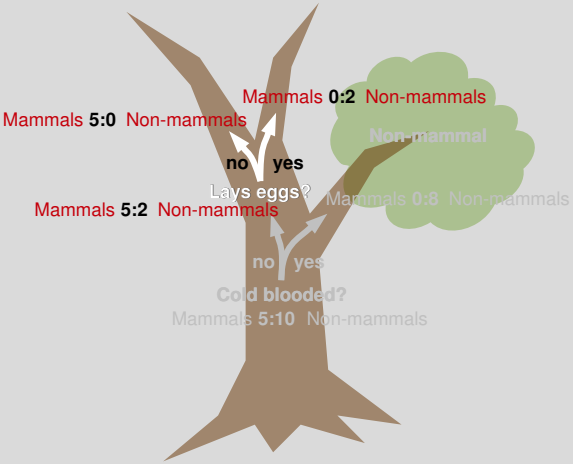
Hunt's algorithm

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



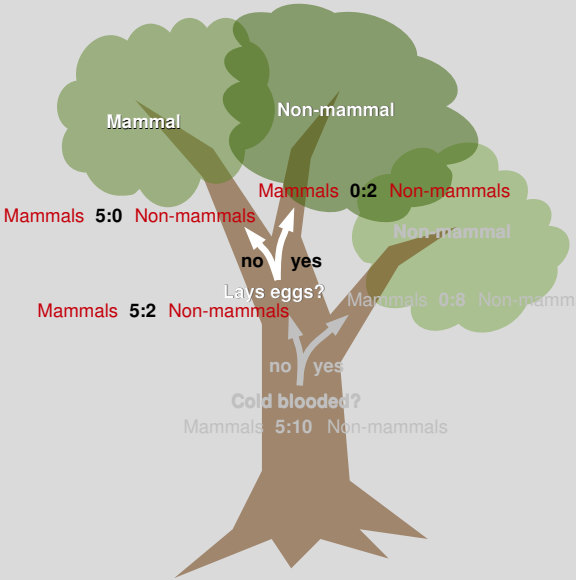
Hunt's algorithm

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



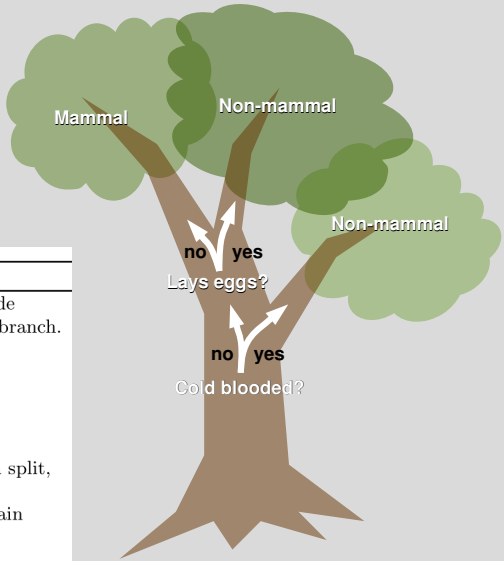
Hunt's algorithm

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



Hunt's algorithm

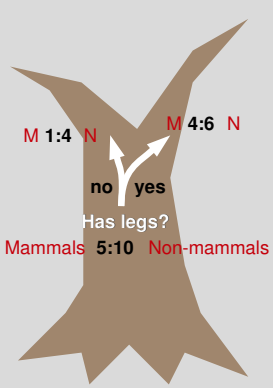
- But how to we find the **best question** at each step?

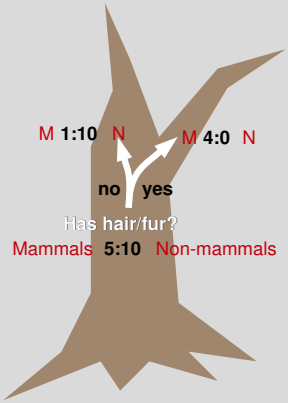


Algorithm 2: Hunt's algorithm for decision trees

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?

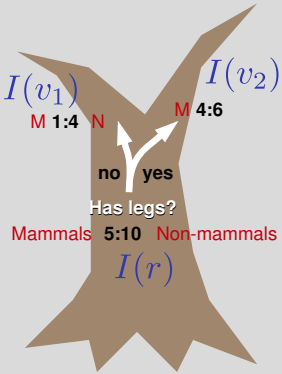




Which split is best?

- 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**:

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



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**:

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

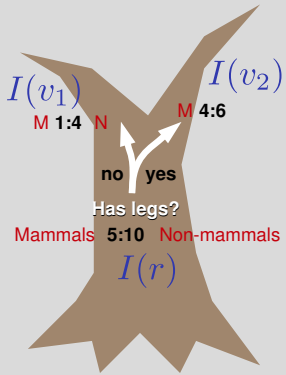
The impurity measure, *I( )*, can be one of the following

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

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

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

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

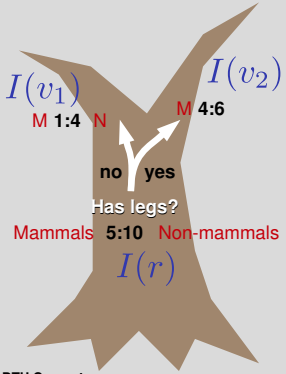


$N(r) = 15,$
$N(v_1) = 5,$
$N(v_2) = 10$

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?

