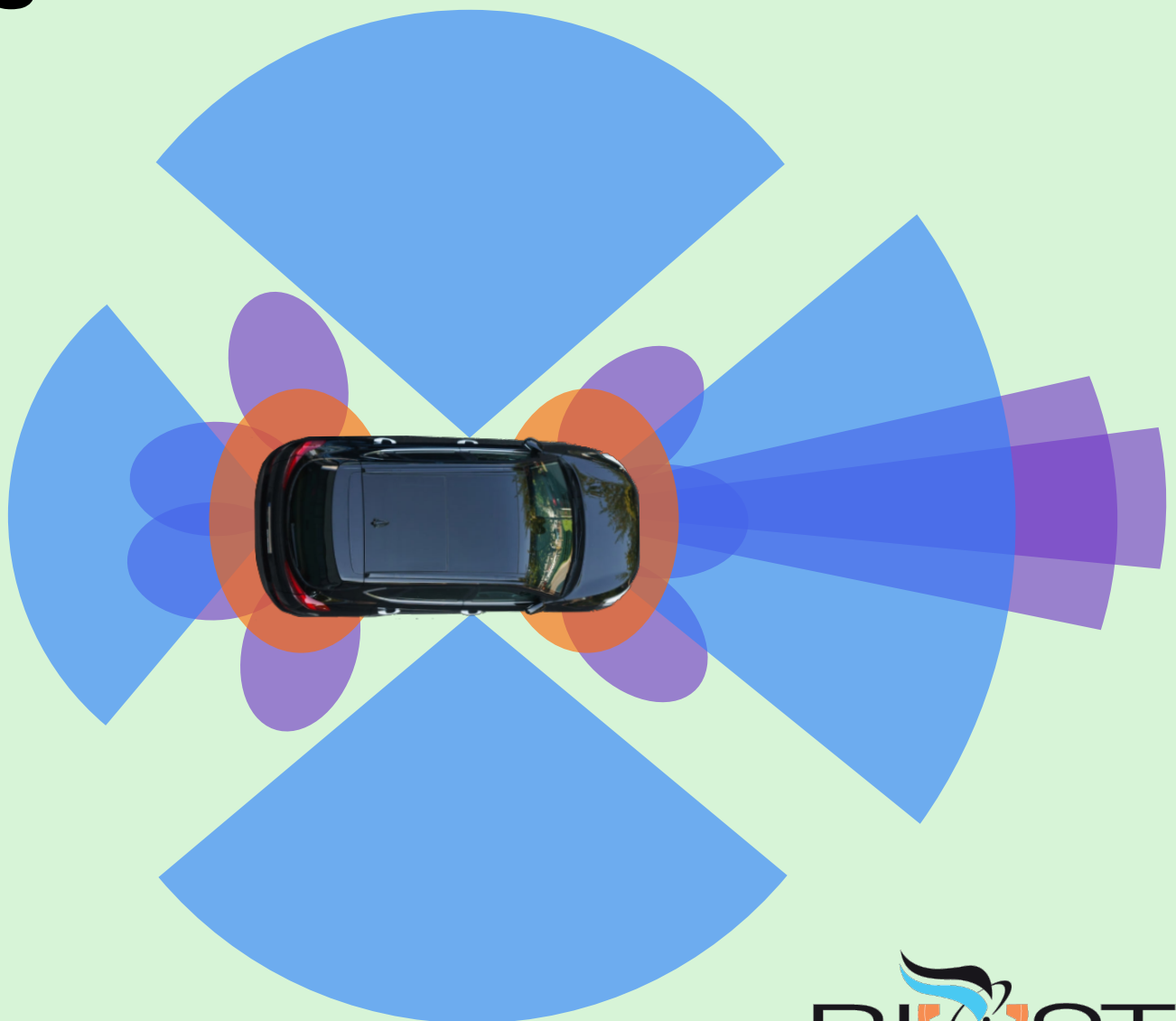


Rodrigo S. Jamisola Jr.

Laboratory Manual

Artificial Intelligence and Machine Learning for Engineers



ISBN: 978-99968-935-0-6

Copyright ©2021 by Botswana International University of Science and Technology (BIUST). All rights reserved.

Copyright and Reprint Permission:

Reproduction is permitted with credit to Botswana International University of Science and Technology.

Other copying, reprint, or reproduction requests should be addressed to Office of the Deputy Vice-Chancellor for Academic Affairs, Botswana International University of Science and Technology, Palapye, Botswana.

ISBN: 978-99968-935-0-6

BIUST Contact Information:

Phone: +267-493-1000

Email: jamisolar@biust.ac.bw

Preface

This volume is a collection of all the laboratory exercises I developed for MCTE 512 Artificial Intelligence and Machine Learning for Engineers throughout my teaching years in BIUST. This is designed to supplement the lecture materials that I developed for the same module. The lectures run on an average of thirteen weeks, and these exercises are designed to deepen the understanding of the students through hands-on computations and computer programming.

There are six major topics covered by the lectures and are reinforced through the exercises here. These are Linear Regression, Decision Trees, Bayes Rule, MLE and MAP, Naive Bayes, Support Vector Machine, and Neural Network. If time allows, additional exercises will be added to cover Deep Learning, Reinforced Learning, and Markov Decision Processes.

Most of the exercises entail the students doing manual computations. I believe that the student will gain a deeper understanding of the subject matter by performing more manual computations than jumping straight into coding programs to implement the solution. However, as the exercises progress towards the end of the class, I add more and more programming exercises for the student to gain experience in using artificial intelligence and machine learning tools that are freely available online. With the combination of both extensive computations and programming experience using freely available resources online, the student is prepared to handle any artificial intelligence and machine learning problems in the real world.

At the end of this book, I attached the course syllabus and the lecture slides as references to the students.

September 9, 2021
Palapye, Botswana

Rodrigo S. Jamisola, Jr.
Associate Professor in Robotics and Machine Learning
Dept. of Mechanical, Energy, and Industrial Engineering
Botswana International University of Science and Technology

Table of Contents

Preface	v
Professor Rodrigo S. Jamisola, Jr.	
Table of Contents	vi
.	
Laboratory Exercise No. 1	1
Background and Motivation	
Laboratory Exercise No. 2	2
Polynomial Regression	
Laboratory Exercise No. 3	3
Decision Trees	
Laboratory Exercise No. 4	4
Decision Trees with Information Gain	
Laboratory Exercise No. 5	5
Bayes' Rule	
Laboratory Exercise No. 6	6
Naive Bayes Rule	
Laboratory Exercise No. 7	7
Support Vector Machine	
Laboratory Exercise No. 8	8
Artificial Neural Network	
Appendix	10
.	
MCTE 512 Syllabus	11
.	
Lecture Topic 1	13
Introduction to Machine Learning	
Lecture Topic 2	28
Polynomial Regression	
Lecture Topic 3	40
Decision Trees	
Lecture Topic 4	71
Bayes' Rule	
Lecture Topic 5	100
Naive Bayes	
Lecture Topic 6	127
Support Vector Machine	

Lecture Topic 7	175
Artificial Neural Network	
Abstract	Back Cover
.	
About the Author	Back Cover
.	

MCTE 512 Artificial Intelligence and Machine Learning for Engineers Laboratory Exercise No. 1 – Background and Motivation

by Prof. Rodrigo S. Jamisola Jr., First Semester 202 __-202 __, BIUST

Current Trends on Research and Applications in Machine Learning

1. Why is Machine Learning important? List all technologies that you think has machine learning on them.
2. History and early advances of Machine Learning
3. Current research areas in Machine Learning
4. If you were given a chance, what could be a technology using machine learning that you would want to develop. Why do you choose this, and tell us how are you going to develop it?

Minimum of five pages and maximum of 10 pages, single space. Due on _____ just before start of laboratory class.

MCTE 512 Artificial Intelligence and Machine Learning for Engineers Laboratory Exercise No. 2 – Polynomial Regression

by Prof. Rodrigo S. Jamisola Jr., First Semester 202 __-202 __, BIUST

- Given function: $\sin(2\pi x)$
- Estimate this function using: a polynomial function

$$y(x, \mathbf{w}) = w_0x^0 + w_1x^1 + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j$$

- Use a polynomial of degree three, $M = 3$, and six equally spaced data points, $N = 6$, to approximate the function $\sin(2\pi x)$ without noise.
- Find the optimal value of $\mathbf{w}^* = [w_0^*, w_1^*, w_2^*]$ such that the error term is minimized

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2.$$

- Update the weights such that

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial E(\mathbf{w})}{\partial w_0} \\ \frac{\partial E(\mathbf{w})}{\partial w_1} \\ \frac{\partial E(\mathbf{w})}{\partial w_2} \\ \frac{\partial E(\mathbf{w})}{\partial w_3} \end{bmatrix}$$

where $\eta = 0.1$, and

$$\frac{\partial E(\mathbf{w})}{\partial w_i} = \sum_{n=1}^N \left\{ \sum_{j=0}^M w_j x_n^j - t_n \right\} x_n^i$$

1. **(50 points)**. Perform hand computation for the problem above up to three iterations, that is, computing three updated values of $\mathbf{w}$. Manually draw the polynomial using the final values of $\mathbf{w}$ together with the manual drawing of the function $\sin(2\pi x)$.
2. **(50 points)**. Use a programming platform of your choice, e.g., Matlab, Python, etc., to solve for the values of $\mathbf{w}$ until error $E(\mathbf{w}) < 0.01$. Plot the polynomial using the final values of $\mathbf{w}$ together with the plot of the function $\sin(2\pi x)$.

Due on _____ just before start of laboratory class.

MCTE 512 Artificial Intelligence and Machine Learning for Engineers Laboratory Exercise No. 3 – Decision Trees

by Prof. Rodrigo S. Jamisola Jr., First Semester 202 __-202 __, BIUST

Boolean Function using Decision Trees

- Given function: $\phi = (A \wedge \neg B) \vee (\neg A \wedge \neg C) \vee (B \wedge \neg C)$
- Estimate this function using: a Decision Tree
- Write your answers manually and submit as a PDF file.

1. **(3.33 points)**. Compose the Truth Table
2. **(6.67 points)**. Build two possible simplified Decision Trees based on the Truth Table above.

Due on _____ just before start of laboratory class.

MCTE 512 Artificial Intelligence and Machine Learning for Engineers Laboratory Exercise No. 4 – Decision Trees with Information Gain

by Prof. Rodrigo S. Jamisola Jr., First Semester 202 __-202 __, BIUST

Choosing Decision Trees Nodes using Information Gain

- Estimate a function using: a Decision Tree
- Create a decision tree using manual computation, by displaying the node, branch, and leaf and information gains used for selecting the attribute in the corresponding layer.
- Compute until the leaf layer, that is, until there is no more possible split where the information gain becomes zero.
- Use the data in the table below.
- Write your answers manually and submit as a PDF file.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Use the following equations:

$$IG(X_a) = H(Y) - H(Y|X_a)$$

$$H(Y) = - \sum_{i=1}^k P(Y = y_i) \log_2 P(Y = y_i)$$

$$H(Y|X_a) = - \sum_{j=1}^v P(X_a = x_j) \sum_{i=1}^k P(Y = y_i | X_a = x_j) \log_2 P(Y = y_i | X_a = x_j)$$

Due on _____ just before start of laboratory class.

MCTE 512 Artificial Intelligence and Machine Learning for Engineers Laboratory Exercise No. 5 - Bayes' Rule

by Prof. Rodrigo S. Jamisola Jr., First Semester 202 __-202 __, BIUST

Using Bayes' Rule to Increase Probability of Winning

- Estimate a function using: Bayes Rule
- Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?
- Use Bayes' Theorem to solve for the probability of:
 1. Not switching
 2. Switching
- Run 1000 experiments in Matlab to show the results.



Figure 1: The Monty Hall Problem.

Due on _____ just before start of laboratory class.

MCTE 512 Artificial Intelligence and Machine Learning for Engineers Laboratory Exercise No. 6 - Naive Bayes Rule

by Prof. Rodrigo S. Jamisola Jr., First Semester 202 __-202 __, BIUST

To Play Tennis or Not?

- Estimate a function using: Naive Bayes Rule
- Given the table below, use Naive Bayes Rule to determine whether the person will Play Tennis or Not with the following parameters

Day	Outlook	Temp	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Sunny	Mild	High	Strong	No

- Parameters are given as {Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong}
- Use two methods to compute:
 - Using Maximum a Posteriori Method
 - Using Maximum Likelihood Method

Due on _____ just before start of laboratory class.

MCTE 512 Artificial Intelligence and Machine Learning for Engineers Laboratory Exercise No. 7 - Support Vector Machine

by Prof. Rodrigo S. Jamisola Jr., First Semester 202 __-202 __, BIUST

Use Support Vector Machine for Classification

- Estimate a function using: Support Vector Machine
- For 50%: Classify the data in Fig. 1 using linear classification.
- For 50%: Classify the data in Fig. 2 by transforming the data in feature space and define the linear classification in that space.

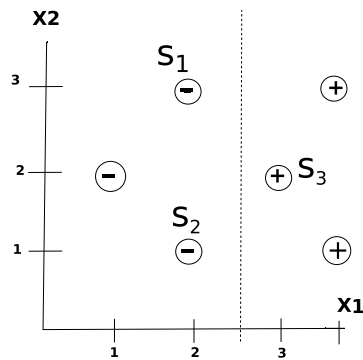


Figure 1: Data points for linear classification.

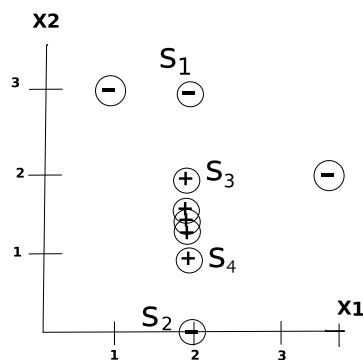


Figure 2: Data points for non-linear classification.

Due on _____ just before start of laboratory class.

MCTE 512 Artificial Intelligence and Machine Learning for Engineers Laboratory Exercise No. 8 - Artificial Neural Network

by Prof. Rodrigo S. Jamisola Jr., First Semester 202 __-202 __, BIUST

Use Artificial Neural Network for Classification

- Estimate a function using: Artificial Neural Network
- Given the data from the XOR table below, use back propagation approach to update the weights w_{jk} and v_{ij} of a sigmoid function neural network with one hidden layer.

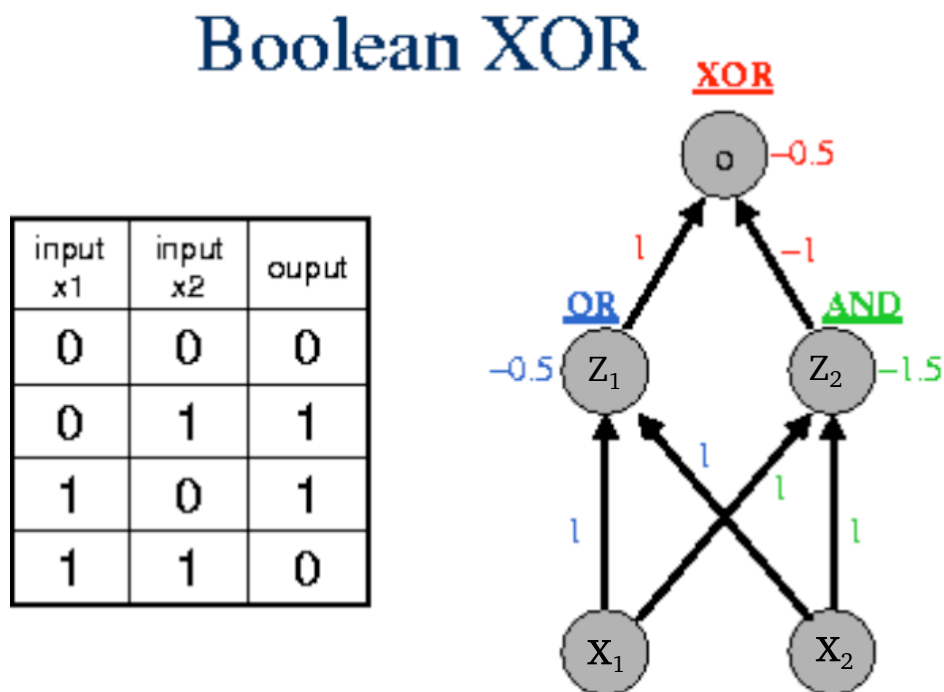


Figure 1: Data for XOR classification using neural network with one hidden layer.

- Use the data $x_1 = 1, x_2 = 0, t = 1$.
- Initialize the weights as

$$\mathbf{w}_1 = [w_{10}, w_{11}, w_{12}]^T = [0.2, 0.4, 0.1]^T$$

$$\mathbf{v}_1 = [v_{10}, v_{11}, v_{12}]^T = [-0.1, 0.2, -0.1]^T$$

$$\mathbf{v}_2 = [v_{20}, v_{21}, v_{22}]^T = [0.3, 0.5, -0.2]^T$$

- Update the weights w_{kj} and v_{ji} in one pass (one feedforward and one backpropagation) using the equations below.

$$\begin{aligned} w_{kj} &\leftarrow w_{kj} + \Delta w_{kj} \\ &\leftarrow w_{kj} + \alpha \delta_j z_j \\ &\leftarrow w_{kj} + \alpha y_k (1 - y_k) (t_k - y_k) z_j \end{aligned}$$

$$\begin{aligned} v_{ji} &\leftarrow v_{ji} + \Delta v_{ji} \\ &\leftarrow v_{ji} + \alpha \delta_j x_i \\ &\leftarrow v_{ji} + \alpha z_j (1 - z_j) \sum \delta_k w_{kj} x_i \\ &\leftarrow v_{ji} + \alpha z_j (1 - z_j) \sum y_k (1 - y_k) (t_k - y_k) w_{kj} x_i \end{aligned}$$

- Use the updated weights to run another pass (one feedforward and one backpropagation) using the data $x_1 = 1, x_2 = 1, t = 0$.
- Use the updated weights and check if they can classify the XOR output correctly.

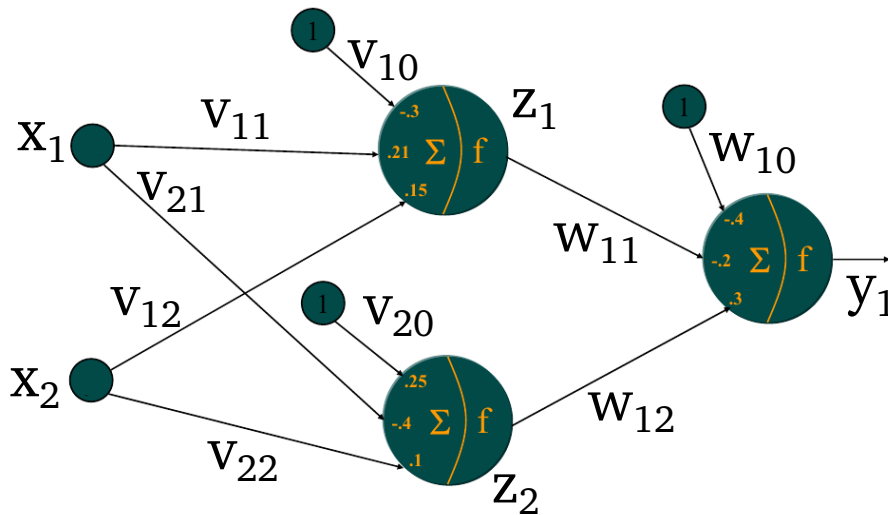


Figure 2: The neural network model showing the corresponding inputs, outputs, and weights.

Due on _____ just before start of laboratory class.

Appendix



Botswana International University of Science and Technology
College of Engineering and Technology
Dept. of Mechanical, Energy and Instrumentation Engineering

Module Information: Artificial Intelligence and Machine Learning for Engineers

- | | |
|--------------------------------|---|
| 1. Course Code | : MCTE 512 |
| 2. No. of Credits | : 3 Credits |
| 3. Term | : 1st Sem, 2021-2022 |
| 4. Prerequisite | : Year 5 |
| 5. Duration | : 12 weeks |
| 6. Lecture hours per week | : 2 hrs |
| 7. Lab tutorial/hours per week | : 2 hrs |
| 8. Faculty | : PROF. RODRIGO S. JAMISOLA, JR PHD
: Room 112, Engineering Building
: Email -- jamisolar@biust.ac.bw |
| 9. Class Schedule | : Lecture – Friday 1100-1250 hrs, Block 108/044
: Lab – Thursday, 1200-1450 hrs, Physics Computer Lab |
| 10. Consultation Hours | : Via Zoom, Mon 0900-1000, Wed 0900-1000 |

11. Course Description: This course Artificial Intelligence and Machine Learning for Engineers is an introductory course to machine intelligence. Machine learning, that includes artificial intelligence, is considered to give computers the ability to learn without being explicitly programmed. The students will familiarize with the major machine learning algorithms, namely, Decision Trees, Bayes Rule, Support Vector Machines, Neural Networks and other techniques, and will understand their strengths and weaknesses. At the same time, the students are expected to identify machine learning techniques that are more appropriate for certain problems, and will study how to design a learning experiment and evaluate the efficacy of the learned solution.

12. Objective and Values:

- To familiarize the students with the introductory concepts of artificial intelligence, neural networks, or machine learning as a whole
- To enable the students to demonstrate knowledge and skills in the techniques of machine learning and use them in many real-world applications

13. Resources:

- Textbook:** Machine Learning (McGraw-Hill Science/Engineering/Math, March 1, 1997) by Tom. M. Mitchell, ISBN: 0070428077.
- Reference Books:**
 - Pattern Recognition and Machine Learning (Springer, 2006) by Christopher M. Bishop, ISBN: 9780387310732.
 - Machine Learning: A Probabilistic Perspective (MIT Press, 2012) by Kevin P. Murphy, ISBN: 9780262018029.
- Software:** Matlab or Python (freeware)
- Learning Management System:** Log in to <https://biust.blackboard.com> for course announcements, availability of course materials, submission of labs and assignments, and grades information.