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



$N(r) = 15,$
$N(v_1) = 5,$
$N(v_2) = 10$

e.g. $p(M|r) = \frac{5}{15}$
$p(\text{Non M}|r) = \frac{10}{15}$

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

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

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

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

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

Quiz 1: Impurity gain — solution

We first identify $p(c|v)$:

$$p(\text{Mammal}|r) = 5/15 = 1/3$$

$$p(\text{Non-Mammal}|r) = 10/15 = 2/3$$

$$p(\text{Mammal}|v_1) = 1/5 \qquad p(\text{Mammal}|v_2) = 4/10$$

$$p(\text{Non-Mammal}|v_1) = 4/5 \qquad p(\text{Non-Mammal}|v_2) = 6/10$$

Using Gini impurity we get:

$$I(r) = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 = \frac{4}{9} \ \ (0.44)$$

$$I(v_1) = 1 - \left(\frac{1}{5}\right)^2 - \left(\frac{4}{5}\right)^2 = \frac{8}{25} \ \ (0.32)$$

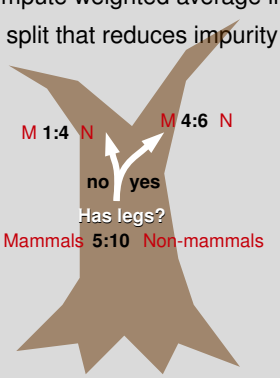
$$I(v_2) = 1 - \left(\frac{4}{10}\right)^2 - \left(\frac{6}{10}\right)^2 = \frac{48}{100} \ \ (0.48)$$

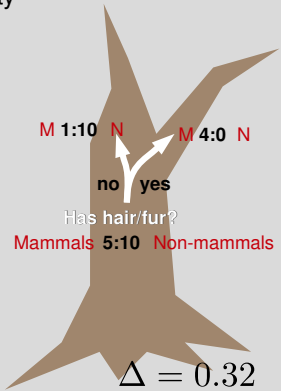
and finally

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

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
  - For each new branch of the tree
  - Compute weighted average impurity
- Choose split that reduces impurity most



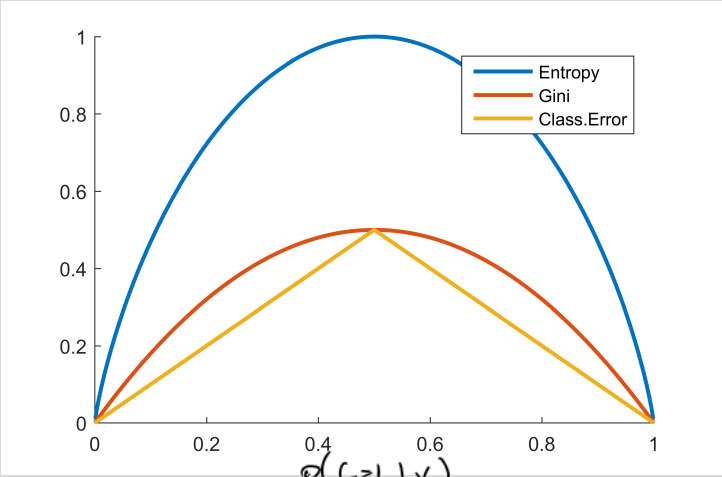


$\Delta = 0.32$

Which impurity measure to use?

For a two class problem:

$C \in \{0, 1\}$



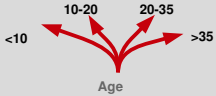
Types of splits to consider?

- **Binary**

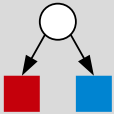
- **Nominal**

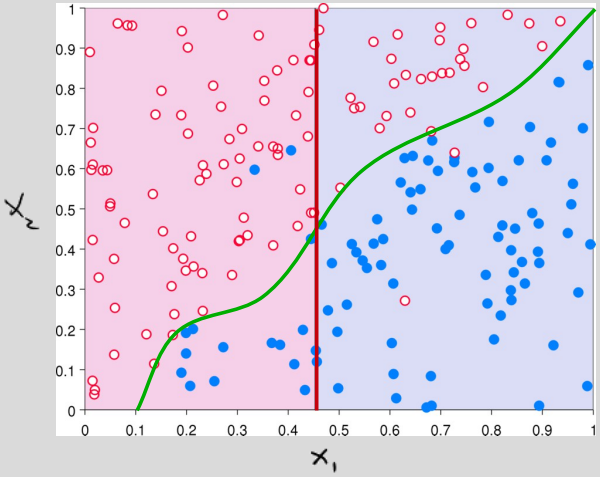
- **Ordinal**

- **Continuous**

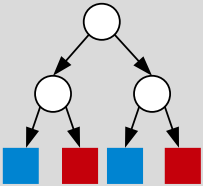


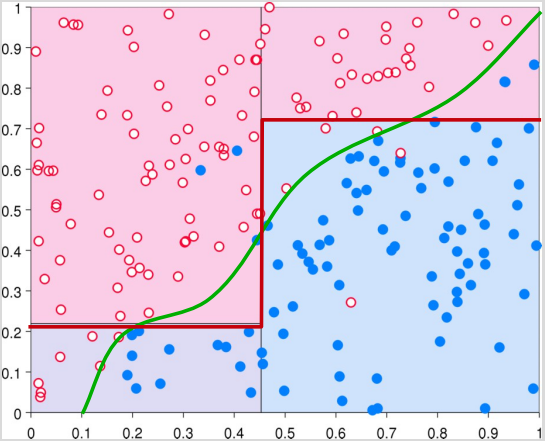
Classification tree



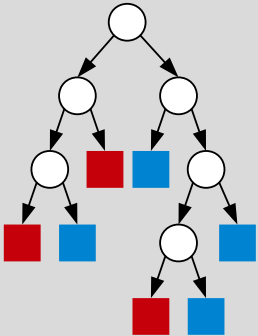


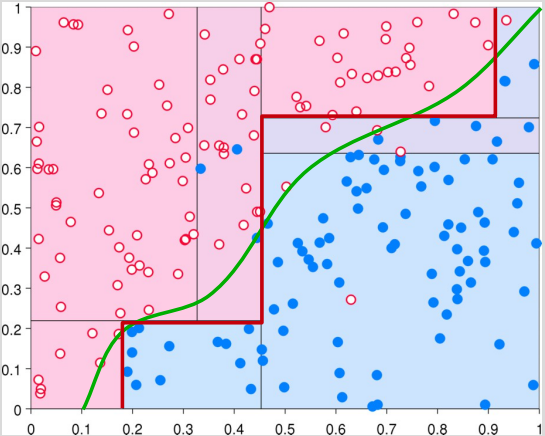
Classification tree



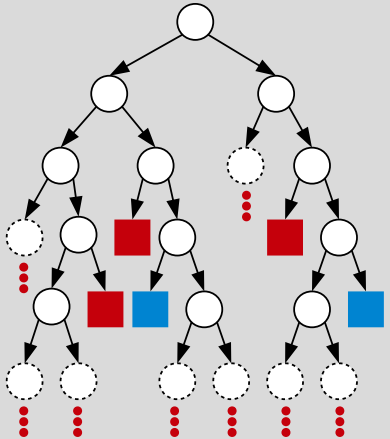


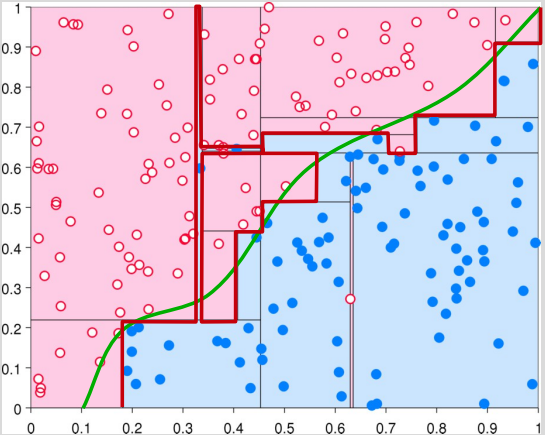
Classification tree





Classification tree





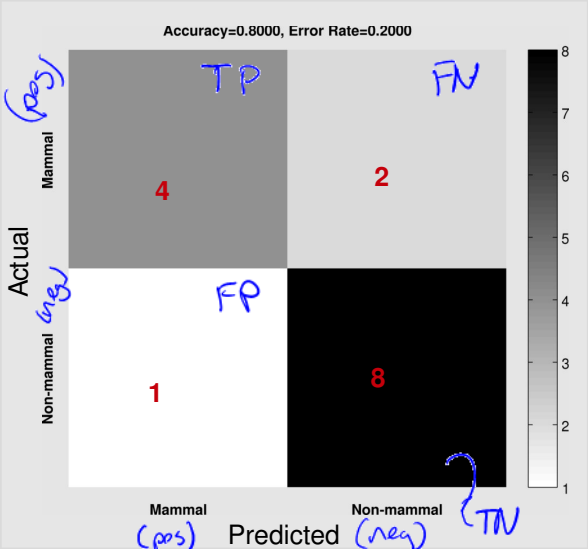
Evaluating a classifier

- Visualizationof actual versus predicted class labels
- Accuracy
  Number of correctly predicted observations divided by the total number of observations

$$\frac{4+8}{4+2+1+8} = 0.8\,(80\%)$$

- Error rate
  Number of incorrectly predicted observations divided by the total number of observations