14. Assessment:

- Requirements:** There will be two (2) tests, at least four (4) labs, and one (1) final exam
- Grading:**

Description	Percentage (%)
Average of Long Tests:	25
Homework, Short Quizzes and Laboratory Exercises	15
Final Exams:	60

Description	Percentage (%)
Total	100

c. Passing: 50 %

15. Duly Performed (DP) Requirements: Students qualify to write the final examination for this module if they have no less than 35% for the continuous assessment mark AND attendance of no less than 80% for lectures and lab sessions.

16. Cheating: Any student caught cheating during tests or lab and assignment submissions will be recommended to Dept HOD for disciplinary actions. Final and Supplementary Examinations cheating policies apply.

17. Turnitin: All submissions through Turnitin in Blackboard must be below 50% similarity grade. Equal to or higher than this value, the submission will be graded automatically zero.

18. Weekly Course Schedule: (Topics does not necessarily follow the order presented.)

WEEK	TOPIC	NO. OF HOURS
1	Introduction to Machine Learning	3
2	Linear Regression	3
3	Decision Trees	3
4	Bayes Rule, MLE and MAP	3
5	Naive Bayes	3
6	Support Vector Machine	3
7	Neural Network	3
8	Deep Learning	3
9	Reinforced Learning and Markov Decision Processes	3

MCTE 512

Artificial Intelligence and Machine Learning for Engineers

Lecture 1 - Introduction to Machine Learning

Rodrigo S. Jamisola Jr.

Associate Professor in Robotics and Machine Learning
Simulations, Machine Learning, Robotics and Technopreneurship Laboratory (SMaRT Lab)
Dept. of Mechanical, Energy and Industrial Engineering
Faculty of Engineering and Technology
Botswana International University of Science and Technology

Machines and Intelligence

- When are Intelligent things?
 - ability to decide on its own
- What is Machine Learning?
 - ability to gather information from data

Adopted from: <http://www.cs.toronto.edu/~hinton/> with figures from Chris Bishop, "Pattern Recognition and Machine Learning"

AI and Machine Learning

- What is Artificial Intelligence?
 - perform tasks normally requiring human intelligence
 - knowledge
 - decision
- What is Machine Learning?
 - the ability to gather information from data
 - data
 - information

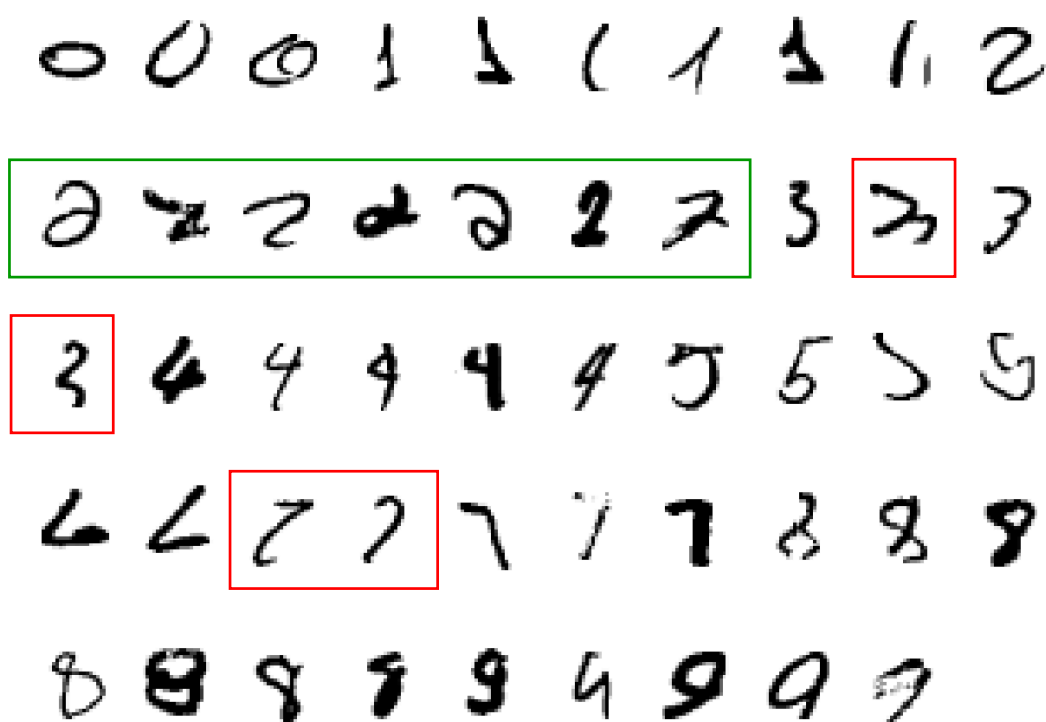
Adopted from: <http://www.cs.toronto.edu/~hinton/> with figures from Chris Bishop, "Pattern Recognition and Machine Learning"

Processes in Machine Learning

- Write a program to recognize a face
 - we don't know what program to write to copy how humans do it
 - even if we have, it will be a very complicated program
- Machine learning takes data and produces a program
 - program looked different from hand-written programs
 - may contain millions of numbers
 - program may predict for unseen data

Adopted from: <http://www.cs.toronto.edu/~hinton/> with figures from Chris Bishop, "Pattern Recognition and Machine Learning"

A Classic Problem: What makes 2?



Adopted from: <http://www.cs.toronto.edu/~hinton/> with figures from Chris Bishop, "Pattern Recognition and Machine Learning"

Some Examples Best Solved by Learning Algorithms

- Recognizing patterns:
 - Facial identities or facial expressions
 - Handwritten or spoken words
 - Medical images
- Generating patterns:
 - Generating images or motion sequences
- Recognizing anomalies:
 - Unusual sequences of credit card transactions
 - Unusual patterns of sensor readings in a nuclear power plant or unusual sound in your car engine.
- Prediction:
 - Future stock prices or currency exchange rates

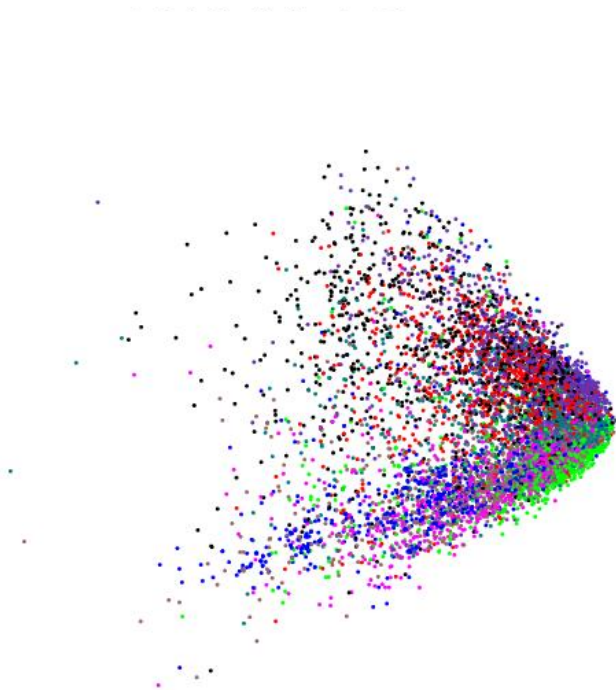
Adopted from: <http://www.cs.toronto.edu/~hinton/> with figures from Chris Bishop, "Pattern Recognition and Machine Learning"

Web-Based Examples

- The web contains a lot of data. Tasks with very big datasets often use machine learning
 - especially if the data is noisy or non-stationary
- Spam filtering, fraud detection:
 - The enemy adapts so we must adapt too
- Recommendation systems:
 - Lots of noisy data. Million dollar prize!
- Information retrieval:
 - Find documents or images with similar content
- Data Visualization:
 - Display a huge database in a revealing way

Adopted from: <http://www.cs.toronto.edu/~hinton/> with figures from Chris Bishop, "Pattern Recognition and Machine Learning"

Displaying the structure of a set of documents using Latent Semantic Analysis (a form of PCA)

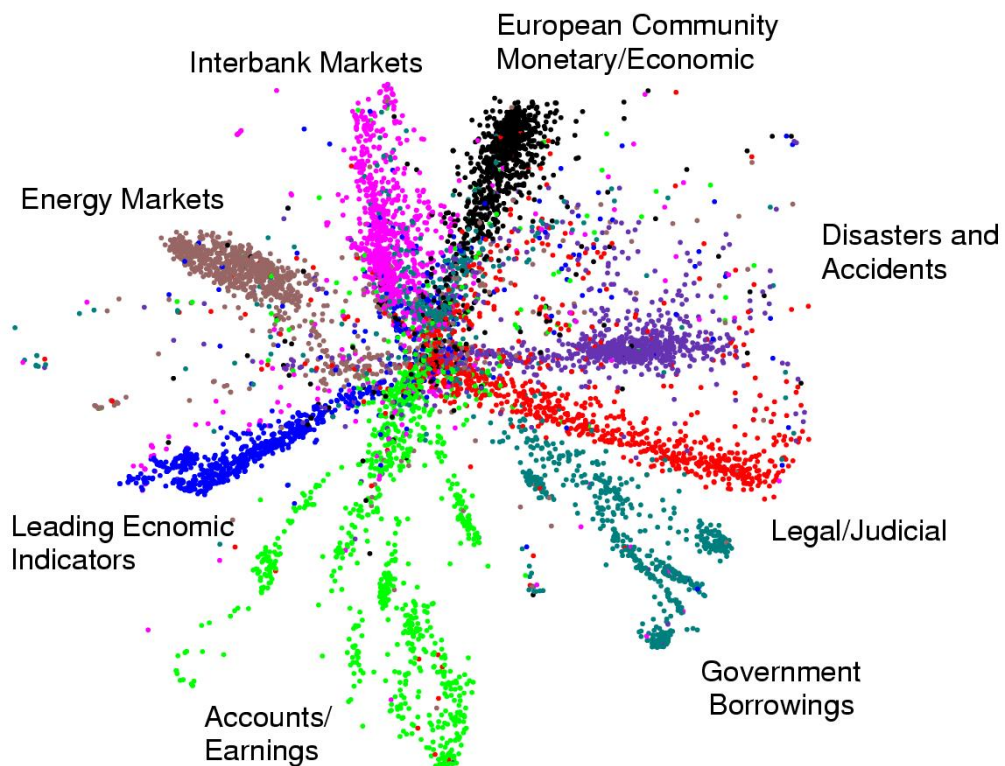


Each document is converted to a vector of word counts. This vector is then mapped to two coordinates and displayed as a colored dot. The colors represent the hand-labeled classes.

When the documents are laid out in 2-D, the classes are not used. So we can judge how good the algorithm is by seeing if the classes are separated.

Adopted from: <http://www.cs.toronto.edu/~hinton/> with figures from Chris Bishop, "Pattern Recognition and Machine Learning"

Displaying the Structure of a Set of Documents using a Deep Neural Network



Adopted from: <http://www.cs.toronto.edu/~hinton/> with figures from Chris Bishop, "Pattern Recognition and Machine Learning"

Machine Learning & Statistics

- A lot of machine learning is just a rediscovery of things that statisticians already knew. This is often disguised by differences in terminology:
 - Ridge regression = weight-decay
 - Fitting = learning
 - Held-out data = test data
- But the emphasis is very different:
 - A good piece of statistics: Clever proof that a relatively simple estimation procedure is asymptotically unbiased.
 - A good piece of machine learning: Demonstration that a complicated algorithm produces impressive results on a specific task.
- Data-mining: Using very simple machine learning techniques on very large databases because computers are too slow to do anything more interesting with ten billion examples.

Adopted from: <http://www.cs.toronto.edu/~hinton/> with figures from Chris Bishop, "Pattern Recognition and Machine Learning"

A Spectrum of Machine Learning Tasks

Statistics-----Artificial Intelligence

- Low-dimensional data (e.g. less than 100 dimensions)
- Lots of noise in the data
- There is not much structure in the data, and what structure there is, can be represented by a fairly simple model.
- The main problem is distinguishing true structure from noise.
- High-dimensional data (e.g. more than 100 dimensions)
- The noise is not sufficient to obscure the structure in the data if we process it right.
- There is a huge amount of structure in the data, but the structure is too complicated to be represented by a simple model.
- The main problem is figuring out a way to represent the complicated structure that allows it to be learned.

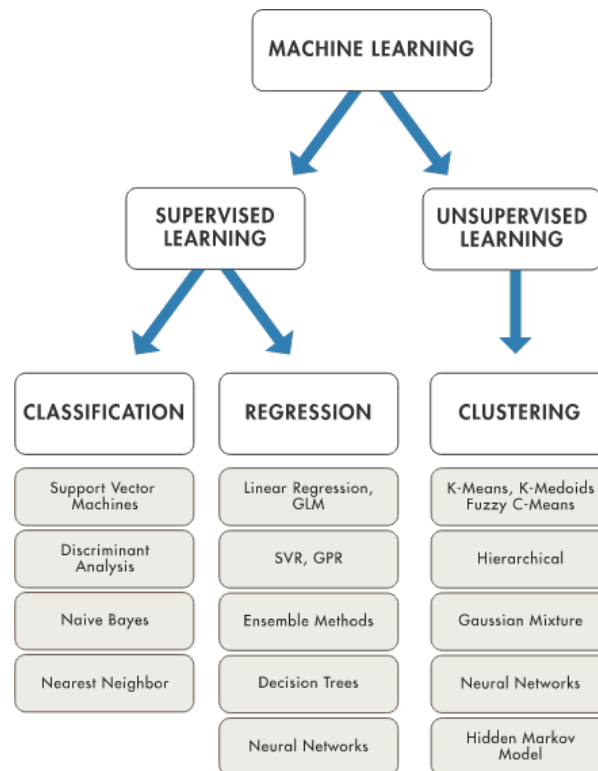
Adopted from: <http://www.cs.toronto.edu/~hinton/> with figures from Chris Bishop, "Pattern Recognition and Machine Learning"

Types of Learning Tasks

- Supervised learning
 - Learn to predict output when given an input vector
 - Who provides the correct answer?
- Reinforcement learning
 - Learn action to maximize payoff
 - Not much information in a payoff signal
 - Payoff is often delayed
 - Reinforcement learning is an important area that will not be covered in this course.
- Unsupervised learning
 - Create an internal representation of the input e.g. form clusters; extract features
 - How do we know if a representation is good?
 - This is the new frontier of machine learning because most big datasets do not come with labels.

Adopted from: <http://www.cs.toronto.edu/~hinton/> with figures from Chris Bishop, "Pattern Recognition and Machine Learning"

Types of Learning Tasks



Adopted from: <https://www.mathworks.com/discovery/machine-learning.html>

The Use of Data

- Divide the total dataset into three subsets:
 - **Training data** is used for learning the parameters of the model
 - **Validation data** is not used of learning but is used for deciding what type of model and what amount of regularization works best
 - **Test data** is used to get a final, unbiased estimate of how well the network works. We expect this estimate to be worse than on the validation data.

Adopted from: <http://www.cs.toronto.edu/~hinton/> with figures from Chris Bishop, "Pattern Recognition and Machine Learning"

Approach in this Class

- Use different machine learning algorithms
 - to solve some real-world problems
- Real-world applications of AI and ML
 - The future of Machine Learning and its Impact on Your Everyday Life
<https://www.youtube.com/watch?v=NjVIQUE8Sgk>
 - Deep Learning Applications | Deep Learning Applications In Real Life | Deep learning | Simplilearn
<https://www.youtube.com/watch?v=1LxmmF88fDw>

MCTE 512

Artificial Intelligence and Machine Learning for Engineers

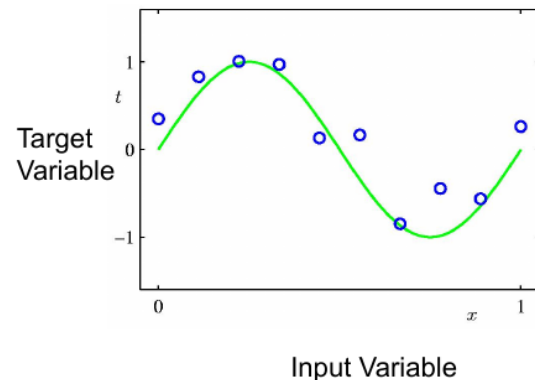
Lecture 2 – Polynomial Regression

Rodrigo S. Jamisola Jr.

Associate Professor in Robotics and Machine Learning
Simulations, Machine Learning, Robotics and Technopreneurship Laboratory (SMaRT Lab)
Dept. of Mechanical, Energy and Industrial Engineering
Faculty of Engineering and Technology
Botswana International University of Science and Technology

Synthetic Data for Regression

- Data gathered from function $\sin(2\pi x)$
 - Where x is the input
- Random noise in target values

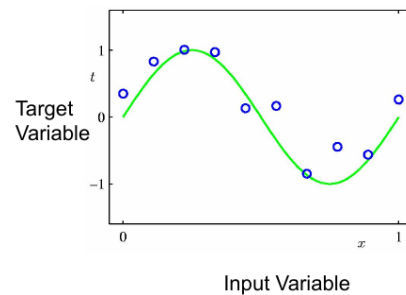


Input values $\{x_n\}$ generated uniformly in range (0,1). Corresponding target values $\{t_n\}$ obtained by first computing corresponding values of $\sin\{2\pi x\}$ then adding random noise with a Gaussian distribution with std dev 0.3

Adopted from: <https://cedar.buffalo.edu/~srihari/>

Training Set

- N observation of x
 - $\mathbf{x} = (x_1, \dots, x_N)^T$
 - $\mathbf{t} = (t_1, \dots, t_N)^T$
- Goal is to exploit training set to predict value t for some new value of x
- Inherently a difficult problem



Adopted from: <https://cedar.buffalo.edu/~srihari/>

A Simple Approach to Curve Fitting

- Fit the data using a polynomial function

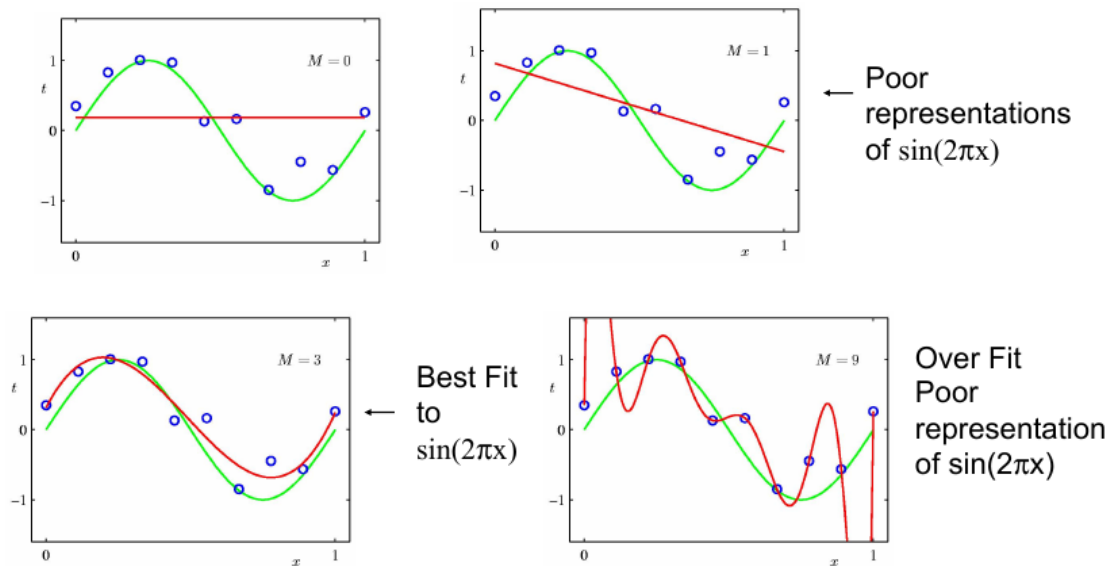
$$y(x, w) = w_0x^0 + w_1x^1 + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j$$

- where **M** is the order of the polynomial
- Is higher the value of M the better?
- Coefficient vector **w** = ($w_1, \dots, w_M$)
- It is a nonlinear function of **x**, but a linear function of the unknown parameters

Adopted from: <https://cedar.buffalo.edu/~srihari/>

Choosing the Order of M

- Model comparison or Model Selection
- Red lines are the best fits with $M=\{0,1,3,9\}$ and $N=10$



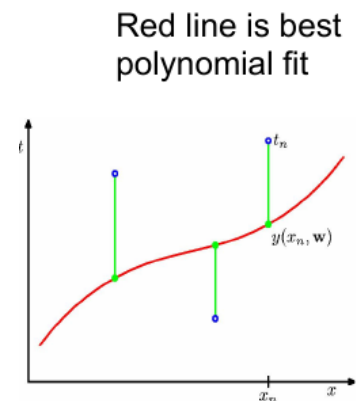
Adopted from: <https://cedar.buffalo.edu/~srihari/>

Error Function

- We can obtain a fit by minimizing an error function
 - sum of squares of the errors between the predictions $y(x_n, w)$ for each data points x_n and target value t_n

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N \{y(x_i, \mathbf{w}) - t_i\}^2$$

- factor $\frac{1}{2}$ included for a later convenience
- Solve by choosing value of w for which small as possible



Adopted from: <https://cedar.buffalo.edu/~srihari/>

To Solve for the Optimal Value of $\mathbf{w}$

- Chose the degree of the polynomial function M

$$y(x, w) = w_0x^0 + w_1x^1 + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j$$

- Choose of optimal values of $\mathbf{w}^* = (w_0^*, \dots, w_M^*)$ by minimizing the error function

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2$$

- Update the weights w such that

$$\mathbf{w} = \mathbf{w} - \eta \left[\frac{\partial E(\mathbf{w})}{\partial w_0}, \dots, \frac{\partial E(\mathbf{w})}{\partial w_M} \right]$$

- where

$$\frac{\partial E(\mathbf{w})}{\partial w_i} = \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\} x_n^i = \sum_{n=1}^N \left\{ \sum_{j=0}^M w_j x_n^j - t_n \right\} x_n^i$$

Adopted from: <https://cedar.buffalo.edu/~srihari/>

Example: Analytical vs. Numerical

x_i	t_i
0.0	20
0.1	30
0.2	50
0.3	60
0.4	70
0.5	20

Find $\mathbf{w} = [w_0, w_1, w_2]$

such that $y(\mathbf{w}, x_i) = w_0x_i^0 + w_1x_i^1 + w_2x_i^2$

Example: Analytical Method

Find $\mathbf{w} = [w_0, w_1, w_2]$

such that $y(\mathbf{w}, x_i) = w_0x_i^0 + w_1x_i^1 + w_2x_i^2$

x_i	t_i
0.0	20
0.1	30
0.2	50
0.3	60
0.4	70
0.5	20

$$\begin{bmatrix} 20 \\ 30 \\ 50 \\ 60 \\ 70 \\ 20 \end{bmatrix} = \begin{bmatrix} 1 & 0.0 & 0.0^2 \\ 1 & 0.1 & 0.1^2 \\ 1 & 0.2 & 0.2^2 \\ 1 & 0.3 & 0.3^2 \\ 1 & 0.4 & 0.4^2 \\ 1 & 0.5 & 0.5^2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

$$\mathbf{w} = \mathbf{X}^+ \mathbf{t}$$

Example: Numerical Method

x_i	t_i
0.0	20
0.1	30
0.2	50
0.3	60
0.4	70
0.5	20

Find $\mathbf{w} = [w_0, w_1, w_2]$

such that $y(\mathbf{w}, x_i) = w_0x_i^0 + w_1x_i^1 + w_2x_i^2$

1. Initialize $\mathbf{w}$
2. Compute gradient of Error

$$\frac{\partial E(\mathbf{w})}{\partial w_i} = \sum_{n=1}^N \left\{ \sum_{j=0}^M w_j x_n^j - t_n \right\} x_n^i$$

3. Take a step to the next approximated value of $\mathbf{w}$

$$\mathbf{w} = \mathbf{w} - \eta \left[\frac{\partial E(\mathbf{w})}{\partial w_0}, \dots, \frac{\partial E(\mathbf{w})}{\partial w_M} \right]$$

4. Repeat the steps until error term is minimized

x_i	t_i
0.0	20
0.1	30
0.2	50
0.3	60
0.4	70
0.5	20

Example: Numerical Method

1. Initialize $\mathbf{w}$ $w_1 = w_2 = w_3 = 1$

2. Compute gradient of Error

$$\frac{\partial E(\mathbf{w})}{\partial w_0} = \begin{Bmatrix} 1 * 0.0^0 + 1 * 0.0^1 + 1 * 0.0^2 - 20 \\ 1 * 0.1^0 + 1 * 0.1^1 + 1 * 0.1^2 - 30 \\ 1 * 0.2^0 + 1 * 0.2^1 + 1 * 0.2^2 - 40 \\ 1 * 0.3^0 + 1 * 0.3^1 + 1 * 0.3^2 - 60 \\ 1 * 0.4^0 + 1 * 0.4^1 + 1 * 0.4^2 - 70 \\ 1 * 0.5^0 + 1 * 0.5^1 + 1 * 0.5^2 - 20 \end{Bmatrix} \begin{matrix} 0.0^0 + \\ 0.1^0 + \\ 0.2^0 + \\ 0.3^0 + \\ 0.4^0 + \\ 0.5^0 \end{matrix}$$

$$\frac{\partial E(\mathbf{w})}{\partial w_1} = \begin{Bmatrix} 1 * 0.0^0 + 1 * 0.0^1 + 1 * 0.0^2 - 20 \\ 1 * 0.1^0 + 1 * 0.1^1 + 1 * 0.1^2 - 30 \\ 1 * 0.2^0 + 1 * 0.2^1 + 1 * 0.2^2 - 40 \\ 1 * 0.3^0 + 1 * 0.3^1 + 1 * 0.3^2 - 60 \\ 1 * 0.4^0 + 1 * 0.4^1 + 1 * 0.4^2 - 70 \\ 1 * 0.5^0 + 1 * 0.5^1 + 1 * 0.5^2 - 20 \end{Bmatrix} \begin{matrix} 0.0^1 + \\ 0.1^1 + \\ 0.2^1 + \\ 0.3^1 + \\ 0.4^1 + \\ 0.5^1 \end{matrix}$$

x_i	t_i
0.0	20
0.1	30
0.2	50
0.3	60
0.4	70
0.5	20

Example: Numerical Method

$$\frac{\partial E(\mathbf{w})}{\partial w_2} = \begin{aligned} &\{1 * 0.0^0 + 1 * 0.0^1 + 1 * 0.0^2 - 20\} 0.0^2 + \\ &\{1 * 0.1^0 + 1 * 0.1^1 + 1 * 0.1^2 - 30\} 0.1^2 + \\ &\{1 * 0.2^0 + 1 * 0.2^1 + 1 * 0.2^2 - 40\} 0.2^2 + \\ &\{1 * 0.3^0 + 1 * 0.3^1 + 1 * 0.3^2 - 60\} 0.3^2 + \\ &\{1 * 0.4^0 + 1 * 0.4^1 + 1 * 0.4^2 - 70\} 0.4^2 + \\ &\{1 * 0.5^0 + 1 * 0.5^1 + 1 * 0.5^2 - 20\} 0.5^2 \end{aligned}$$

3. Update $\mathbf{w}$

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial E(\mathbf{w})}{\partial w_0} \\ \frac{\partial E(\mathbf{w})}{\partial w_1} \\ \frac{\partial E(\mathbf{w})}{\partial w_2} \end{bmatrix}$$

4. Repeat the process 2 and 3 until error is minimized

MCTE 512

Artificial Intelligence and Machine Learning for Engineers

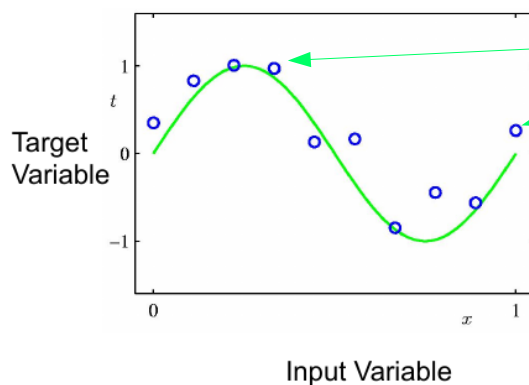
Lecture 3 – Decision Trees

Rodrigo S. Jamisola Jr.

Associate Professor in Robotics and Machine Learning
Simulations, Machine Learning, Robotics and Technopreneurship Laboratory (SMaRT Lab)
Dept. of Mechanical, Energy and Industrial Engineering
Faculty of Engineering and Technology
Botswana International University of Science and Technology

Review on Polynomial Regression

- Given data
 - We want to estimate a function $y = f(x)$ that best approximates the data



Data = $\{\mathbf{x}, \mathbf{t}\}$

$$y = f(\mathbf{x})$$

$$y = w_0x^0 + w_1x^1 + w_2x^2 + \dots + w_Mx^M$$

$$E = \|\mathbf{y} - \mathbf{t}\|$$

We need to find the function f

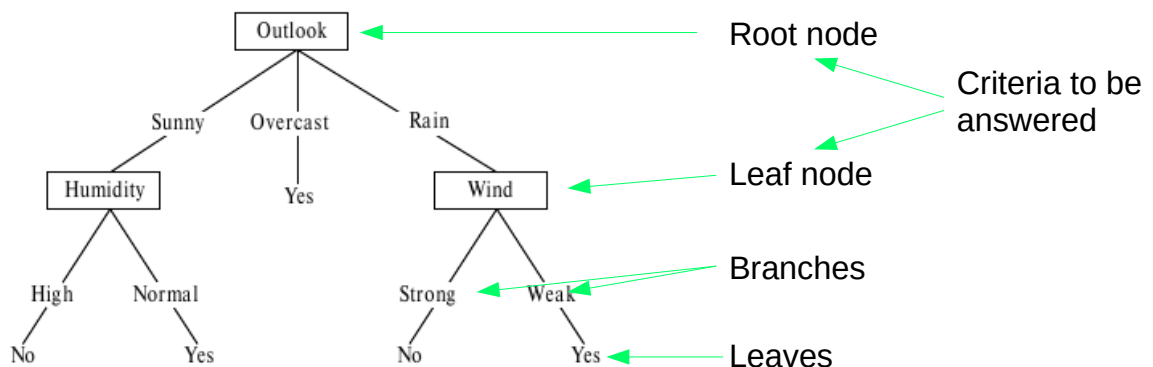
- that best approximates
- the relationship of input vs. output
- from the given data

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

What is a Decision Tree?

- A decision tree is a support tool that
 - Uses a tree-like graph or model of decisions
 - And their possible consequences
 - Include chance event of outcomes, resource costs, and utility
- Example: A decision tree to *PlayTennis*

$$Decision = f(Inputs)$$

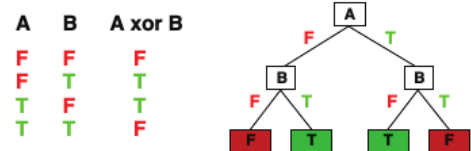


$$Decision = f(Outlook, Humidity, Wind)$$

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

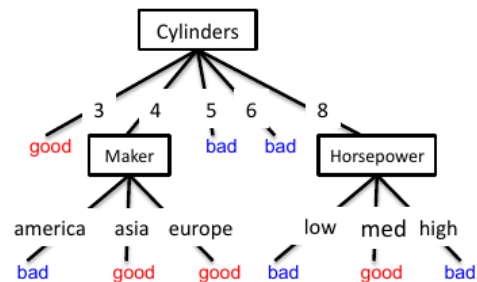
What Function can be Represented?

- Decision trees can represent any function of the input attributes!
- For Boolean functions, path to leaf gives truth table row
- But it could require exponentially many nodes



(Figure from Stuart Russell)

$\text{cyl}=3 \vee (\text{cyl}=4 \wedge (\text{maker}=\text{asia} \vee \text{maker}=\text{europe})) \vee \dots$

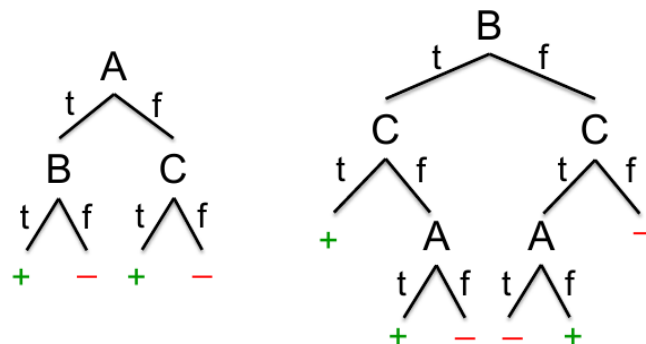


Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

Are all Decision Trees Equal?

- Many trees can represent the same concept
- But, not all trees will have the same size!

- e.g., $\phi = (A \wedge B) \vee (\neg A \wedge C) \rightarrow ((A \text{ and } B) \text{ or } (\text{not } A \text{ and } C))$



- Which tree do we prefer?

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

Videos of Decision Trees of Boolean Function

Boolean AND, OR, and XOR

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

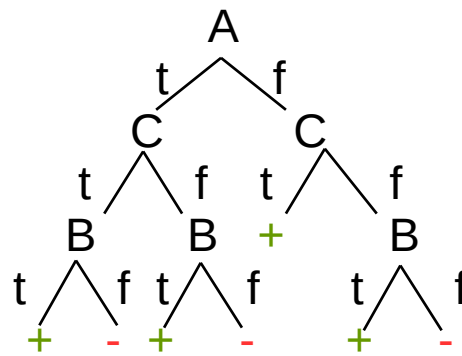
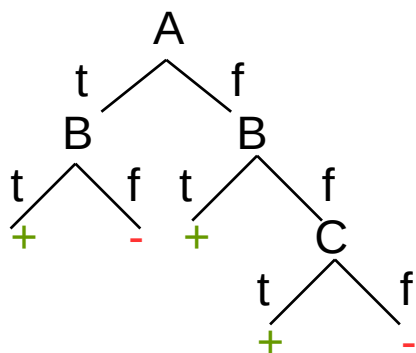
Example: Boolean Function Decision Trees

$$\phi = (\neg A \wedge C) \vee (B \wedge \neg C) \vee (A \wedge B)$$

A	B	C	$\neg A$	$\neg C$	$\neg A \wedge C$	$B \wedge \neg C$	$A \wedge B$	ϕ
T	T	T	F	F	F	F	T	T
T	T	F	F	T	F	T	T	T
T	F	T	F	F	F	F	F	F
T	F	F	F	T	F	F	F	F
F	T	T	T	F	T	F	F	T
F	T	F	T	T	F	T	F	T
F	F	T	T	F	T	F	F	T
F	F	F	T	T	F	F	F	F

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

Example: Boolean Function Decision Trees



Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

Example: Predict Fuel Efficiency

- 40 data points
- Predict: Miles per gallon (MPG)
- Find $Y = f(X)$
- Discrete data

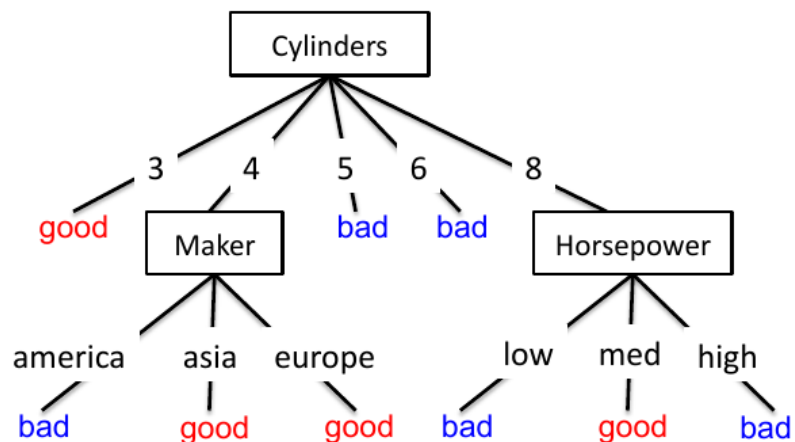
mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europa
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europa
bad	5	medium	medium	medium	medium	75to78	europa

Y
 X

Adopted from: UCI repository by Ross Quinlan

Hypothesis: Decision Trees $f: X \rightarrow Y$

- Each internal node tests an attribute x_i
- Each branch assigns an attribute value $x_i = v$
- Each leaf assigns a class y
- To classify input x , transverse the tree from root to leaf, and output the labeled y



Adopted from: UCI repository by Ross Quinlan

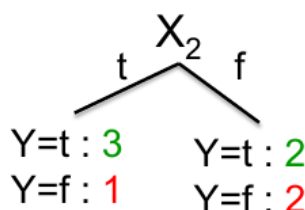
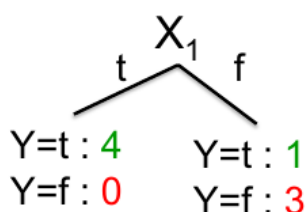
Learning Decision Trees is Hard!

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest 76]
 - NP (nondeterministic polynomial) problem
 - its solution can be guessed and verified in polynomial time
 - it can be solved by a restricted class of brute force search algorithms
 - nondeterministic means
 - no particular rule is followed to make the guess
- Resort to greedy heuristic
 - Start from empty decision tree
 - Split on next best attribute (feature)
 - Recurse

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

Splitting: Choosing a Good Attribute

- Would we prefer to split on X_1 or X_2 ?



Use counts at leaves to define probability distributions, so we can measure uncertainty

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

Coin Flip Data

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

Measuring Uncertainty

- Good split if we are more certain about classification after split
 - Deterministic good (all TRUE or all FALSE)
 - Uniform distribution – BAD
 - What about distributions in between?

$P(Y=A) = 1/2$	$P(Y=B) = 1/4$	$P(Y=C) = 1/8$	$P(Y=D) = 1/8$
----------------	----------------	----------------	----------------

$P(Y=A) = 1/4$	$P(Y=B) = 1/4$	$P(Y=C) = 1/4$	$P(Y=D) = 1/4$
----------------	----------------	----------------	----------------

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

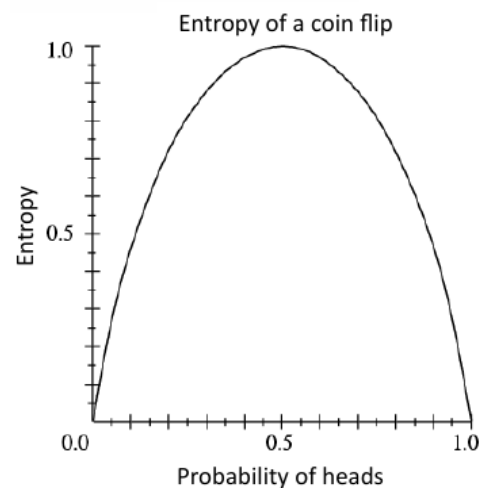
Entropy

- Entropy $H(Y)$ of a random variable Y

$$H(Y) = - \sum_{i=1}^k P(Y = y_i) \log_2 P(Y = y_i)$$

More uncertainty, more entropy!

Information Theory interpretation: $H(Y)$ is the expected number of bits needed to encode a randomly drawn value of Y (under most efficient code)



Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

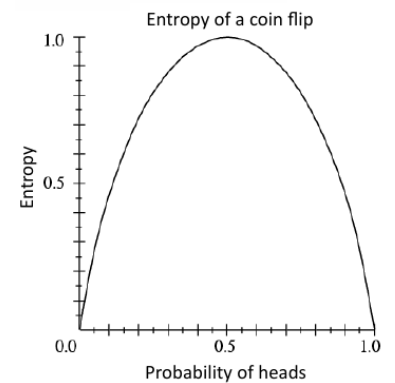
High, Low Entropy

- “High Entropy”
 - Y is from uniform-like distribution
 - Flat histogram
 - Values sampled from are less predictable
- “Low Entropy”
 - Y is from varied (peaks and valleys) distribution
 - Histogram has many lows and highs
 - Values sampled from are more predictable

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

Entropy Example

$$H(Y) = - \sum_{i=1}^k P(Y = y_i) \log_2 P(Y = y_i)$$



$$P(Y=t) = 5/6$$

$$P(Y=f) = 1/6$$

$$H(Y) = - 5/6 \log_2 5/6 - 1/6 \log_2 1/6$$

$$= 0.65$$

X ₁	X ₂	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

Conditional Entropy

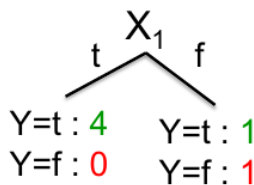
- Conditional Entropy $H(Y|X)$ of a random variable Y conditioned on random variable X

$$H(Y|X) = - \sum_{j=1}^v P(X = x_j) \sum_{i=1}^k P(Y = y_i | X = x_j) \log_2 P(Y = y_i | X = x_j)$$

Example:

$$P(X_1=t) = 4/6$$

$$P(X_1=f) = 2/6$$



$$\begin{aligned} H(Y|X_1) &= - 4/6 (1 \log_2 1 + 0 \log_2 0) \\ &\quad - 2/6 (1/2 \log_2 1/2 + 1/2 \log_2 1/2) \\ &= 2/6 \end{aligned}$$

X ₁	X ₂	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

Information Gain

- Decrease in entropy (uncertainty) after splitting

$$IG(X_a) = H(Y) - H(Y|X_a)$$

attribute *a*
all outputs
all outputs given attribute *a*

In our running example:

$$\begin{aligned}
 IG(X_1) &= H(Y) - H(Y|X_1) \\
 &= 0.65 - 0.33
 \end{aligned}$$

$IG(X_1) > 0 \rightarrow$ we prefer the split!