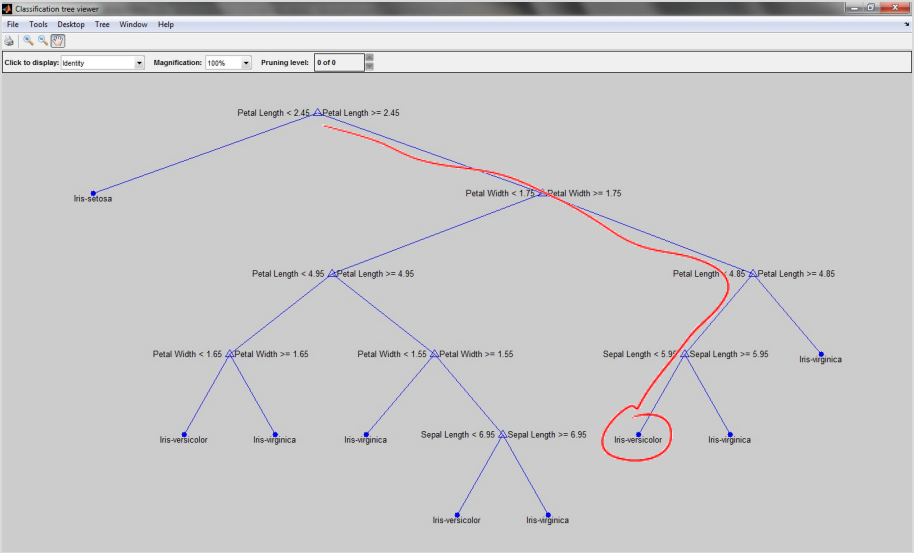
$$\frac{2+1}{4+2+1+8} = 0.2\,(20\%)$$



Example: The Iris Dataset

- **Three types of Iris 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

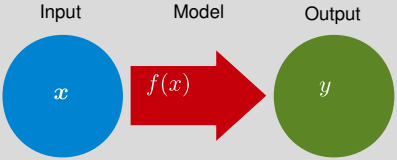
Fower ID	Attribute			
	Sepal Length	Sepal Width	Petal Length	Petal Width
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
.	.	.	.	.
.	.	.	.	.
150	5.9	3.0	5.1	1.8

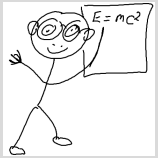


What would the following Iris flower be classified as?

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 (ratio, interval) output

Supervised learning

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

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

- Model
  - Function with parameters $\boldsymbol{w}$ that maps inputs to outputs *(what we are trying to determine)*

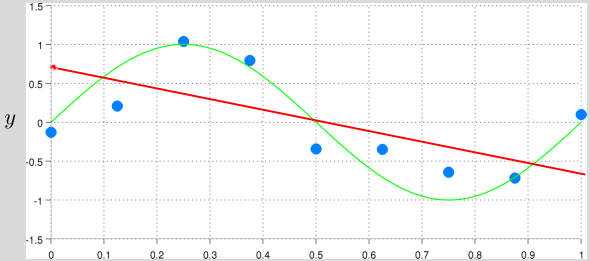
$$f(\boldsymbol{x}; \boldsymbol{w})$$

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

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

Regression

- **Definition:** Learning a function, $f(\boldsymbol{x})$, 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
- **Methods:** Regression trees, linear regression. *Later:* Regularized linear regression/ridge regression, neural networks, ensembles.



Evaluating a regression model

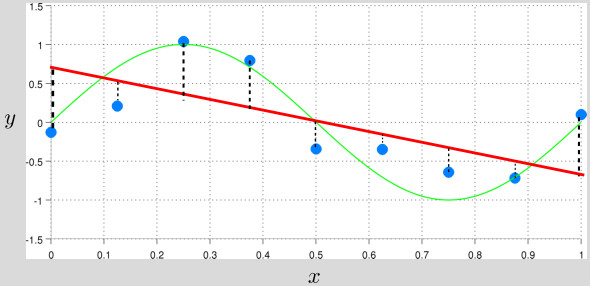
$\hat{y}_i = f(x_i)$

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)



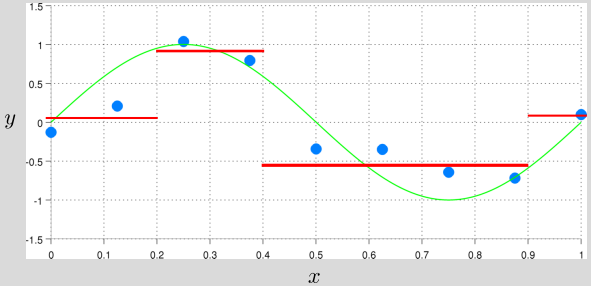
Regression trees

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

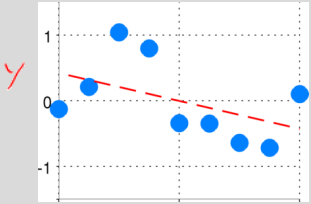
Use mean square error as purity gain

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

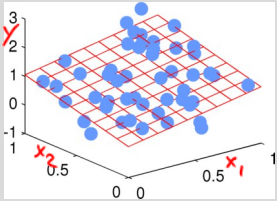


Linear models for regression

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



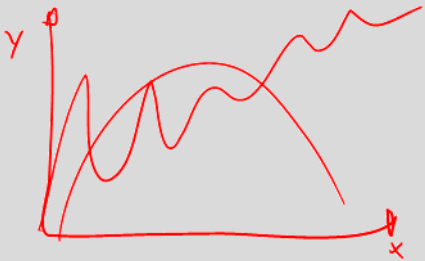
- 2-dimensional inputs
  $f(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2$
  $f(\underline{x}; \underline{w})$



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

Linear models for 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
  $f(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_K x^K$
  $f(x) = w_0 + w_1 sin(x) + w_2 cos(x)$
  $f(\boldsymbol{x}) = w_0 + w_1 sin(x_1) + w_2 x_2^3$



Allows non-linear functions in the input, x, but the model is still linear in the weights!