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

Example: PlayTennis

Outlook	Temperature	Humidity	Windy	Play?
overcast	cool	normal	true	yes
overcast	hot	high	false	yes
overcast	hot	normal	false	yes
overcast	mild	high	true	yes
rainy	cool	normal	true	no
rainy	mild	high	true	no
rainy	cool	normal	false	yes
rainy	mild	high	false	yes
rainy	mild	normal	false	yes
sunny	hot	high	false	no
sunny	hot	high	true	no
sunny	mild	high	false	no
sunny	cool	normal	false	yes
sunny	mild	normal	true	yes

Adopted from: <https://datascienceinsider.in/2015/10/19/decision-tree-explained/>

Example: PlayTennis

$H(Y)$ Before: 14 records, 9 are "yes"

$$-\left(\frac{9}{14} \log_2 \frac{9}{14} + \frac{5}{14} \log_2 \frac{5}{14}\right) = 0.94$$

$H(Y|X_{\text{outlook}})$

If we choose **outlook**:

overcast : 4 records, 4 are "yes"

$$-\left(\frac{4}{4} \log_2 \frac{4}{4}\right) = 0$$

rainy : 5 records, 3 are "yes"

$$-\left(\frac{3}{5} \log_2 \frac{3}{5} + \frac{2}{5} \log_2 \frac{2}{5}\right) = 0.97$$

sunny : 5 records, 2 are "yes"

$$-\left(\frac{2}{5} \log_2 \frac{2}{5} + \frac{3}{5} \log_2 \frac{3}{5}\right) = 0.97$$

Expected new entropy:

$$\frac{4}{14} \times 0.0 + \frac{5}{14} \times 0.97 + \frac{5}{14} \times 0.97$$

$$= \underline{0.69}$$

Outlook	Temperature	Humidity	Windy	Play?
overcast	cool	normal	true	yes
overcast	hot	high	false	yes
overcast	hot	normal	false	yes
overcast	mild	high	true	yes
rainy	cool	normal	true	no
rainy	mild	high	true	no
rainy	cool	normal	false	yes
rainy	mild	high	false	yes
rainy	mild	normal	false	yes
sunny	hot	high	false	no
sunny	hot	high	true	no
sunny	mild	high	false	no
sunny	cool	normal	false	yes
sunny	mild	normal	true	yes

$$IG(X_a) = H(Y) - H(Y|X_a)$$

$$H(Y) = - \sum_{i=1}^k P(Y = y_i) \log_2 P(Y = y_i)$$

$$H(Y|X) = - \sum_{j=1}^v P(X = x_j) \sum_{i=1}^k P(Y = y_i | X = x_j) \log_2 P(Y = y_i | X = x_j)$$

Adopted from: <https://datascienceinsider.in/2015/10/19/decision-tree-explained/>

Example: PlayTennis

$H(Y)$ Before: 14 records, 9 are "yes"

$$-\left(\frac{9}{14} \log_2 \frac{9}{14} + \frac{5}{14} \log_2 \frac{5}{14}\right) = 0.94$$

$H(Y|X_{\text{temperature}})$

If we choose **temperature**:

cool : 4 records, 3 are "yes"

0.81

hot : 4 records, 2 are "yes"

1.0

mild : 6 records, 4 are "yes"

0.92

Expected new entropy:

$$0.81(4/14) + 1.0(4/14) + 0.92(6/14) \\ = \underline{0.91}$$

Outlook	Temperature	Humidity	Windy	Play?
overcast	cool	normal	true	yes
overcast	hot	high	false	yes
overcast	hot	normal	false	yes
overcast	mild	high	true	yes
rainy	cool	normal	true	no
rainy	mild	high	true	no
rainy	cool	normal	false	yes
rainy	mild	high	false	yes
rainy	mild	normal	false	yes
sunny	hot	high	false	no
sunny	hot	high	true	no
sunny	mild	high	false	no
sunny	cool	normal	false	yes
sunny	mild	normal	true	yes

$$IG(X_a) = H(Y) - H(Y|X_a)$$

$$H(Y) = - \sum_{i=1}^k P(Y = y_i) \log_2 P(Y = y_i)$$

$$H(Y|X) = - \sum_{j=1}^v P(X = x_j) \sum_{i=1}^k P(Y = y_i | X = x_j) \log_2 P(Y = y_i | X = x_j)$$

Adopted from: <https://datascienceinsider.in/2015/10/19/decision-tree-explained/>

Example: PlayTennis

Before: 14 records, 9 are "yes"

$H(Y)$

$$-\left(\frac{9}{14} \log_2 \frac{9}{14} + \frac{5}{14} \log_2 \frac{5}{14}\right) = 0.94$$

$H(Y|X_{outlook})$

outlook

$$0.94 - 0.69 = 0.25 \quad \text{highest gain}$$

$H(Y|X_{temperature})$

temperature

$$0.94 - 0.91 = 0.03$$

$H(Y|X_{humidity})$

humidity

$$0.94 - 0.725 = 0.215$$

$H(Y|X_{windy})$

windy

$$0.94 - 0.87 = 0.07$$

Outlook	Temperature	Humidity	Windy	Play?
overcast	cool	normal	true	yes
overcast	hot	high	false	yes
overcast	hot	normal	false	yes
overcast	mild	high	true	yes
rainy	cool	normal	true	no
rainy	mild	high	true	no
rainy	cool	normal	false	yes
rainy	mild	high	false	yes
rainy	mild	normal	false	yes
sunny	hot	high	false	no
sunny	hot	high	true	no
sunny	mild	high	false	no
sunny	cool	normal	false	yes
sunny	mild	normal	true	yes

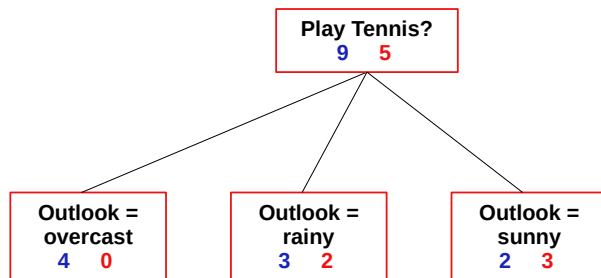
$$IG(X_a) = H(Y) - H(Y|X_a)$$

$$H(Y) = - \sum_{i=1}^k P(Y = y_i) \log_2 P(Y = y_i)$$

$$H(Y|X) = - \sum_{j=1}^v P(X = x_j) \sum_{i=1}^k P(Y = y_i | X = x_j) \log_2 P(Y = y_i | X = x_j)$$

Adopted from: <https://datascienceinsider.in/2015/10/19/decision-tree-explained/>

Example: PlayTennis



For Outlook = rainy $-\left(\frac{3}{5} \log_2 \frac{3}{5} + \frac{2}{5} \log_2 \frac{2}{5}\right)$

$H(Y|X_temperature):$

cool: 2 records, 1 "yes" $-\frac{2}{5}^*$

$$\left(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}\right) +$$

mild: 3 records, 2 "yes" $-\frac{3}{5}^*$

$$\left(\frac{2}{3} \log_2 \frac{2}{3} + \frac{1}{3} \log_2 \frac{1}{3}\right)$$

$H(Y|X_humidity):$

normal: 3 records, 2 "yes" $-\frac{3}{5}^*$

$$\left(\frac{2}{3} \log_2 \frac{2}{3} + \frac{1}{3} \log_2 \frac{1}{3}\right) +$$

high: 2 records, 1 "yes" $-\frac{2}{5}^*$

$$\left(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}\right)$$

$H(Y|X_windy):$

true: 2 records, 0 "yes" $-\frac{2}{5}^*$

$$\left(\frac{2}{2} \log_2 \frac{2}{2}\right)$$

false: 3 records, 3 "yes" $-\frac{3}{5}^*$

$$\left(\frac{3}{3} \log_2 \frac{3}{3}\right)$$

Outlook	Temperature	Humidity	Windy	Play?
overcast	cool	normal	true	yes
overcast	hot	high	false	yes
overcast	hot	normal	false	yes
overcast	mild	high	true	yes
rainy	cool	normal	true	no
rainy	mild	high	true	no
rainy	cool	normal	false	yes
rainy	mild	high	false	yes
rainy	mild	normal	false	yes
sunny	hot	high	false	no
sunny	hot	high	true	no
sunny	mild	high	false	no
sunny	cool	normal	false	yes
sunny	mild	normal	true	yes

$$IG(X_a) = H(Y) - H(Y|X_a)$$

$$H(Y) = - \sum_{i=1}^k P(Y = y_i) \log_2 P(Y = y_i)$$

$$H(Y|X) = - \sum_{j=1}^v P(X = x_j) \sum_{i=1}^k P(Y = y_i | X = x_j) \log_2 P(Y = y_i | X = x_j)$$

Adopted from: <https://datascienceinsider.in/2015/10/19/decision-tree-explained/>

Example: Predict Fuel Efficiency

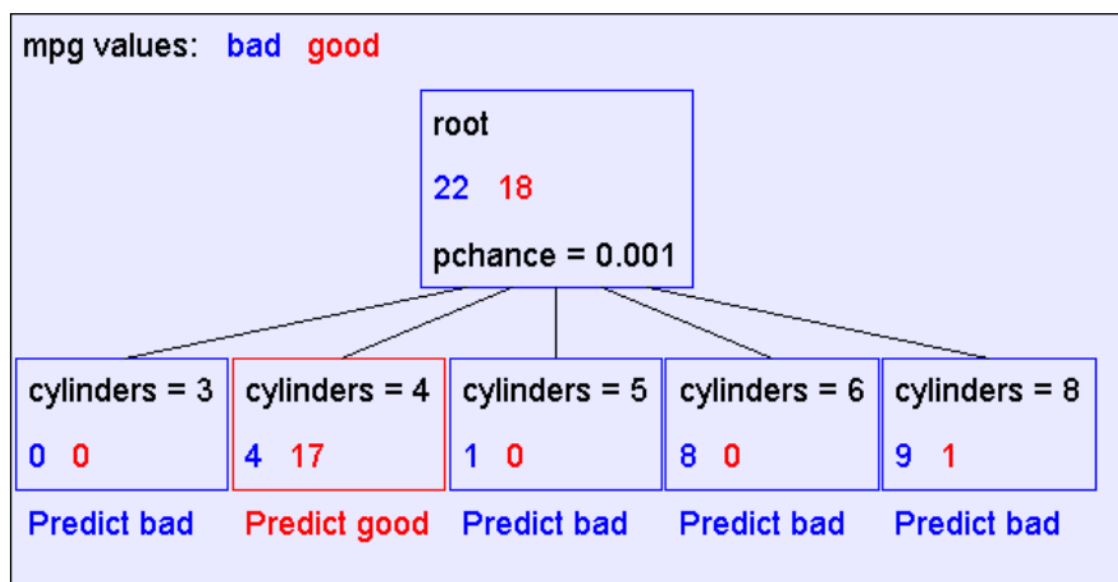
- 40 data points
- Predict: Miles per gallon (MPG)
- Find $Y = f(X)$
- Discrete data

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europa
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europa
bad	5	medium	medium	medium	medium	75to78	europa

Y
X

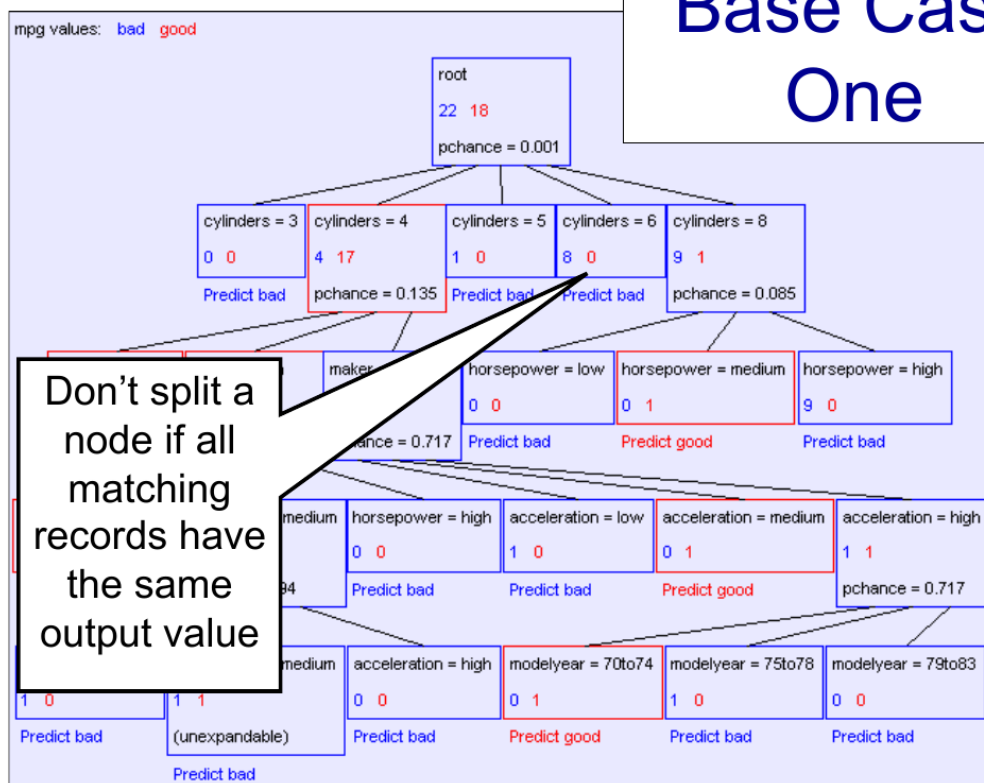
Adopted from: UCI repository by Ross Quinlan

A Decision Stump

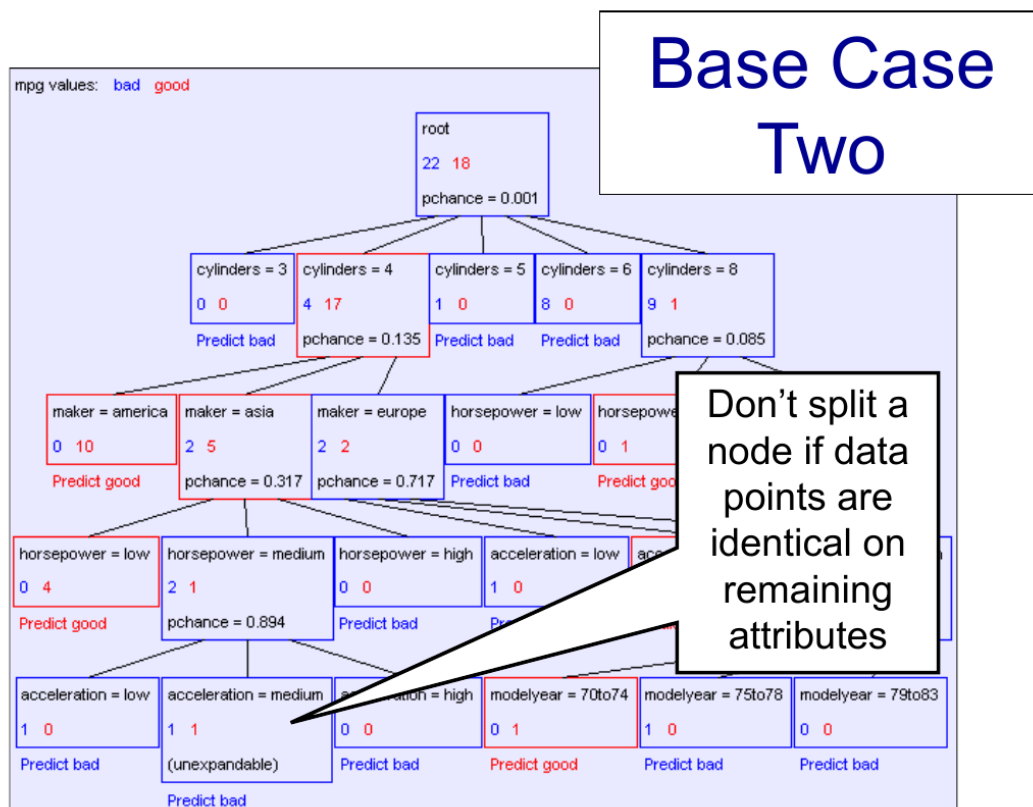


First split looks good! But, when do we stop?

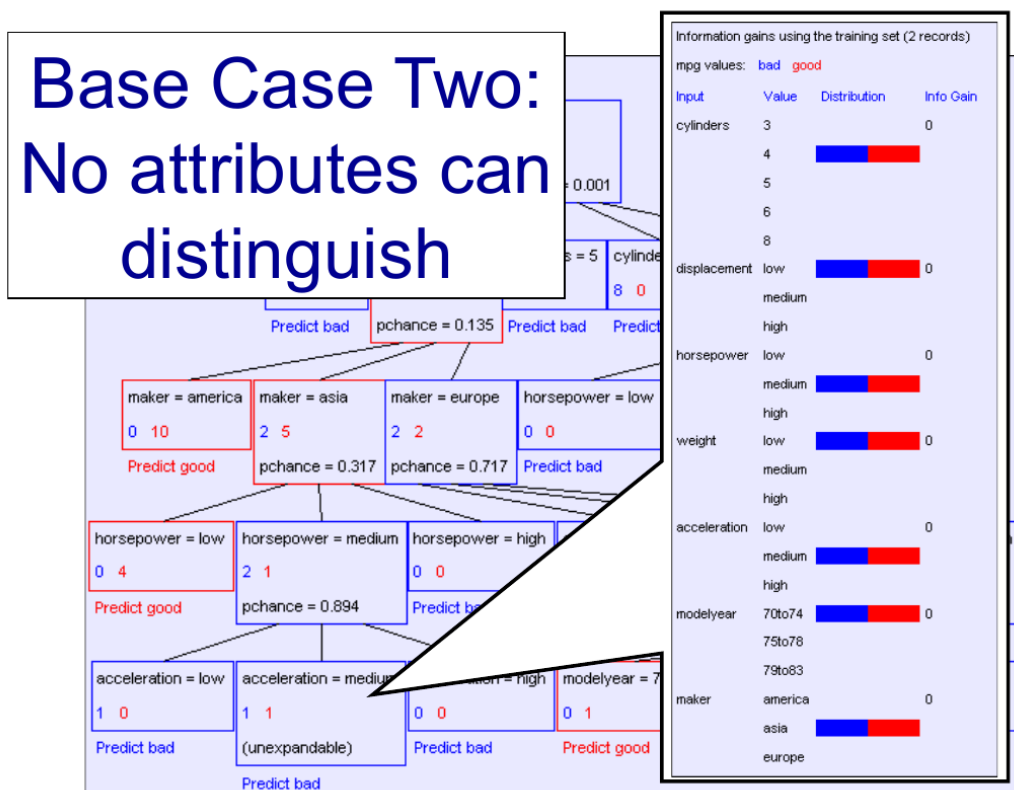
Adopted from: UCI repository by Ross Quinlan



26 of 31

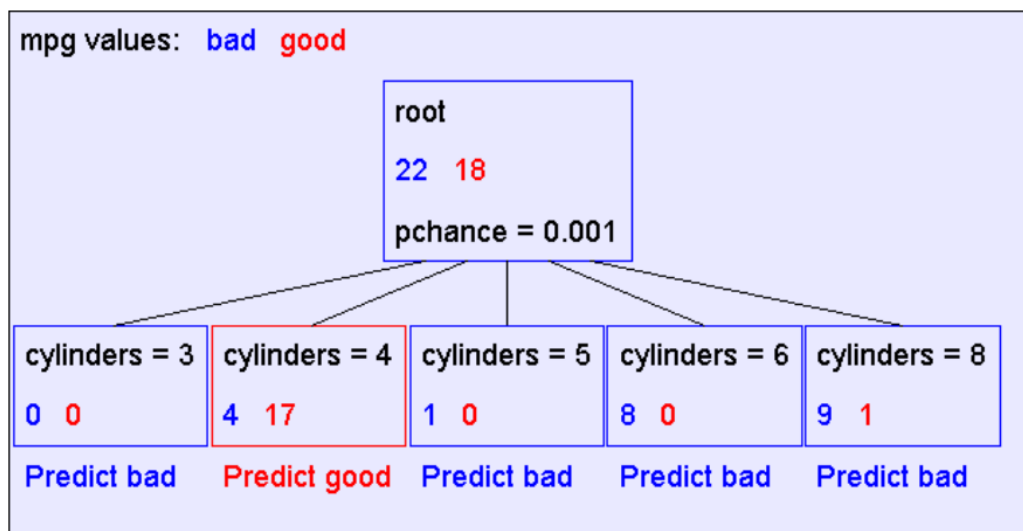


Adopted from: UCI repository by Ross Quinlan



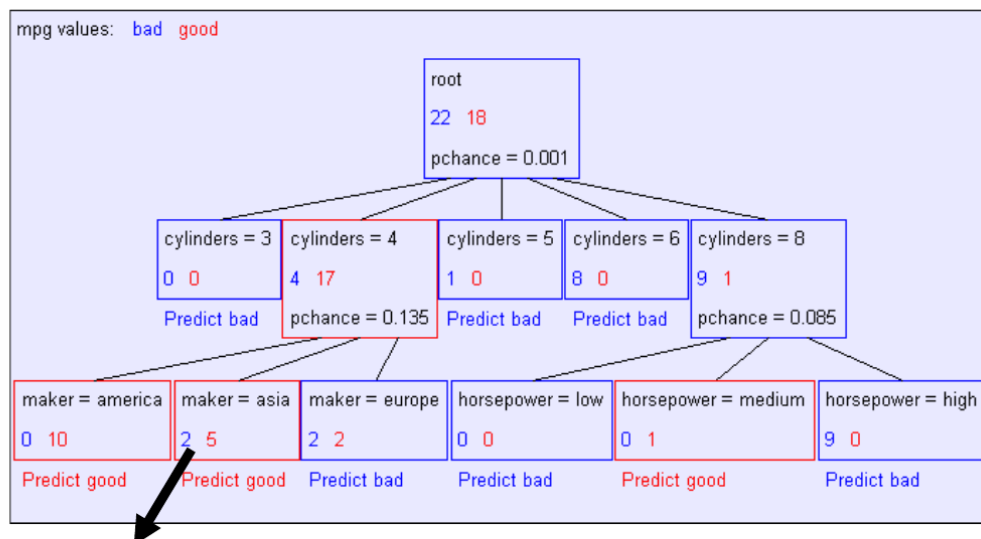
Adopted from: UCI repository by Ross Quinlan

A Decision Stump



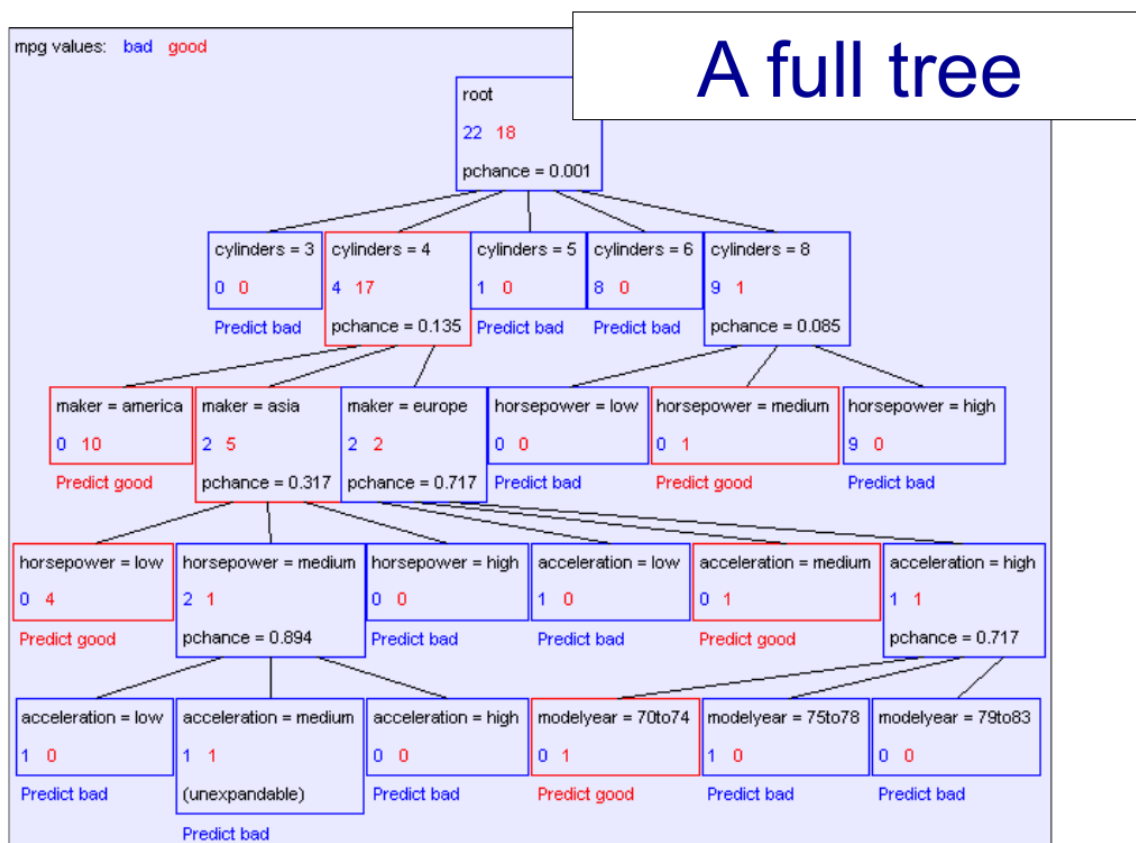
Adopted from: UCI repository by Ross Quinlan

Second level of tree



Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)



MCTE 512

Artificial Intelligence and Machine Learning for Engineers

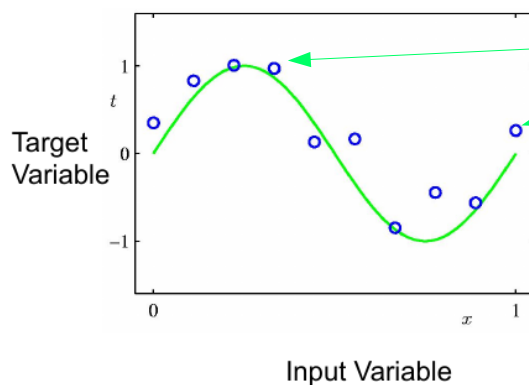
Lecture 4 – Bayes Rule

Rodrigo S. Jamisola Jr.

Associate Professor in Robotics and Machine Learning
Simulations, Machine Learning, Robotics and Technopreneurship Laboratory (SMaRT Lab)
Dept. of Mechanical, Energy and Industrial Engineering
Faculty of Engineering and Technology
Botswana International University of Science and Technology

Review on Polynomial Regression

- Given data
 - We want to estimate a function $y = f(x)$ that best approximates the data



Data = $\{\mathbf{x}, \mathbf{t}\}$

$$y = f(\mathbf{x})$$

$$y = w_0x^0 + w_1x^1 + w_2x^2 + \dots + w_Mx^M$$

$$E = \|\mathbf{y} - \mathbf{t}\|$$

We need to find the function f

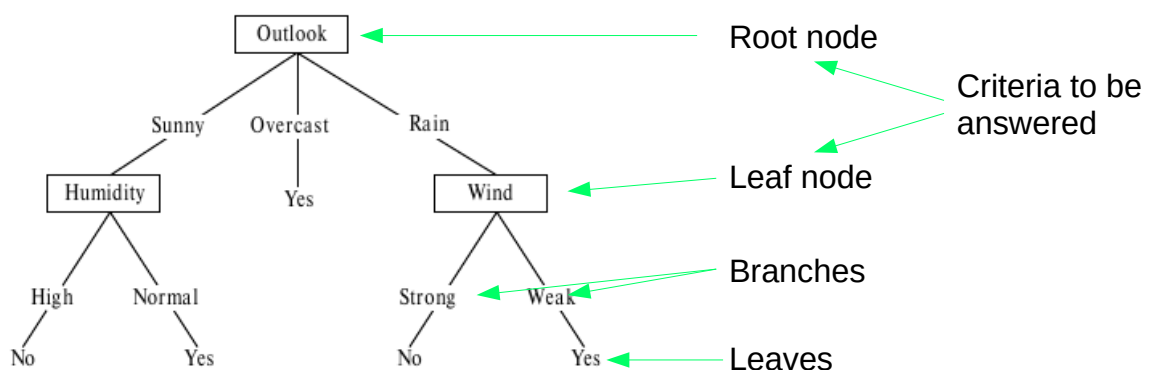
- that best approximates
- the relationship of input vs. output
- from the given data

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

Review on Decision Trees

- A decision tree is a support tool that
 - Uses a tree-like graph or model of decisions
 - And their possible consequences
 - Include chance event of outcomes, resource costs, and utility
- Example: A decision tree to *PlayTennis*

$$Decision = f(Inputs)$$



$$Decision = f(Outlook, Humidity, Wind)$$

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

Bayes Rule

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Which is shorthand for:

$$(\forall i, j) P(Y = y_i | X = x_j) = \frac{P(X = x_j | Y = y_i) P(Y = y_i)}{P(X = x_j)}$$

Equivalently:

$$(\forall i, j) P(Y = y_i | X = x_j) = \frac{P(X = x_j | Y = y_i) P(Y = y_i)}{\sum_k P(X = x_j | Y = y_k) P(Y = y_k)}$$

Adopted from: Tom M. Mitchell

Example of Bayes Rule

- Three jars contain colored balls as described in the table below.
 - One jar is chosen at random and a ball is selected. If the ball is red, what is the probability that it came from the 2nd jar?

Jar #	Red	White	Blue
1	3	4	1
2	1	2	3
3	4	3	2

Adopted from: somewhere I cannot trace the source

Example of Bayes Rule

- We will define the following events:
 - J_1 is the event that *first* jar is chosen
 - J_2 is the event that *second* jar is chosen
 - J_3 is the event that *third* jar is chosen
 - R is the event that a *red* ball is selected

Adopted from: somewhere I cannot trace the source

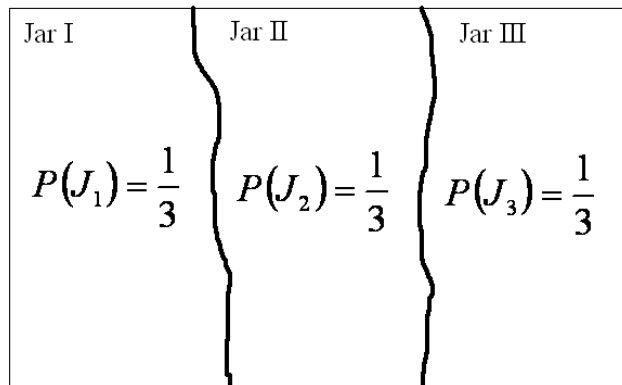
Example of Bayes Rule

- The events J_1 , J_2 , and J_3 mutually exclusive
("Mutually exclusive" is a statistical term describing two or more events that cannot occur simultaneously.)
 - Why?
 - *You can't chose two different jars at the same time*
- Because of this, our sample space has been divided or *partitioned* along these three events

Adopted from: somewhere I cannot trace the source

Venn Diagram

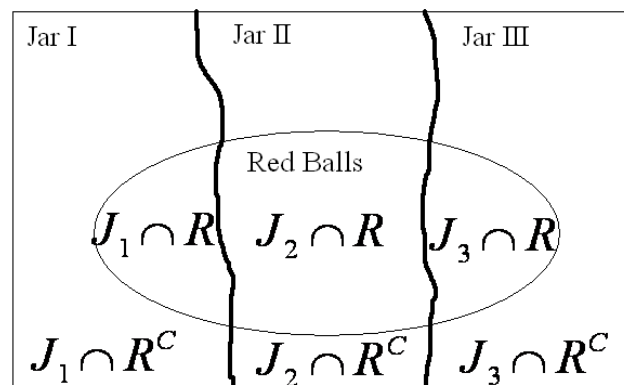
- Let's look at the Venn Diagram



Adopted from: somewhere I cannot trace the source

Venn Diagram

- All of the red balls are in the first, second, and third jar so their set overlaps all three sets of our partition



Adopted from: somewhere I cannot trace the source

Finding Probabilities

- What are the probabilities for each of the events in our sample space?
- How do we find them?

$$P(A \cap B) = P(A|B) P(B)$$

Adopted from: somewhere I cannot trace the source

Computing Probabilities

$$P(J_1 \cap R) = P(R|J_1)P(J_1) = \frac{3}{8} \cdot \frac{1}{3} = \frac{1}{8}$$

- Similar calculations show:

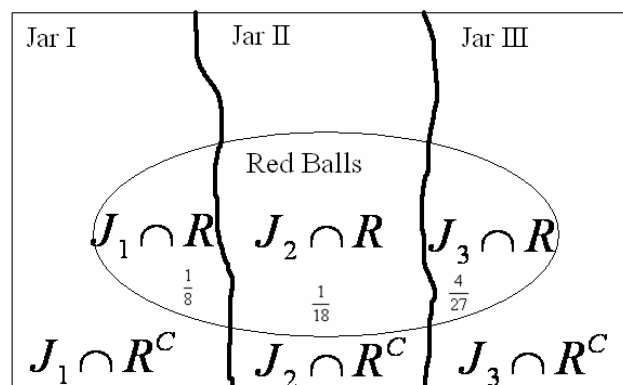
$$P(J_2 \cap R) = P(R|J_2)P(J_2) = \frac{1}{6} \cdot \frac{1}{3} = \frac{1}{18}$$

$$P(J_3 \cap R) = P(R|J_3)P(J_3) = \frac{4}{9} \cdot \frac{1}{3} = \frac{4}{27}$$

Adopted from: somewhere I cannot trace the source

Venn Diagram

- Updating our Venn Diagram with these probabilities:



Adopted from: somewhere I cannot trace the source

Taking the Other Probabilities

- Our original problem was:
 - One jar is chosen at random and a ball is selected. If the ball is red, what is the probability that it came from the 2nd jar?
- In terms of the events we've defined we want:

$$P(J_2 | R) = \frac{P(J_2 \cap R)}{P(R)}$$

Adopted from: somewhere I cannot trace the source

Finding our Probabilities

- We already know what the numerator portion is from our Venn Diagram
- What is the denominator portion?

$$P(J_2 | R) = \frac{P(J_2 \cap R)}{P(R)}$$
$$\frac{P(J_2 \cap R)}{P(J_1 \cap R) + P(J_2 \cap R) + P(J_3 \cap R)}$$

Adopted from: somewhere I cannot trace the source

Computing the Values

- Plugging in the appropriate values:

$$P(J_2|R) = \frac{P(J_2 \cap R)}{P(J_1 \cap R) + P(J_2 \cap R) + P(J_3 \cap R)}$$
$$\frac{\left(\frac{1}{18}\right)}{\left(\frac{1}{8}\right) + \left(\frac{1}{18}\right) + \left(\frac{4}{27}\right)} = \frac{12}{71} \approx 0.17$$

Adopted from: somewhere I cannot trace the source

Example: Tree Diagrams

All tractors made by a company are produced on one of three assembly lines, named Red, White, and Blue. The chances that a tractor will not start when it rolls off of a line are 6%, 11%, and 8% for lines Red, White, and Blue, respectively. 48% of the company's tractors are made on the Red line and 31% are made on the Blue line. What fraction of the company's tractors do not start when they roll off of an assembly line?

Adopted from: somewhere I cannot trace the source

Define Events

- Let R be the event that the tractor was made by the red company
- Let W be the event that the tractor was made by the white company
- Let B be the event that the tractor was made by the blue company
- Let D be the event that the tractor won't start

Adopted from: somewhere I cannot trace the source

Extracting the Information

- In terms of probabilities for the events we've defined, this what we know:

$$P(R) = 0.48$$

$$P(W) = 0.21$$

$$P(B) = 0.31$$

$$P(D|R) = 0.06$$

$$P(D|W) = 0.11$$

$$P(D|B) = 0.08$$

Adopted from: somewhere I cannot trace the source

What are We Trying to Find?

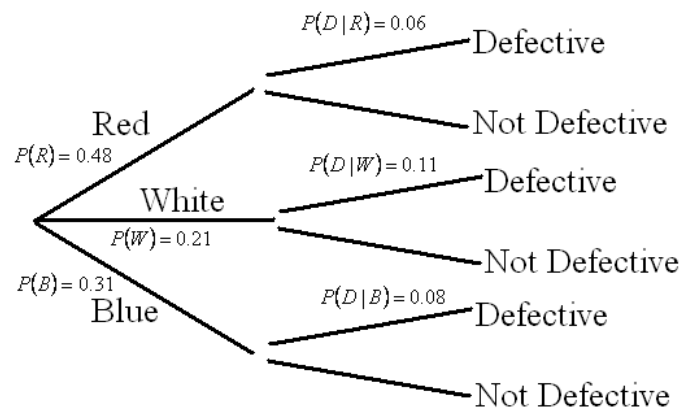
- Our problem asked for us to find:
 - The fraction of the company's tractors that do not start when rolled off the assembly line?
 - In other words:

$$P(D)$$

Adopted from: somewhere I cannot trace the source

Tree Diagram

- Because there are three companies producing tractors we will divide or *partition* our sample space along those events only this time we'll be using a tree diagram



Adopted from: somewhere I cannot trace the source

Follow the Branch

- There are three ways for a tractor to be defective:
 - ☐ It was made by the Red Company
 - ☐ It was made by the White Company
 - ☐ It was made by the Blue Company

- To find all the defective ones, we need to know how many were:
 - ☐ Defective and made by the Red Company?
 - ☐ Defective and made by the White Company?
 - ☐ Defective and made by the Blue Company?

Adopted from: somewhere I cannot trace the source

Choose the Path

- In terms of probabilities, we want:

$$P(R \cap D)$$

$$P(W \cap D)$$

$$P(B \cap D)$$

Adopted from: somewhere I cannot trace the source

Computing Probabilities

- To find each of these probabilities we simply need to multiply the probabilities along each branch
- Doing this we find

$$\begin{aligned}P(R \cap D) &= P(D|R)P(R) \\ P(W \cap D) &= P(D|W)P(W) \\ P(B \cap D) &= P(D|B)P(B)\end{aligned}$$

Adopted from: somewhere I cannot trace the source

Putting it all Together

- Because each of these events represents an instance where a tractor is defective to find the total probability that a tractor is defective, we simply add up all our probabilities:

$$P(D) = P(D|R)P(R) + P(D|W)P(W) + P(D|B)P(B)$$

Adopted from: somewhere I cannot trace the source

Bonus Question

- What is the probability that a tractor came from the red company given that it was defective?

$$P(R|D) = \frac{P(R \cap D)}{P(D)}$$

Adopted from: somewhere I cannot trace the source

I thought this was called Bayes' Theorem?

- Bayes' Theorem
- Suppose that $B_1, B_2, B_3, \dots, B_n$ partition the outcomes of an experiment and that A is another event. For any number, k , with $1 \leq k \leq n$, we have the formula:

$$P(B_k|A) = \frac{P(A|B_k) \cdot P(B_k)}{P(A)} = \frac{P(A|B_k) \cdot P(B_k)}{\sum_{i=1}^n P(A|B_i) \cdot P(B_i)}$$

Adopted from: somewhere I cannot trace the source

In English Please?

- What does Bayes' Formula helps to find?

- Helps us to find:

$$P(B|A)$$

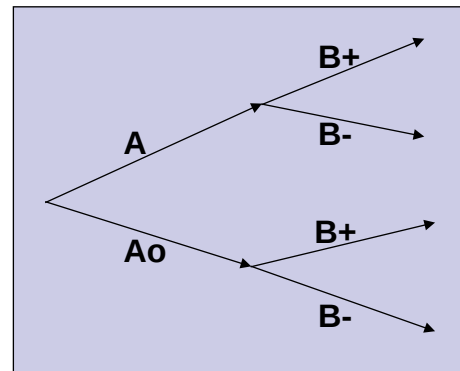
- By having already known:

$$P(A|B)$$

Adopted from: somewhere I cannot trace the source

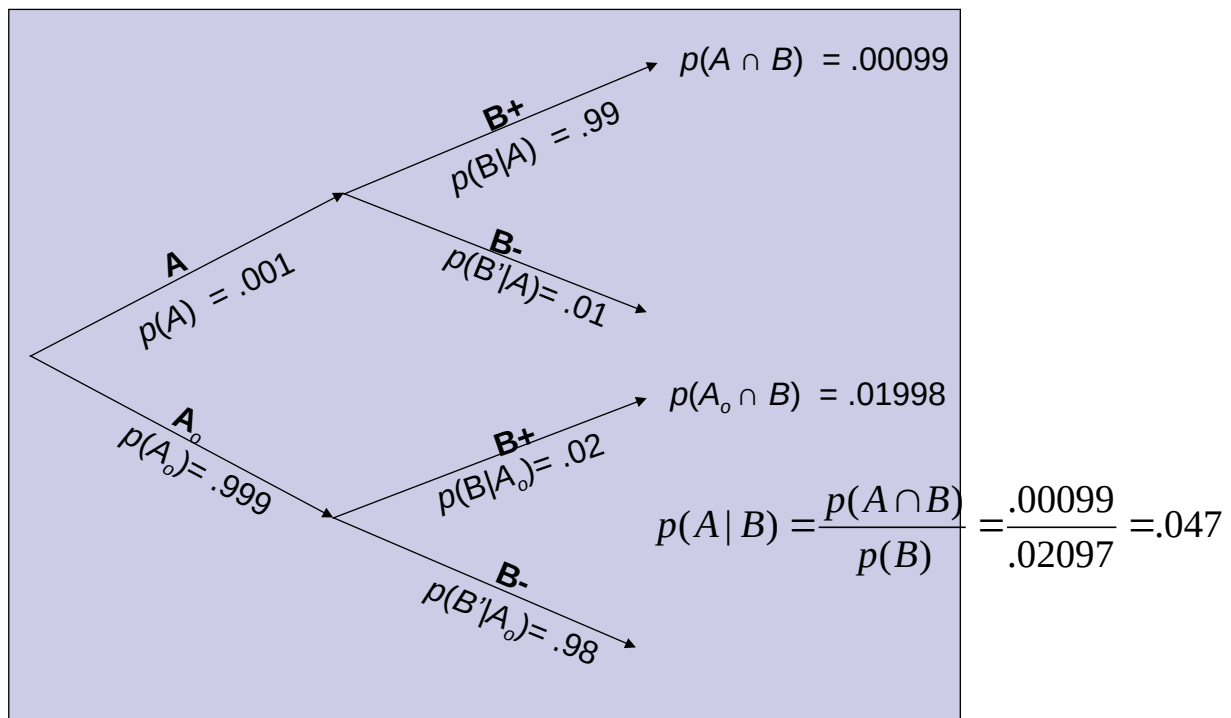
Numerical Example

- Only 1 in 1000 people have rare disease A
 - $TP = .99$ $FP = .02$
 - If one randomly tested individual is positive, what is the probability they have the disease
- Label events:
 - A = has disease A_o = no disease
 - B = Positive test result
- Examine probabilities
 - $p(A) = .001$
 - $p(A_o) = .999$
 - $p(B|A) = .99$
 - $p(B|A_o) = .02$



Adopted from: somewhere I cannot trace the source

Numerical Example



Adopted from: somewhere I cannot trace the source

MCTE 512

Artificial Intelligence and Machine Learning for Engineers

Lecture 5 – Naive Bayes

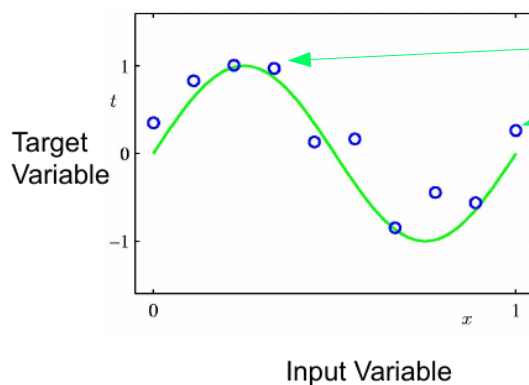
Rodrigo S. Jamisola Jr.