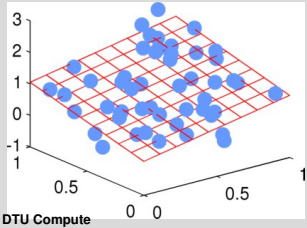
Linear regression

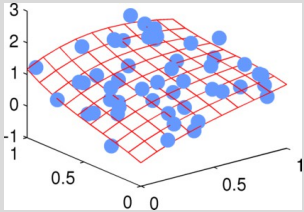
- K-dimensional inputs
  $f(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_K x_K$
- Non-linearly transformed inputs
  $f(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_K x^K$
  $f(x) = w_0 + w_1 sin(x) + w_2 cos(x)$
  $f(\boldsymbol{x}) = w_0 + w_1 sin(x_1) + w_2 x_2^3$

Example:

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

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





Linear regression in vector notation

The linear model can be written compactly using vector notation

$$
\begin{aligned}
f(\boldsymbol{x}) &= w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_K x_K \\
&= \sum_{k=0}^{K} w_k x_k \\
&= \tilde{\boldsymbol{x}}^\top \boldsymbol{w} \\
&= \boldsymbol{w}^\top \tilde{\boldsymbol{x}}
\end{aligned}
$$

where $x_0 = 1$.

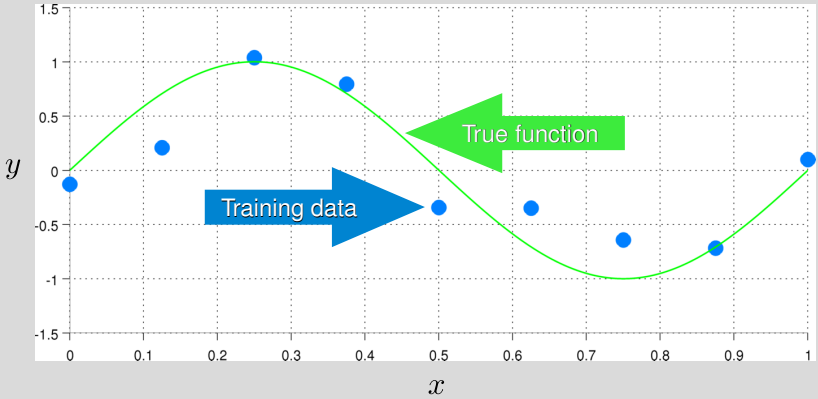
$\tilde{\underline{x}} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_M \end{bmatrix}$

$\underline{X} = \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}$

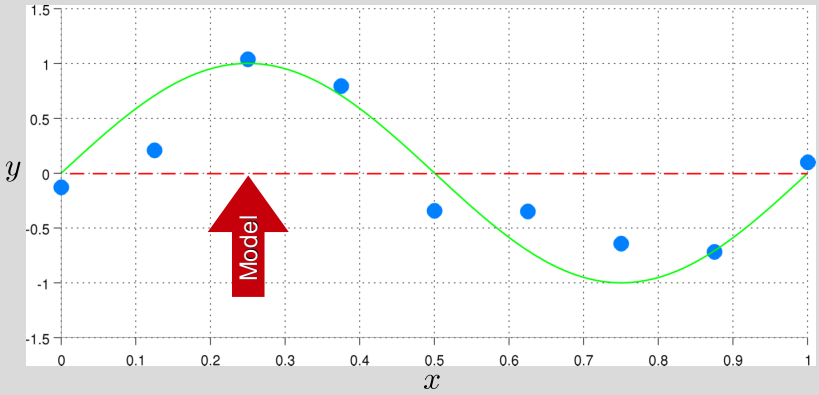
$\underline{\underline{\tilde{X}}} = \begin{bmatrix} 1 & \\ 1 & \\ 1 & \\ 1 & \end{bmatrix}$

$\underline{y} = \underline{\underline{\tilde{X}}}\,\underline{w}$

Linear regression in machine learning



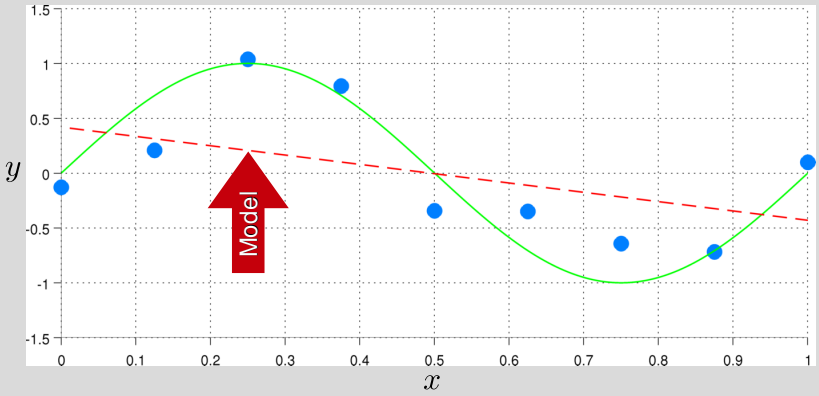
Linear regression



Model:

$f(x) = w_0$

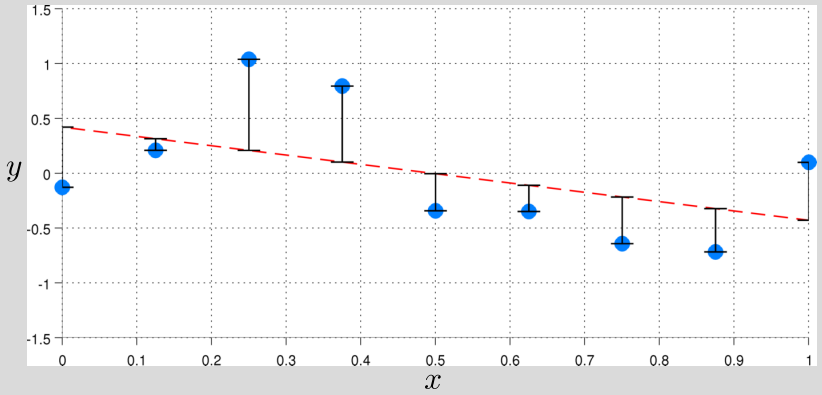
Linear regression



Model:

$f(x) = w_0 + w_1 x$

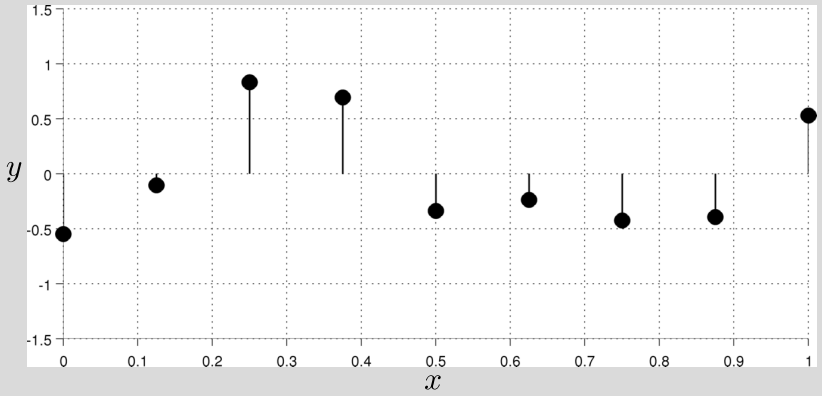
Linear regression — residual error



Model:

$f(x) = w_0 + w_1 x$

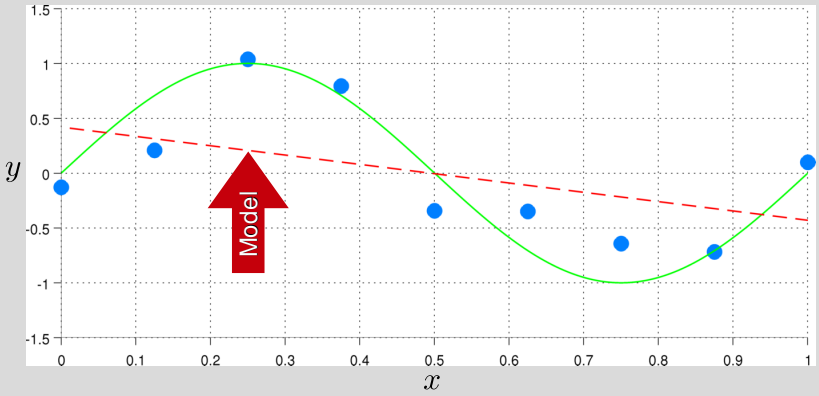
Linear regression — residual error



Model:

$f(x) = w_0 + w_1 x$

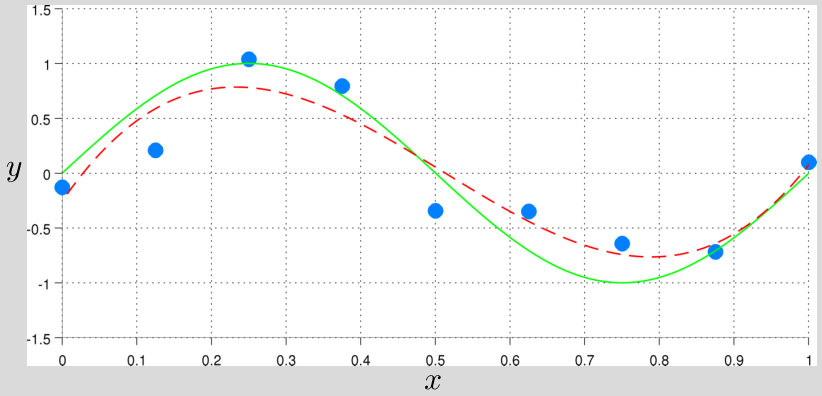
Linear regression



Model:

$f(x) = w_0 + w_1 x$

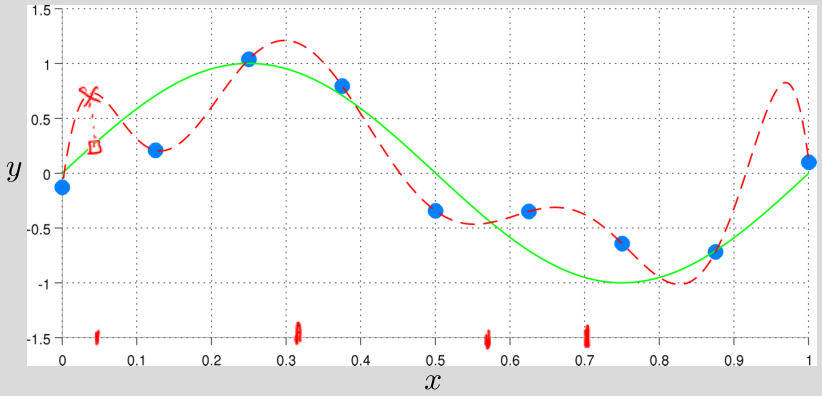
Linear regression



Model:

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

Linear regression

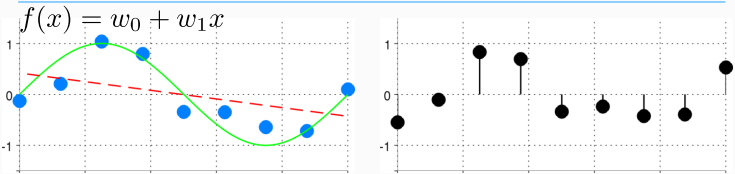


Model:

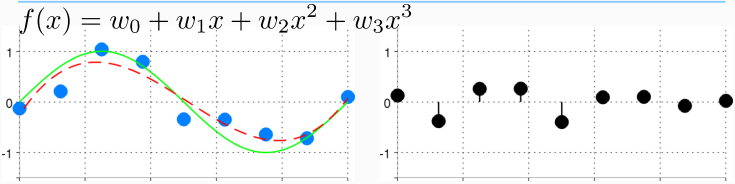
$$f(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + w_5 x^5 + w_6 x^6 + w_7 x^7 + w_8 x^8$$

Linear regression — model order

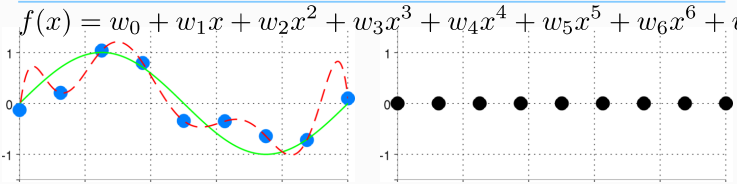
$f(x) = w_0 + w_1 x$



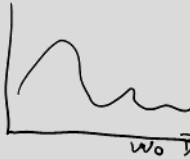
$f(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$



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



Learning - the classic Least Squares solution



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

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

Measuring the discrepancy and defining the loss function $E(\boldsymbol{w})$ function using the squared loss:

$$E(\boldsymbol{w}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i; \boldsymbol{w}))^2 = \frac{1}{N} \left\| \boldsymbol{y} - \tilde{\boldsymbol{X}} \boldsymbol{w} \right\|_2^2 \quad \text{(aka RSS)}$$

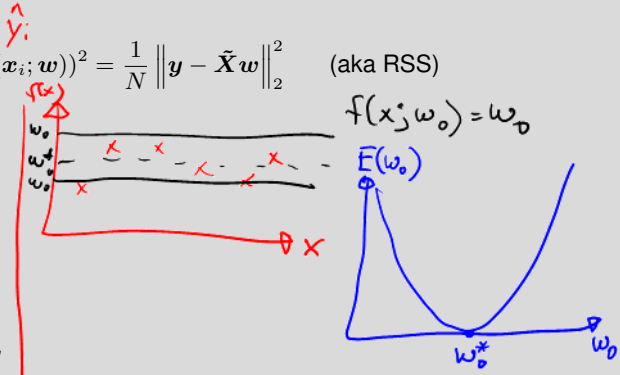
Learning (via optimization): $\boldsymbol{w}$

Optimal $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w})$

Solution:

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

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



Learning - the MAP approach from last week

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 - the MAP approach from last week

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 - the MAP approach from last week

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:

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

Learning - the MAP approach from last week

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:

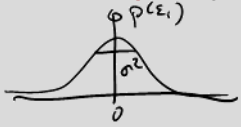
$$\mathcal{N}(x | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- This means that

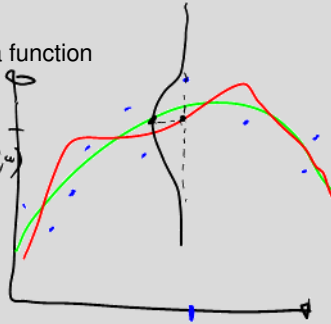
$$p(y_i | \boldsymbol{x}_i, \boldsymbol{w}) =$$

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:

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

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

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

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

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

Recall from last time: Maximum A Posteriori (MAP)

- 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**

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

- Then

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

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

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

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

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)

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

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

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

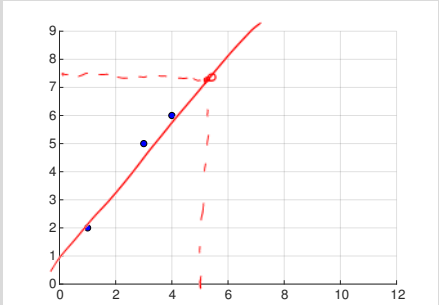
Quiz 2: The linear model

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.