Associate Professor in Robotics and Machine Learning
Simulations, Machine Learning, Robotics and Technopreneurship Laboratory (SMaRT Lab)
Dept. of Mechanical, Energy and Industrial Engineering
Faculty of Engineering and Technology
Botswana International University of Science and Technology



Review on Polynomial Regression

- Given data
 - We want to estimate a function $y = f(x)$ that best approximates the data



Data = $\{\mathbf{x}, \mathbf{t}\}$

$$y = f(\mathbf{x})$$

$$y = w_0x^0 + w_1x^1 + w_2x^2 + \dots + w_Mx^M$$

$$E = \|\mathbf{y} - \mathbf{t}\|$$

We need to find the function f

- that best approximates
- the relationship of input vs. output
- from the given data

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

Review on Decision Trees

- A decision tree is a support tool that
 - Uses a tree-like graph or model of decisions
 - And their possible consequences
 - Include chance event of outcomes, resource costs, and utility
- Example: A decision tree to *PlayTennis*

$$Decision = f(Inputs)$$

$$Decision = f(Outlook, Humidity, Wind)$$

$$IG(X_a) = H(Y) - H(Y|X_a)$$

$$H(Y) = - \sum_{i=1}^k P(Y = y_i) \log_2 P(Y = y_i)$$

$$H(Y|X) = - \sum_{j=1}^v P(X = x_j) \sum_{i=1}^k P(Y = y_i|X = x_j) \log_2 P(Y = y_i|X = x_j)$$

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

Review on Bayes Rule

Conditional Probability = f(Inputs)

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Which is shorthand for:

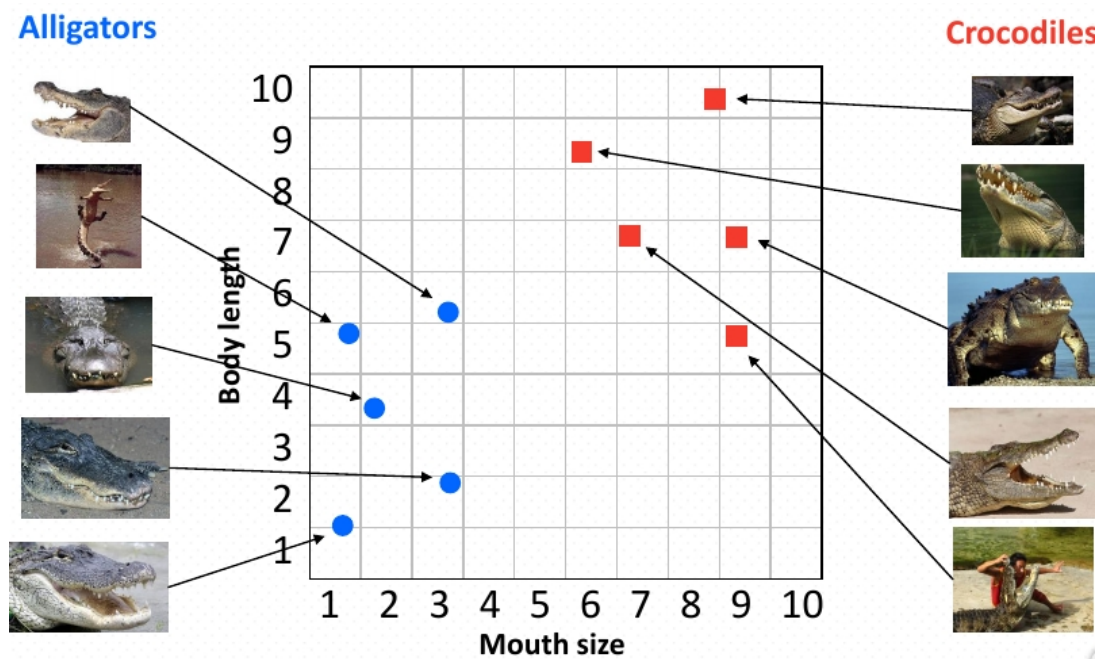
$$(\forall i, j) P(Y = y_i | X = x_j) = \frac{P(X = x_j | Y = y_i) P(Y = y_i)}{P(X = x_j)}$$

Equivalently:

$$(\forall i, j) P(Y = y_i | X = x_j) = \frac{P(X = x_j | Y = y_i) P(Y = y_i)}{\sum_k P(X = x_j | Y = y_k) P(Y = y_k)}$$

Adopted from: Tom M. Mitchell

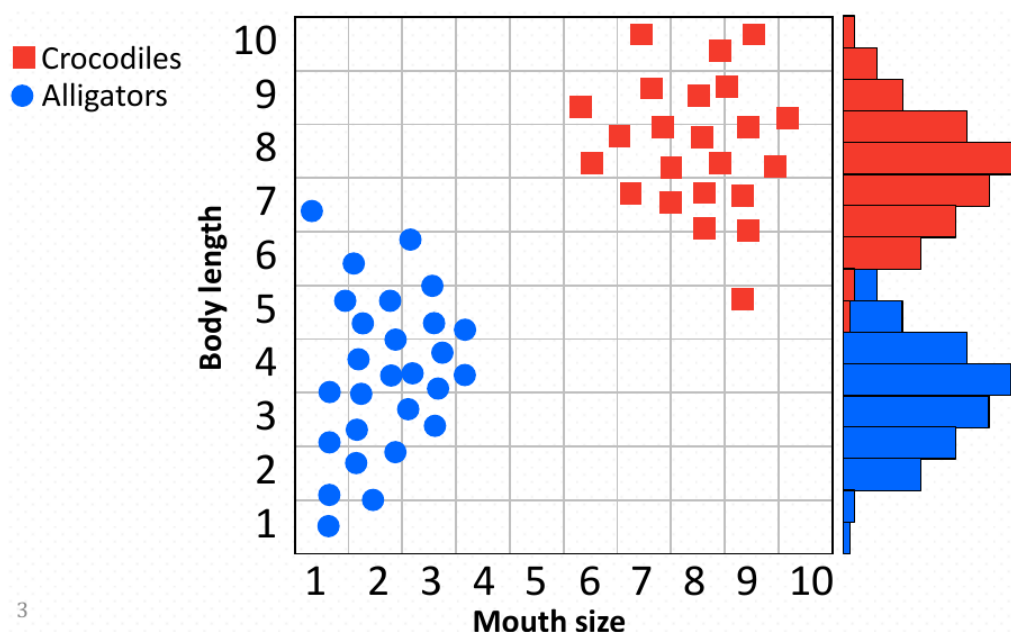
Naïve Bayes Classifier



Adopted from: Eamonn Keogh

Naïve Bayes Classifier

Suppose we had a lot of data. We could use that data to build a histogram. Below is one built for the *body length* feature:

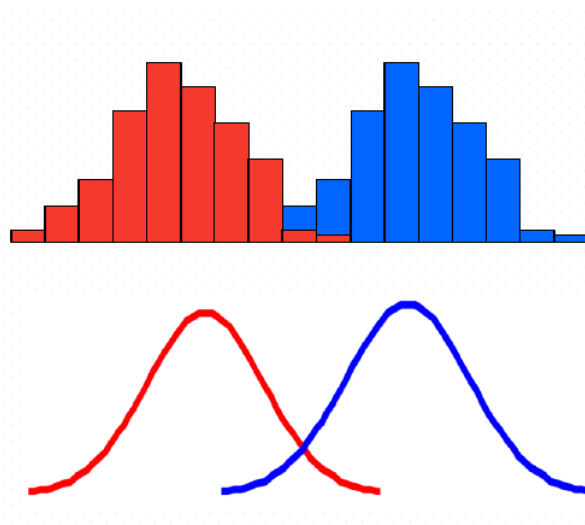


3

Adopted from: Eamonn Keogh

Naïve Bayes Classifier

We can summarize these histograms as *two* normal distributions.

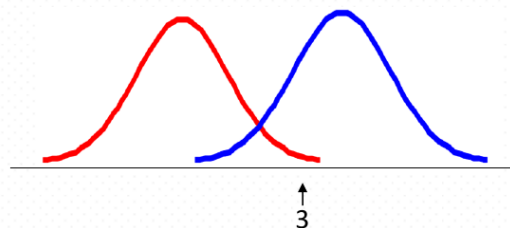


Adopted from: Eamonn Keogh

Naïve Bayes Classifier

- * Suppose we wish to classify a new animal that we just found. Its body length is 3 units. How can we classify it?
- * *One way to do this is, given the distributions of that feature, we can analyze which class is more probable: **Crocodile** or **Alligator**.*
- * *More formally:*

$p(c_j|d)$ = probability of class c_j , given that we observed d



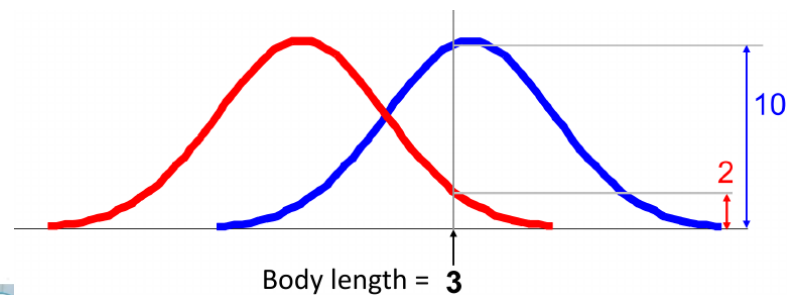
Adopted from: Eamonn Keogh

Naïve Bayes Classifier

$p(c_j|d)$ = probability of class c_j , given that we observed d

$$p(\text{Alligator} | \text{body length} = 3) = 10 / (10 + 2) = 0.833$$

$$p(\text{Crocodile} | \text{body length} = 3) = 2 / (10 + 2) = 0.166$$



Adopted from: Eamonn Keo

Naïve Bayes Classifier

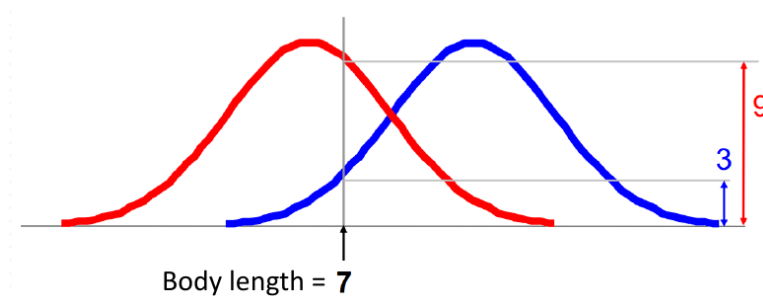
$p(c_j|d)$ = probability of class c_j , given that we observed d

$$p(\text{Alligator} | \text{body length} = 7) = 3 / (3 + 9) = 0.25$$

$$p(\text{Crocodile} | \text{body length} = 7) = 9 / (3 + 9) = 0.75$$



Adopted from: Eamonn Keogh

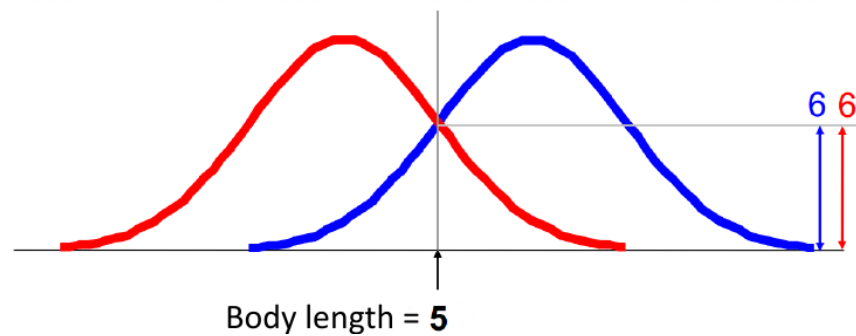


Naïve Bayes Classifier

$p(c_j|d)$ = probability of class c_j , given that we observed d

$$p(\text{Alligator} | \text{body length} = 5) = 6 / (6 + 6) = 0.6$$

$$p(\text{Crocodile} | \text{body length} = 5) = 6 / (6 + 6) = 0.6$$



Adopted from: Eamonn Keogh

Naïve Bayes Classifier

✿ This visual intuition describes a simple Bayes classifier commonly known as:

- ✚ Naïve Bayes
- ✚ Simple Bayes
- ✚ Idiot Bayes

✿ While going through the math, keep in mind the basic idea:

- ✚ *Given a new unseen instance, we (1) find its probability of it belonging to each class, and (2) pick the most probable.*

Adopted from: Eamonn Keogh

Bayes Theorem

Diagram illustrating Bayes Theorem with labels for each part of the equation:

$$P(c_j|d) = \frac{P(d|c_j)P(c_j)}{P(d)}$$

Labels and arrows:

- Posterior probability points to $P(c_j|d)$
- Likelihood points to $P(d|c_j)$
- Class prior probability points to $P(c_j)$
- Predictor prior probability points to $P(d)$

- $P(c_j|d)$ = probability of instance d being in class c_j
- $P(d|c_j)$ = probability of generating instance d given class c_j
- $P(c_j)$ = probability of occurrence of class c_j
- $P(d)$ = probability of instance d occurring

Adopted from: Eamonn Keogh

Naïve Bayes Classifier

❁ What if we have several features?

Name	Over 6ft	Eye Color	Hair Style	Sex
Morgan	Yes	Blue	Long	Female
Bob	No	Brown	None	Male
Vincent	Yes	Brown	Short	Male
Amanda	No	Brown	Short	Female
Reid	No	Blue	Short	Male
Lauren	No	Purple	Long	Female
Elisa	Yes	Brown	Long	Female

Adopted from: Eamonn Keogh

Naïve Bayes Classifier

$$P(X_i, \dots, X_N | Y) = \prod_i P(X_i | Y)$$

- *Naive Bayes assumes that all features are independent (i.e., they have independent distributions).
- *The probability of class c_j generating instance d can be estimated as:

$$\begin{aligned} p(d|c_j) &= p(d_1, d_2, d_3, \dots, d_n | c_j) \\ &= p(d_1 | d_2, d_3, \dots, d_n, c_j) \times p(d_2 | d_3, \dots, d_n, c_j) \times \dots \times p(d_n | c_j) \\ &= p(d_1 | c_j) \times p(d_2 | c_j) \times \dots \times p(d_n | c_j) \end{aligned}$$

Probability of class c_j
generating the observed
value of feature 1

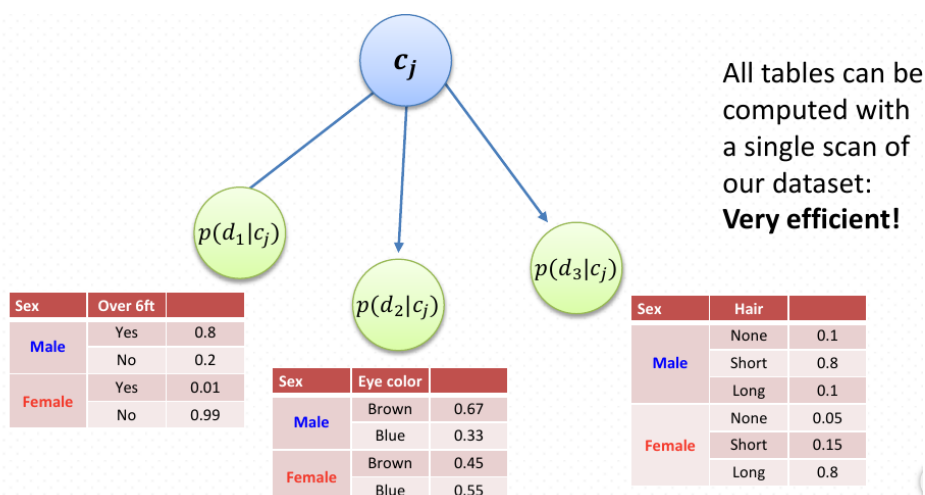
Probability of class c_j
generating the observed
value of feature 2

Chain rule of probability
dependence

Naïve Bayes Classifier

* A visual representation:

+ Each class implies certain feature with certain probabilities



Adopted from: Eamonn Keogh

Naïve Bayes Classifier

- Naïve Bayes is not sensitive to irrelevant features.
 - For instance, if we were trying to classify a person's sex based on a set of features that included *eye color* (which is irrelevant to their gender):

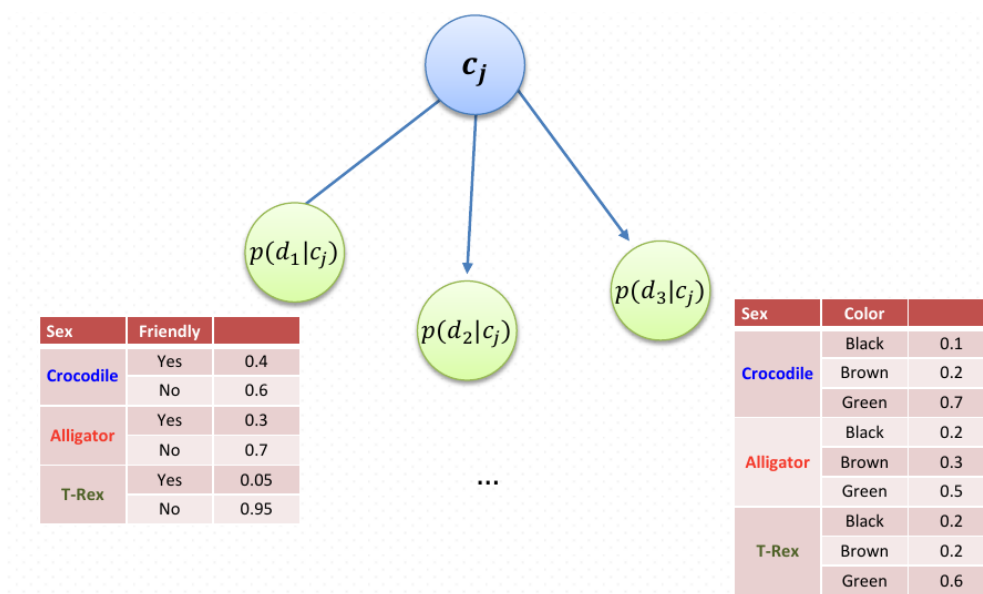
$$P(\text{Reid} | c_j) = P(\text{eye} = \text{blue} | c_j) \times P(\text{wears makeup} = \text{Yes} | c_j) \times \dots$$
$$P(\text{Reid} | \text{Female}) = 9,000/10,000 \times 9,500/10,000 \times \dots$$
$$P(\text{Reid} | \text{Male}) = 9,007/10,000 \times 1/10,000 \times \dots$$

Nearly identical!

- The more data, the better!

Naïve Bayes Classifier

* Naive Bayes works with any number of classes or features



Adopted from: Eamonn Keogh

Naïve Bayes – Class Exercise

❁ Predict if Bob will default his loan

Bob
Home owner: *No*
Marital status: *Married*
Job experience: *3*

Home owner	Marital Status	Job experience (1-5)	Defaulted
Yes	Single	3	No
No	Married	4	No
No	Single	5	No
Yes	Married	4	No
No	Divorced	2	Yes
No	Married	4	No
Yes	Divorced	2	No
No	Married	3	Yes
No	Married	3	No
Yes	Single	2	Yes

Naïve Bayes – Class Exercise

Bob

Home owner: *No*

Marital status: *Married*

Job experience: *3*

- $P(Y = \text{No}) = 7/10$
- $P(\text{Home owner} = \text{No} | Y = \text{No}) = 4/7$
- $P(\text{Marital status} = \text{Married} | Y = \text{No}) = 4/7$
- $P(\text{Job experience} = 3 | Y = \text{No}) = 2/7$

Home owner	Marital Status	Job experience (1-5)	Defaulted
Yes	Single	3	No
No	Married	4	No
No	Single	5	No
Yes	Married	4	No
No	Divorced	2	Yes
No	Married	4	No
Yes	Divorced	2	No
No	Married	3	Yes
No	Married	3	No
Yes	Single	2	Yes

$$P(\text{Bob will NOT default}) = \frac{7}{10} \times \frac{4}{7} \times \frac{4}{7} \times \frac{2}{7} = \mathbf{0.065}$$

Naïve Bayes – Class Exercise

Bob

Home owner: *No*

Marital status: *Married*

Job experience: 3

- $P(Y = \text{Yes}) = 3/10$
- $P(\text{Home owner} = \text{No} | Y = \text{Yes}) = 1/3$
- $P(\text{Marital status} = \text{Married} | Y = \text{Yes}) = 1/3$
- $P(\text{Job experience} = 3 | Y = \text{Yes}) = 1/3$

Home owner	Marital Status	Job experience (1-5)	Defaulted
Yes	Single	3	No
No	Married	4	No
No	Single	5	No
Yes	Married	4	No
No	Divorced	2	Yes
No	Married	4	No
Yes	Divorced	2	No
No	Married	3	Yes
No	Married	3	No
Yes	Single	2	Yes

$$P(\text{Bob will default}) = \frac{3}{10} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3} = \mathbf{0.011}$$

Naïve Bayes – Class Exercise

Sample ML task

SYMPTOM	OCCUPATION	AILMENT
sneezing	nurse	flu
sneezing	farmer	hayfever
headache	builder	concussion
headache	builder	flu
sneezing	teacher	flu
headache	teacher	concussion
sneezing	builder	???

What ailment should we predict for a sneezing builder and why?

Introduction

Clustering and nearest-neighbour methods are ideally suited for use with numeric data.

However, data often use using categorical values, i.e., names or symbols.

In this situation, it may be better to use a probabilistic method, such as the Naive Bayes Classifier (NBC).

Probabilities

Let's say we have some data that lists symptoms and ailments for everybody in a certain group.

SYMPTOM	AILMENT
---------	---------

Adopted from: Chris Thornton

Naïve Bayes – Class Exercise

Normalization needed

Unfortunately, this value is purely based on cases of flu. It doesn't take into account how common this ailment is.

We need to factor in the probability of this combination of attributes associating with flu in particular, rather than some other ailment.

We do this by expressing the value in proportion to the probability of seeing the combination of attributes.

$$P(\text{flu} \mid \text{sneezing}, \text{builder}) = \frac{P(\text{flu})P(\text{sneezing}, \text{builder} \mid \text{flu})}{P(\text{sneezing}, \text{builder})}$$

This gives us the value we want.

The answer

Assemble all the constituents needed

$$P(\text{flu}) = 0.5$$

$$P(\text{sneezing} \mid \text{flu}) = 0.66$$

$$P(\text{builder} \mid \text{flu}) = 0.33$$

$$P(\text{sneezing}, \text{builder} \mid \text{flu}) = (0.66 \times 0.33) = 0.22$$

$$P(\text{sneezing}) = 0.5$$

$$P(\text{builder}) = 0.33$$

$$P(\text{sneezing}, \text{builder}) = (0.5 \times 0.33) = 0.165$$

Plug values into the formula:

$$\frac{0.5 \times 0.22}{0.165} = 0.66$$

It turns out the sneezing builder has flu with probability 0.66.

Bayes rule

What we've worked out here is just an application of **Bayes rule**, the standard formula for inverting conditional probabilities.

Adopted from: Chris Thornton

Naïve Bayes – Class Exercise

Estimating Prior Probabilities

- Probabilities of different target values estimated from frequencies over 14 training examples

$$P(\text{PlayTennis} = \text{yes}) = 9/14 = .64 \quad P(\text{PlayTennis} = \text{no}) = 5/14 = .36$$

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Adopted from: Chris Thornton

Naïve Bayes – Class Exercise

Estimating Conditional Probabilities

- Similarly, can estimate conditional probabilities. For example, those for *Wind = strong* are:

$$P(Wind = strong | PlayTennis = yes) = 3/9 = .33$$

$$P(Wind = strong | PlayTennis = no) = 3/5 = .60$$

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Adopted from: Chris Thornton

Maximum A Posteriori

Based on Bayes Rule, we can compute the *maximum a posteriori* hypothesis for the data:

$$\begin{aligned}(2) \quad h_{\text{MAP}} &= \arg \max_{h \in H} P(h|D) \\ &= \arg \max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\ &= \arg \max_{h \in H} P(D|h)P(h)\end{aligned}$$

H : set of all hypotheses

Note that we can drop $P(D)$ as the probability of the data is constant (and independent of the hypothesis).

Maximum Likelihood

Now assume that all hypotheses are equally probable a priori, i.e, $P(h_i) = P(h_j)$ for all $h_i, h_j \in H$.

This is called assuming a *uniform prior*. It simplifies computing the posterior:

$$(3) \quad h_{\text{ML}} = \arg \max_{h \in H} P(D|h)$$

This hypothesis is called the *maximum likelihood hypothesis*.

MCTE 512

Artificial Intelligence and Machine Learning for Engineers

Lecture 6 – Support Vector Machine

Rodrigo S. Jamisola Jr.

Associate Professor in Robotics and Machine Learning
Simulations, Machine Learning, Robotics and Technopreneurship Laboratory (SMaRT Lab)
Dept. of Mechanical, Energy and Industrial Engineering
Faculty of Engineering and Technology
Botswana International University of Science and Technology

Review on Polynomial Regression

Data : $\{\mathbf{x}, \mathbf{t}\}$

$$y = f(\mathbf{x})$$

$$= w_0x^0 + w_1x^1 + w_2x^2 + \dots + w_Mx^M$$

$$E(\mathbf{w}) = \|\mathbf{y} - \mathbf{t}\| = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2$$

$$\frac{\partial E(\mathbf{w})}{\partial w_i} = \sum_{n=1}^N \left\{ \sum_{j=0}^M w_j x_n^j - t_n \right\} x_n^i$$

$$\mathbf{w} = \mathbf{w} - \eta \left[\frac{\partial E(\mathbf{w})}{\partial w_0}, \dots, \frac{\partial E(\mathbf{w})}{\partial w_M} \right]$$

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

Review on Decision Trees

Decision = $f(Inputs)$

for example

$\{Yes, No\} = f(Outlook, Humidity, Wind)$

$$IG(X_a) = H(Y) - H(Y|X_a)$$

$$H(Y) = - \sum_{i=1}^k P(Y = y_i) \log_2 P(Y = y_i)$$

$$H(Y|X_a) = - \sum_{j=1}^v P(X_a = x_j) \sum_{i=1}^k P(Y = y_i | X_a = x_j) \log_2 P(Y = y_i | X_a = x_j)$$

To decide next layer, choose the highest information gain

Adopted from: <https://people.csail.mit.edu/dsontag/>, Luke Zettlemoyer, Carlos Guestrin, Andrew Moore, and Tom Mitchell

Review on Bayes Rule

Conditional Probability = f(Inputs)

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Which is shorthand for:

$$(\forall i, j) P(Y = y_i | X = x_j) = \frac{P(X = x_j | Y = y_i) P(Y = y_i)}{P(X = x_j)}$$

Equivalently:

$$(\forall i, j) P(Y = y_i | X = x_j) = \frac{P(X = x_j | Y = y_i) P(Y = y_i)}{\sum_k P(X = x_j | Y = y_k) P(Y = y_k)}$$

Adopted from: Tom M. Mitchell

Review on Naive Bayes Rule

$$P(X_1, \dots, X_N | Y) = \prod_i P(X_i | Y)$$

$$p(\mathbf{d} | c_j) = p(d_1, d_2, d_3, \dots, d_n | c_j)$$

By Chain Rule of Probability Dependence
-all parameters are independent of each other

$$= p(d_1 | d_2, d_3, \dots, d_n, c_j) \times p(d_2 | d_3, \dots, d_n, c_j) \times \dots \times p(d_n | c_j)$$

$$= p(d_1 | c_j) \times p(d_2 | c_j) \times \dots \times p(d_n | c_j)$$

* Maximum A Posteriori

+ probability of data is constant

$$\begin{aligned} h_{\text{MAP}} &= \arg \max_{h \in H} P(h | D) \\ &= \arg \max_{h \in H} \frac{P(D | h) P(h)}{P(D)} \\ &= \arg \max_{h \in H} P(D | h) P(h) \end{aligned}$$

* Maximum Likelihood

+ all hypotheses are equally probable

$$h_{ML} = \arg \max_{h \in H} P(D | h)$$

Adopted from: Tom M. Mitchell

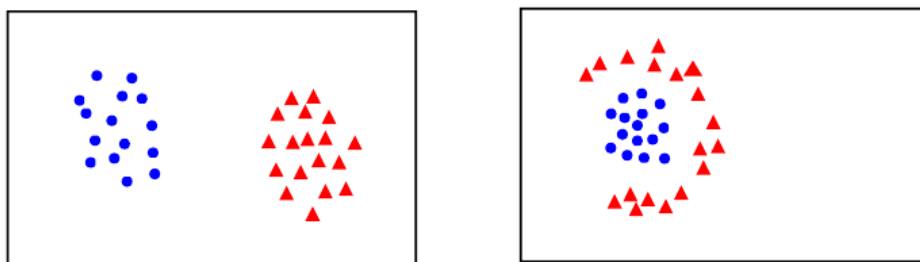
Support Vector Machine

* Binary Classification

Given training data $(\mathbf{x}_i, y_i)$ for $i = 1 \dots N$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, learn a classifier $f(\mathbf{x})$ such that

$$f(\mathbf{x}_i) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

i.e. $y_i f(\mathbf{x}_i) > 0$ for a correct classification.

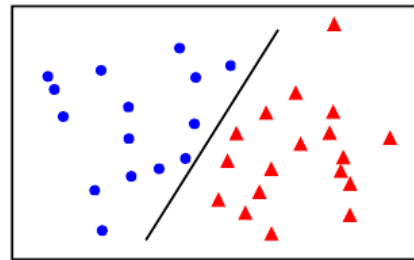
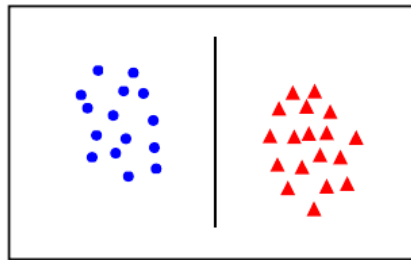


Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

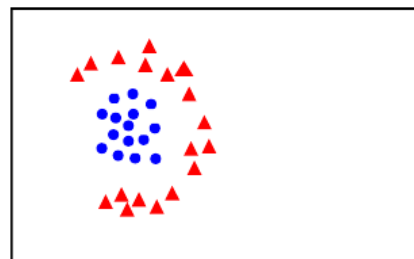
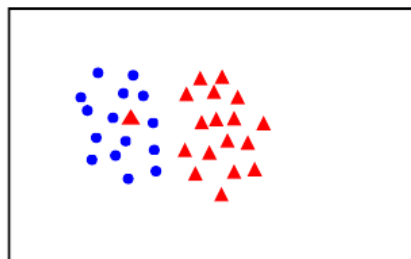
Support Vector Machine

* Linear Separability

linearly
separable



not
linearly
separable



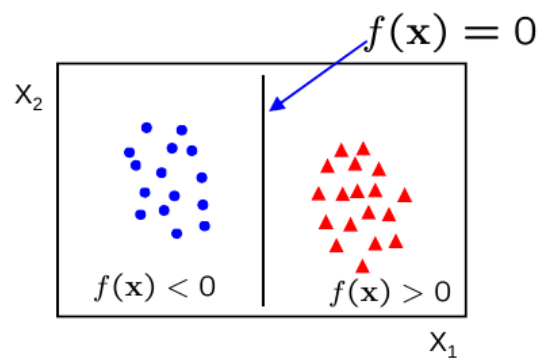
Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

Support Vector Machine

* Linear Classifiers

A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$



- in 2D the discriminant is a line
- $\mathbf{w}$ is the **normal** to the line, and b the **bias**
- $\mathbf{w}$ is known as the **weight vector**

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

Support Vector Machine

* The Perceptron Classifier

Given linearly separable data $\mathbf{x}_i$ labelled into two categories $y_i = \{-1, 1\}$, find a weight vector $\mathbf{w}$ such that the discriminant function

$$f(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i + b$$

separates the categories for $i = 1, \dots, N$

- how can we find this separating hyperplane ?

The Perceptron Algorithm

Write classifier as $f(\mathbf{x}_i) = \tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}_i + w_0 = \mathbf{w}^\top \mathbf{x}_i$

where $\mathbf{w} = (\tilde{\mathbf{w}}, w_0)$, $\mathbf{x}_i = (\tilde{\mathbf{x}}_i, 1)$

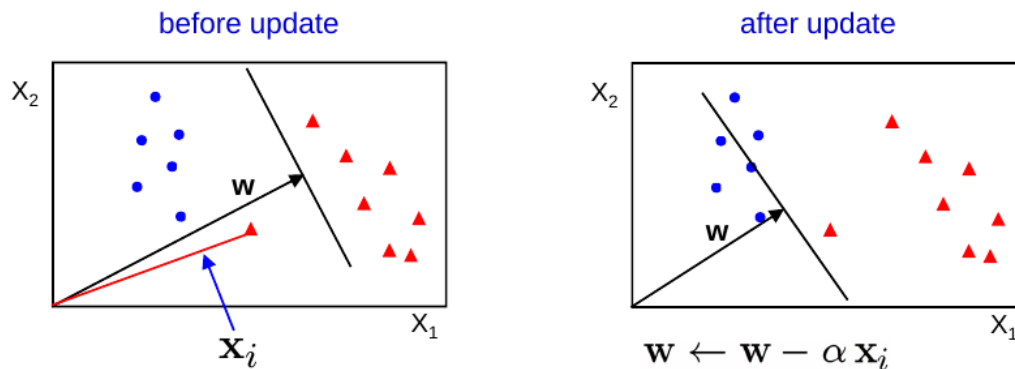
- Initialize $\mathbf{w} = 0$
- Cycle through the data points $\{\mathbf{x}_i, y_i\}$
 - if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha \text{sign}(f(\mathbf{x}_i)) \mathbf{x}_i$
- Until all the data is correctly classified

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

Support Vector Machine

❁ Example in 2d

- Initialize $\mathbf{w} = 0$
- Cycle through the data points $\{\mathbf{x}_i, y_i\}$
 - if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha \text{sign}(f(\mathbf{x}_i)) \mathbf{x}_i$
- Until all the data is correctly classified

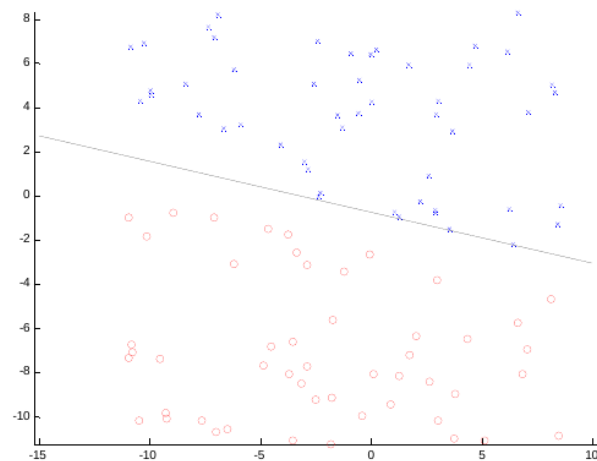


NB after convergence $\mathbf{w} = \sum_i^N \alpha_i \mathbf{x}_i$

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

Support Vector Machine

Perceptron
example

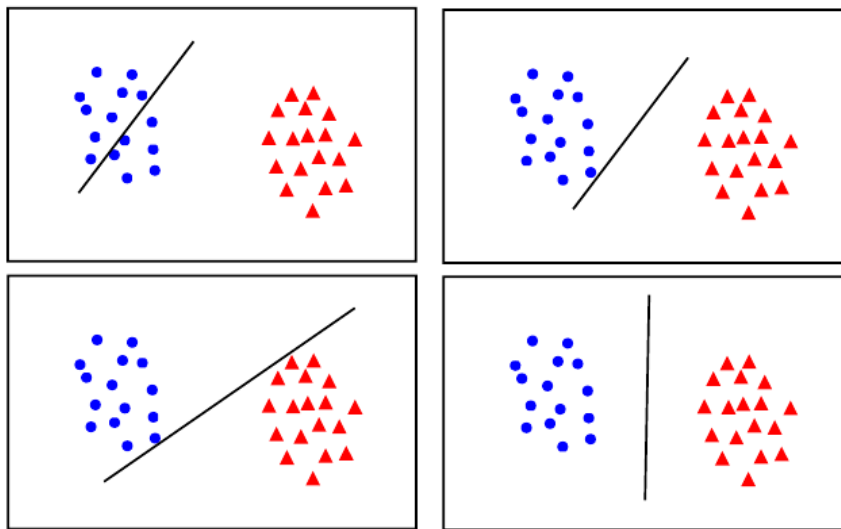


- if the data is linearly separable, then the algorithm will converge
- convergence can be slow ...
- separating line close to training data
- we would prefer a larger margin for generalization

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

Support Vector Machine

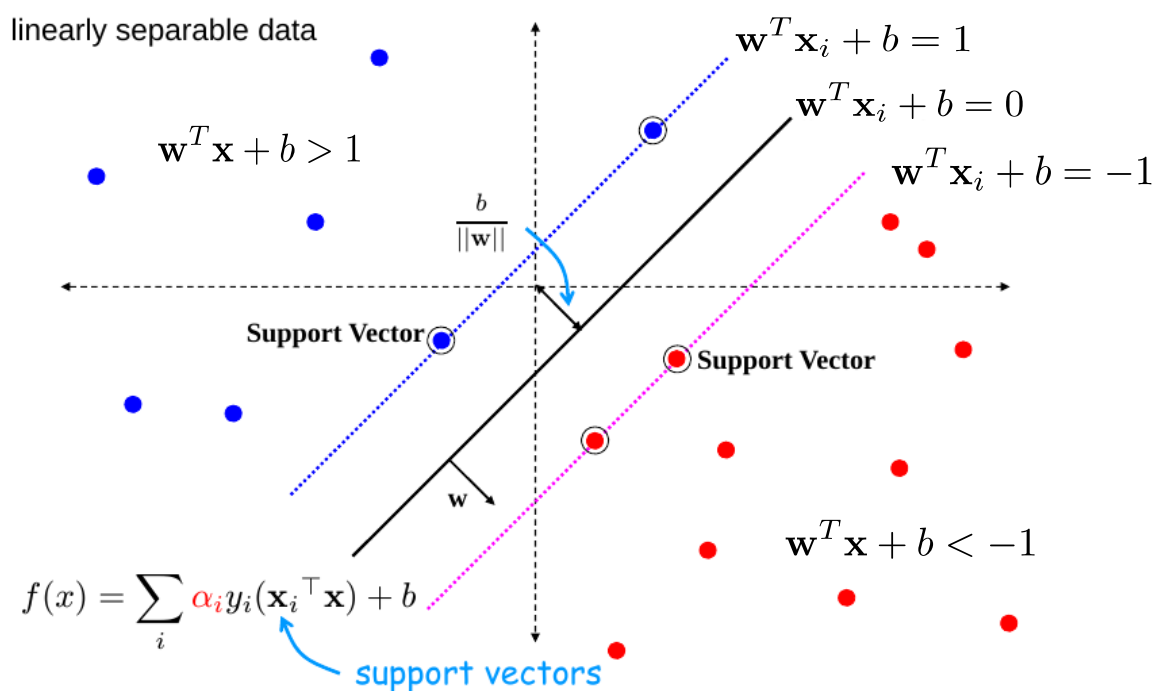
❁ What is the best w ?



- **maximum margin** solution: most stable under perturbations of the inputs

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

Support Vector Machine



Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

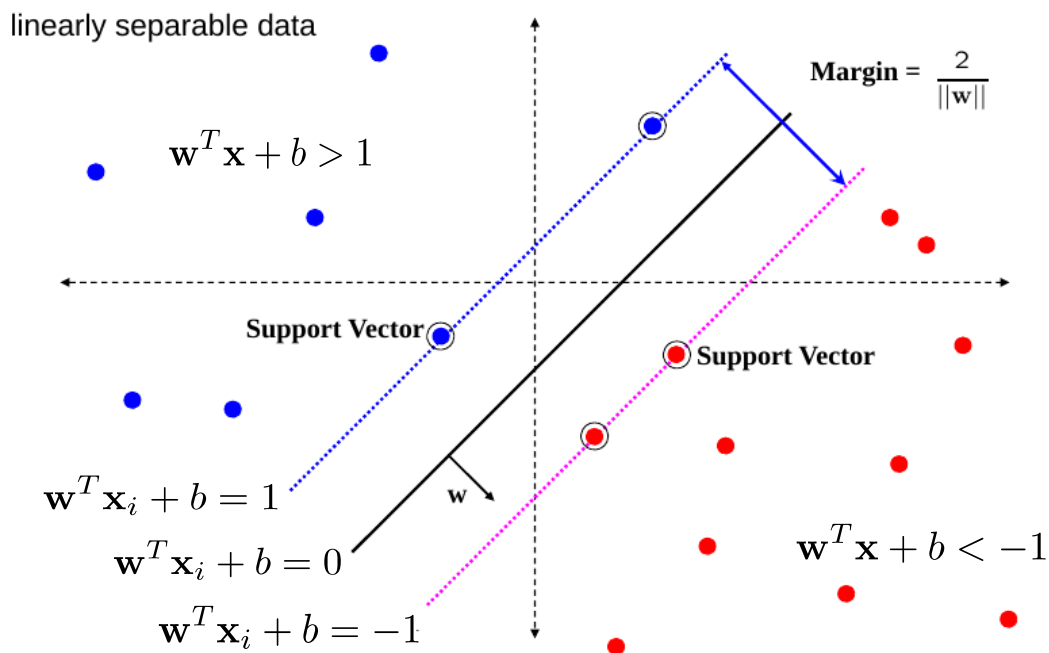
SVM Optimization

- Since $\mathbf{w}^\top \mathbf{x} + b = 0$ and $c(\mathbf{w}^\top \mathbf{x} + b) = 0$ define the same plane, we have the freedom to choose the normalization of $\mathbf{w}$
- Choose normalization such that $\mathbf{w}^\top \mathbf{x}_+ + b = +1$ and $\mathbf{w}^\top \mathbf{x}_- + b = -1$ for the positive and negative support vectors respectively
- Then the **margin** is given by

$$\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot (\mathbf{x}_+ - \mathbf{x}_-) = \frac{\mathbf{w}^\top (\mathbf{x}_+ - \mathbf{x}_-)}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

SVM Optimization



Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

SVM Optimization

- Learning the SVM can be formulated as an optimization:

$$\max_{\mathbf{w}} \frac{2}{\|\mathbf{w}\|} \text{ subject to } \mathbf{w}^\top \mathbf{x}_i + b \begin{cases} \geq 1 & \text{if } y_i = +1 \\ \leq -1 & \text{if } y_i = -1 \end{cases} \text{ for } i = 1 \dots N$$

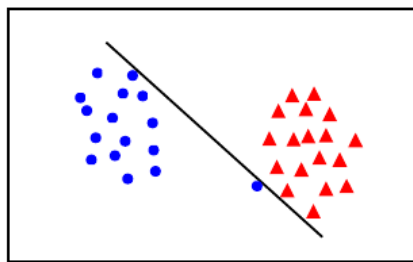
- Or equivalently

$$\min_{\mathbf{w}} \|\mathbf{w}\|^2 \text{ subject to } y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \text{ for } i = 1 \dots N$$

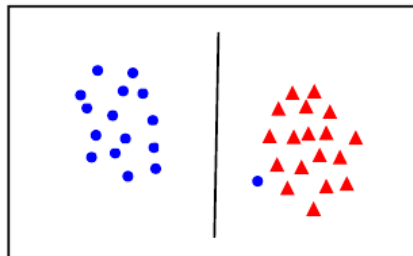
- This is a quadratic optimization problem subject to linear constraints and there is a unique minimum

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

SVM Optimization



- the points can be linearly separated but there is a very narrow margin

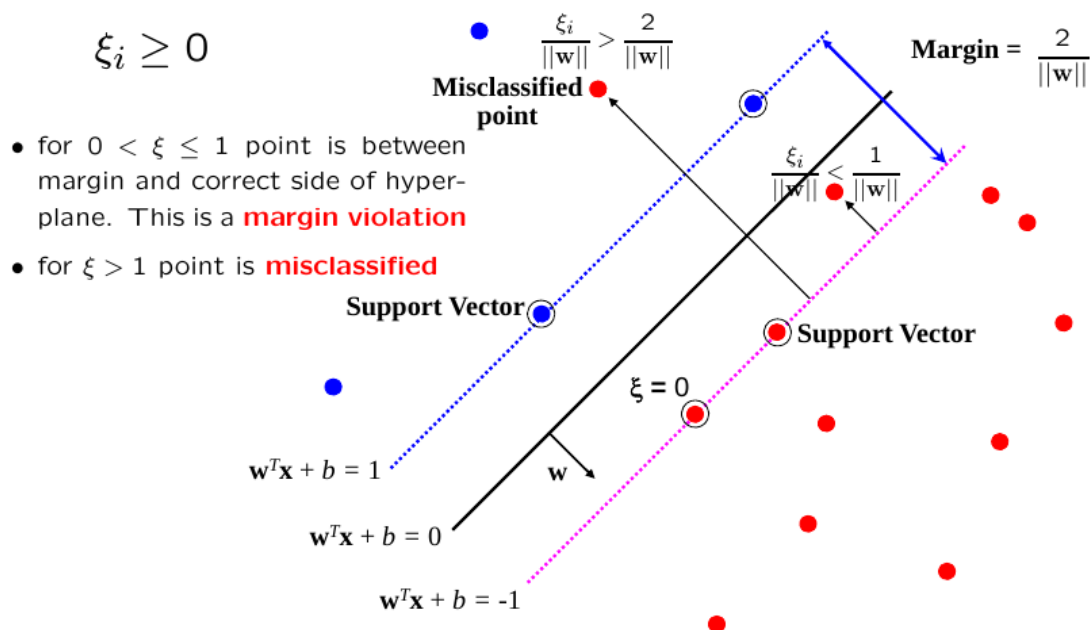


- but possibly the large margin solution is better, even though one constraint is violated

In general there is a trade off between the margin and the number of mistakes on the training data

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

SVM Optimization



Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

SVM Optimization

* Soft margin solution

The optimization problem becomes

$$\min_{\mathbf{w} \in \mathbb{R}^d, \xi_i \in \mathbb{R}^+} \|\mathbf{w}\|^2 + C \sum_i^N \xi_i$$

subject to

$$y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \text{ for } i = 1 \dots N$$

- Every constraint can be satisfied if ξ_i is sufficiently large
- C is a **regularization** parameter:
 - small C allows constraints to be easily ignored $\rightarrow$ large margin
 - large C makes constraints hard to ignore $\rightarrow$ narrow margin
 - $C = \infty$ enforces all constraints: hard margin
- This is still a quadratic optimization problem and there is a unique minimum. Note, there is only one parameter, C .