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

Pen & paper solution (+ basic calculator)

① realize you need $\tilde{X}$ not X (ie add const. 1 to X)

$x_{test} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$ $\quad \tilde{X} = \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}$

② Find w^*

a) $\tilde{X}^T \tilde{X} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 4 \end{bmatrix} \begin{bmatrix} 3 & 8 \\ 8 & 26 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}$

b) use that inverse of $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ is $A^{-1} = \frac{1}{ad - cb} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$ so $(\tilde{X}^T \tilde{X})^{-1} = \frac{1}{3 \cdot 26 - 8 \cdot 8} \begin{bmatrix} 26 & -8 \\ -8 & 3 \end{bmatrix}$

$\approx \begin{bmatrix} 1.8571 & -0.5714 \\ 0.5714 & 0.2143 \end{bmatrix}$

③ $\tilde{X}^T y = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 4 \end{bmatrix} \begin{bmatrix} 13 \\ 41 \\ 2 \\ 5 \\ 6 \end{bmatrix}$

④ $w^* = \begin{bmatrix} 1.8571 & -0.5714 \\ 0.5714 & 0.2143 \end{bmatrix} \begin{bmatrix} 1.8571 \cdot 13 - 0.5714 \cdot 41 \\ 0.5714 \cdot 13 + 0.2143 \cdot 41 \end{bmatrix} \approx \begin{bmatrix} 0.715 \\ 1.3581 \end{bmatrix}$

$\begin{bmatrix} 13 \\ 41 \end{bmatrix}$

⑤ predict for $\tilde{x}_{test}$

$\begin{bmatrix} 1 \\ 5 \end{bmatrix}$

$y_{test} = \begin{bmatrix} 0.715 & 1.3581 \end{bmatrix} \begin{bmatrix} 7.51 \end{bmatrix} \approx 7.5$

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:

Python/Matlab/R solution

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

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

Logistic regression — linear models for classification

- Assume we are given $(\boldsymbol{X}, \boldsymbol{y})$, but assume y is *binary*: $y_i \in \{0, 1\}$
- 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 $\boldsymbol{x}_i$.

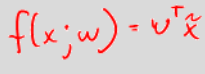
- **Problem:** θ_i must belong to the unit interval, but $f(\boldsymbol{x}_i, \boldsymbol{w}) = \tilde{\boldsymbol{x}}_i^\top \boldsymbol{w}$ won't
- **Solution:** Assume

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

  Then

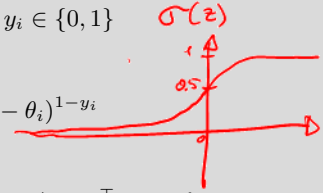
$$-\log p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) =$$

Logistic regression — linear models for classification



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

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



  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

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

  Then

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

Recall: Maximum A Posteriori (MAP)

- Consider some data $\boldsymbol{X} = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ and $\boldsymbol{y} = y_1, \ldots, y_N$
- **Assume**

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

- Then

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

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

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

- 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

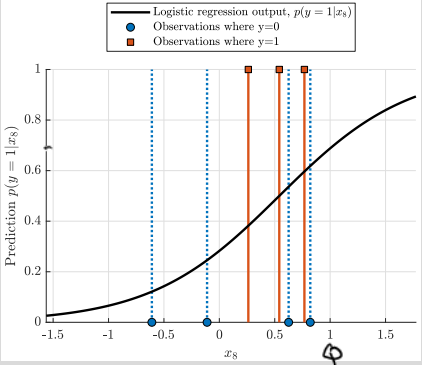


Figure 1: Output of a logistic regression classifier trained on 7 observations from the dataset.

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.

Pen & paper solution (+ basic calculator with exp())

1

2

$p(y|w,x) = \theta^y (1-\theta)^{1-y}$

$y=1$:

$p(y|w,x) = \theta$

$\theta = \sigma(w^T \tilde{x})$

$= \frac{1}{1+e^{-(w_0 + w_1 x_8)}}$

3

let's try $x_8 = 1$

$\frac{1}{1+e^{-(w_0+w_1)}} \approx$

we know from the plot that this should be ≈ 0.7 @ $x_8 = 1$

4 Try out the different options

A: $\frac{1}{1+e^{-(-0.93+1.72)}} \approx 0.69$:)

B: ≈ 0.06

C: ≈ 0.85

D: ≈ 0.34

Get to know $\sigma(z)$!!!

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.

Generalized linear model

$f(x) = w^T x$

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

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

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

Generalized 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:

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

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

Summary

Summary

- Decision trees
  - Classification
  - Regression *(Regression tree)*
- Linear models
  - Regression *(Linear regression)*
  - Classificaiton *(Logistic regression)*
  - Generalized linear models

Resources

DTU Compute Our interactive regression demo

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