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

SVM Optimization

Learning an SVM has been formulated as a **constrained** optimization problem over $\mathbf{w}$ and ξ

$$\min_{\mathbf{w} \in \mathbb{R}^d, \xi_i \in \mathbb{R}^+} \|\mathbf{w}\|^2 + C \sum_i^N \xi_i \text{ subject to } y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \text{ for } i = 1 \dots N$$

The constraint $y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i$, can be written more concisely as

$$y_i f(\mathbf{x}_i) \geq 1 - \xi_i$$

which, together with $\xi_i \geq 0$, is equivalent to

$$\xi_i = \max(0, 1 - y_i f(\mathbf{x}_i))$$

Hence the learning problem is equivalent to the **unconstrained** optimization problem over $\mathbf{w}$

$$\min_{\mathbf{w} \in \mathbb{R}^d} \underbrace{\|\mathbf{w}\|^2}_{\text{regularization}} + C \sum_i^N \underbrace{\max(0, 1 - y_i f(\mathbf{x}_i))}_{\text{loss function}}$$

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

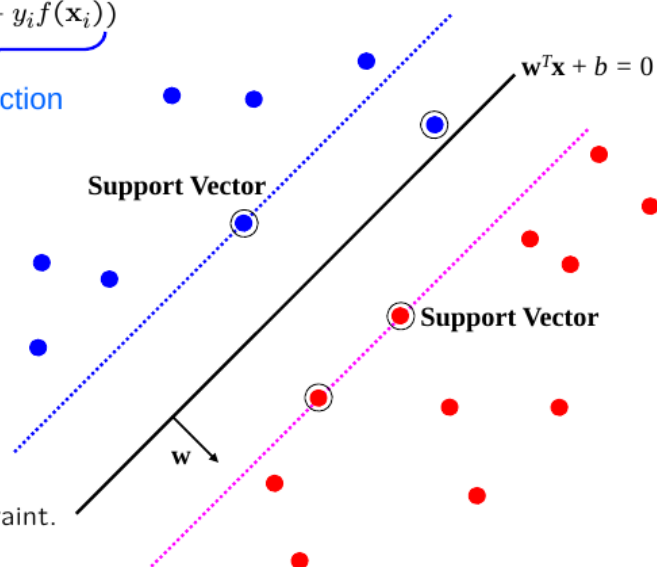
SVM Optimization

* Loss function

$$\min_{\mathbf{w} \in \mathbb{R}^d} \|\mathbf{w}\|^2 + C \sum_i^N \underbrace{\max(0, 1 - y_i f(\mathbf{x}_i))}_{\text{loss function}}$$

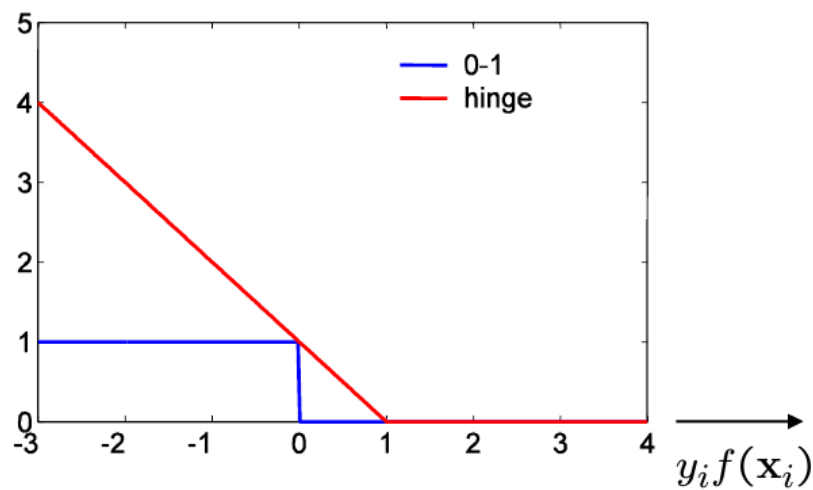
Points are in three categories:

1. $y_i f(\mathbf{x}_i) > 1$
Point is outside margin.
No contribution to loss.
2. $y_i f(\mathbf{x}_i) = 1$
Point is on margin.
No contribution to loss.
As in hard margin case.
3. $y_i f(\mathbf{x}_i) < 1$
Point violates margin constraint.
Contributes to loss



Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

SVM Optimization

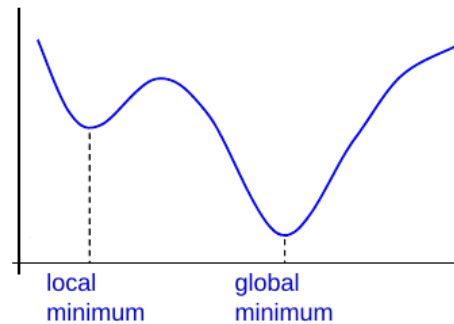


- SVM uses “hinge” loss $\max(0, 1 - y_i f(x_i))$
- an approximation to the 0-1 loss

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

SVM Optimization

$$\min_{\mathbf{w} \in \mathbb{R}^d} C \sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i)) + \|\mathbf{w}\|^2$$



- Does this cost function have a unique solution?
- Does the solution depend on the starting point of an iterative optimization algorithm (such as gradient descent)?

If the cost function is **convex**, then a locally optimal point is globally optimal (provided the optimization is over a convex set, which it is in our case)

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

SVM Optimization

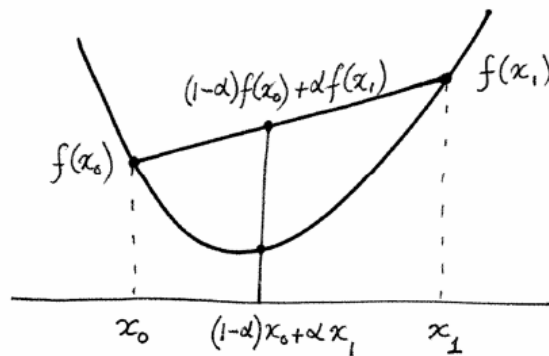
* Convex function

D – a domain in $\mathbb{R}^n$.

A **convex function** $f : D \rightarrow \mathbb{R}$ is one that satisfies, for any x_0 and x_1 in D :

$$f((1 - \alpha)x_0 + \alpha x_1) \leq (1 - \alpha)f(x_0) + \alpha f(x_1) \quad .$$

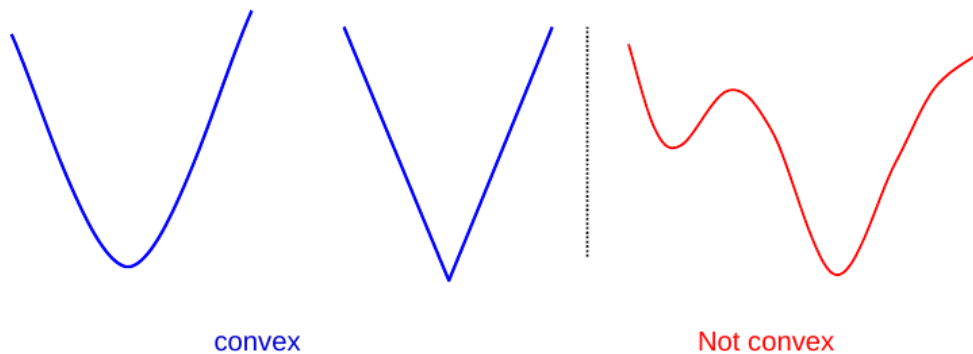
Line joining $(x_0, f(x_0))$
and $(x_1, f(x_1))$ lies
above the function graph.



Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

SVM Optimization

❁ Convex function



A non-negative sum of convex functions is convex

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

SVM Optimization

❁ Convex function



SVM

$$\min_{\mathbf{w} \in \mathbb{R}^d} C \sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i)) + \|\mathbf{w}\|^2 \quad \text{convex}$$

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

SVM Optimization

* Gradient descent algorithm for SVM

To minimize a cost function $\mathcal{C}(\mathbf{w})$ use the iterative update

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}} \mathcal{C}(\mathbf{w}_t)$$

where η is the learning rate.

First, rewrite the optimization problem as an **average**

$$\begin{aligned} \min_{\mathbf{w}} \mathcal{C}(\mathbf{w}) &= \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i)) \\ &= \frac{1}{N} \sum_i^N \left(\frac{\lambda}{2} \|\mathbf{w}\|^2 + \max(0, 1 - y_i f(\mathbf{x}_i)) \right) \end{aligned}$$

(with $\lambda = 2/(NC)$ up to an overall scale of the problem) and
 $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$

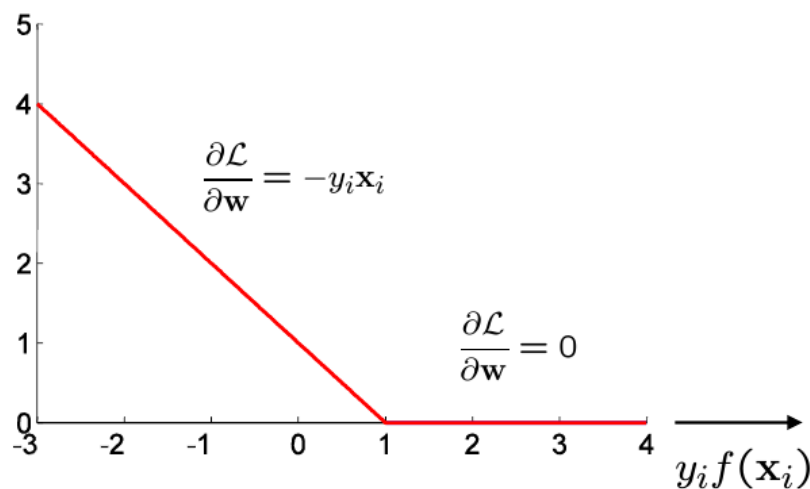
Because the hinge loss is not differentiable, a **sub-gradient** is computed

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

SVM Optimization

❁ Sub-gradient for hinge loss

$$\mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w}) = \max(0, 1 - y_i f(\mathbf{x}_i)) \quad f(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i + b$$



Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

SVM Optimization

❁ Sub-gradient descent algorithm for SVM

$$\mathcal{C}(\mathbf{w}) = \frac{1}{N} \sum_i^N \left(\frac{\lambda}{2} \|\mathbf{w}\|^2 + \mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w}) \right)$$

The iterative update is

$$\begin{aligned} \mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t - \eta \nabla_{\mathbf{w}_t} \mathcal{C}(\mathbf{w}_t) \\ &\leftarrow \mathbf{w}_t - \eta \frac{1}{N} \sum_i^N (\lambda \mathbf{w}_t + \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w}_t)) \end{aligned}$$

where η is the learning rate.

Then each iteration t involves cycling through the training data with the updates:

$$\begin{aligned} \mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t - \eta(\lambda \mathbf{w}_t - y_i \mathbf{x}_i) && \text{if } y_i f(\mathbf{x}_i) < 1 \\ &\leftarrow \mathbf{w}_t - \eta \lambda \mathbf{w}_t && \text{otherwise} \end{aligned}$$

In the Pegasos algorithm the learning rate is set at $\eta_t = \frac{1}{\lambda t}$

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

Primal-Dual SVM

The **Representer Theorem** states that the solution $\mathbf{w}$ can always be written as a linear combination of the training data:

$$\mathbf{w} = \sum_{j=1}^N \alpha_j y_j \mathbf{x}_j$$

Proof: **see example sheet** .

Now, substitute for $\mathbf{w}$ in $f(x) = \mathbf{w}^\top \mathbf{x} + b$

$$f(x) = \left(\sum_{j=1}^N \alpha_j y_j \mathbf{x}_j \right)^\top \mathbf{x} + b = \sum_{j=1}^N \alpha_j y_j (\mathbf{x}_j^\top \mathbf{x}) + b$$

and for $\mathbf{w}$ in the cost function $\min_{\mathbf{w}} \|\mathbf{w}\|^2$ subject to $y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \forall i$

$$\|\mathbf{w}\|^2 = \left\{ \sum_j \alpha_j y_j \mathbf{x}_j \right\}^\top \left\{ \sum_k \alpha_k y_k \mathbf{x}_k \right\} = \sum_{jk} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j^\top \mathbf{x}_k)$$

Hence, an equivalent optimization problem is over α_j

$$\min_{\alpha_j} \sum_{jk} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j^\top \mathbf{x}_k) \quad \text{subject to } y_i \left(\sum_{j=1}^N \alpha_j y_j (\mathbf{x}_j^\top \mathbf{x}_i) + b \right) \geq 1, \forall i$$

and a few more steps are required to complete the derivation.

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

Primal-Dual SVM

Primal version of classifier:

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

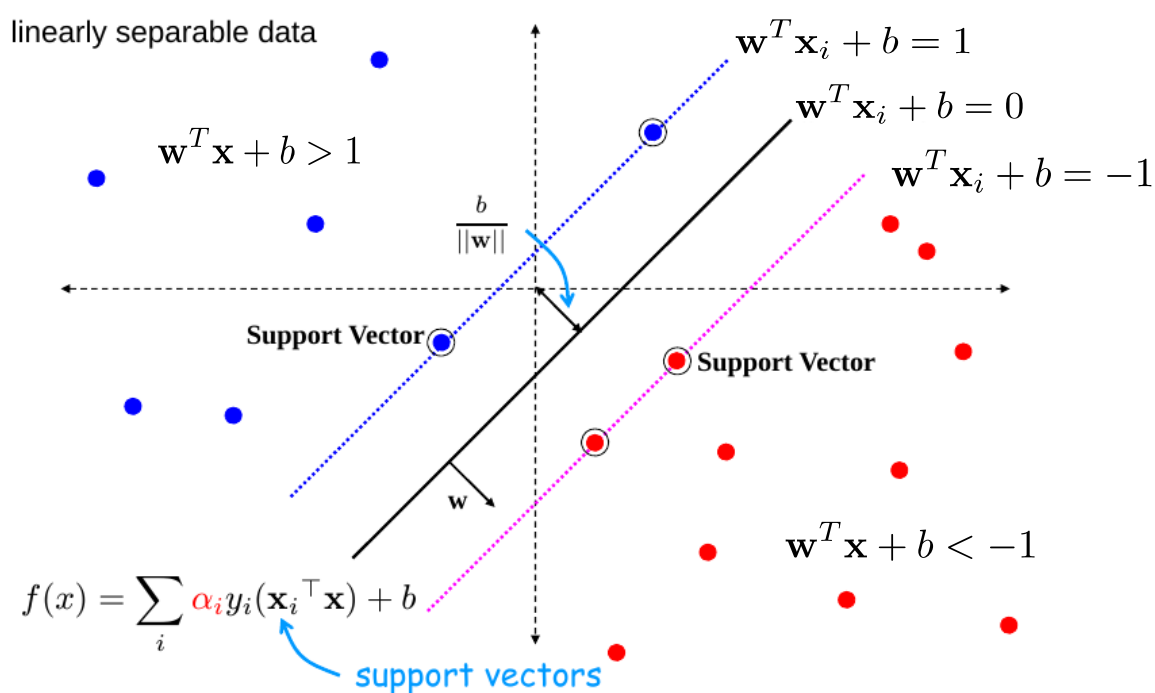
Dual version of classifier:

$$f(\mathbf{x}) = \sum_i^N \alpha_i y_i (\mathbf{x}_i^\top \mathbf{x}) + b$$

At first sight the dual form appears to have the disadvantage of a K-NN classifier – it requires the training data points $\mathbf{x}_i$. However, many of the α_i 's are zero. The ones that are non-zero define the support vectors $\mathbf{x}_i$.

Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

Primal-Dual SVM



Adopted from: Andrew Zisserman, Professor of Computer Science, Oxford University <https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>

Lagrangian Function

❁ Minimize Lagrangian primal

$$\mathcal{L}_{\mathcal{P}} = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i + \sum_{i=1}^m \alpha_i - \sum_{i=1}^m \alpha_i [y^{(i)} (w^T x^{(i)} + b) + \xi_i] - \sum_{i=1}^m r_i \xi_i$$

First, compute minimal w , b and ξ .

$$\nabla_w \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} = 0$$

Can solve for w ! $\rightarrow w = \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)}$

New constraint $\rightarrow \frac{\partial}{\partial b} \mathcal{L}(w, b, \alpha) = \sum_{i=1}^m \alpha_i y^{(i)} = 0$

$\frac{\partial}{\partial \xi} \mathcal{L}(w, b, \alpha) = C - \alpha_i - r_i = 0 \rightarrow \alpha_i = C - r_i$

$0 \leq \alpha_i \leq C$

New constraint

Substitute in

We use to solve for the weights w

Adopted from: Jerry Jeychandra, Blohm Lab

Lagrangian Function

❁ Maximize Lagrangian dual

$$\max_{\alpha} \mathcal{L}_{\mathcal{D}} = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y^{(i)} y^{(j)} \langle \mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)} \rangle$$

The Kernel Trick 

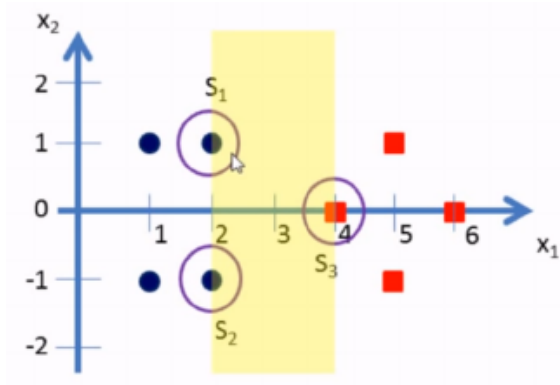
- S.T: $\forall i : \sum_{i=1}^m \alpha_i y^{(i)} = 0, \xi \geq 0, 0 \leq \alpha_i \leq C$ and KKT conditions
- Compute **inner product of $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$**
 - No longer rely on **w** or **b**
 - Can replace with $K(x,z)$ for **kernels**
- Can solve for **α explicitly**
 - All values not on functional margin have $\alpha = 0$, more efficient!

Karush-Khan-Tucker (KKT) conditions

Adopted from: Jerry Jeychandra, Blohm Lab

Example SVM – Φ is trivial

- * Φ is mapping function that transforms data in input space to data in feature space
- * Choose the support vectors S_1, S_2, S_3
 - + Find the separating hyperplane, $\mathbf{w}^T \mathbf{x}_i + b = 0$



$$S_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

$$S_2 = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$

$$S_3 = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$$

augmented with 1 as bias input

$$\begin{aligned} \tilde{S}_1 &= \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \\ \tilde{S}_2 &= \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \\ \tilde{S}_3 &= \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \end{aligned}$$

Adopted from: <http://scholastic tutors.webs.com>

Example SVM – Φ is trivial

To solve for optimal weights

$$w = \sum_i^m \alpha_i y^{(i)} x^{(i)}$$

Considering only the support vectors

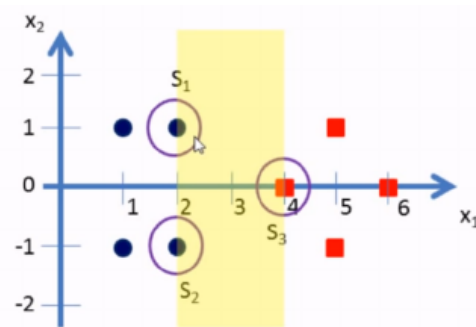
$$w = \sum_i^n \alpha_i s_i$$

$$w \cdot s_j = \pm 1 = \sum_i^n \alpha_i s_i \cdot s_j$$

Adopted from: <http://scholastic tutors.webs.com>

Example SVM – Φ is trivial

❁ Choose the support vectors S_1, S_2, S_3



Now we need to find 3 parameters α_1, α_2 , and α_3 based on the following 3 linear equations:

$$\alpha_1 \widetilde{S}_1 \cdot \widetilde{S}_1 + \alpha_2 \widetilde{S}_2 \cdot \widetilde{S}_1 + \alpha_3 \widetilde{S}_3 \cdot \widetilde{S}_1 = -1 \text{ (-ve class)}$$

$$\alpha_1 \widetilde{S}_1 \cdot \widetilde{S}_2 + \alpha_2 \widetilde{S}_2 \cdot \widetilde{S}_2 + \alpha_3 \widetilde{S}_3 \cdot \widetilde{S}_2 = -1 \text{ (-ve class)}$$

$$\alpha_1 \widetilde{S}_1 \cdot \widetilde{S}_3 + \alpha_2 \widetilde{S}_2 \cdot \widetilde{S}_3 + \alpha_3 \widetilde{S}_3 \cdot \widetilde{S}_3 = +1 \text{ (+ve class)}$$

Adopted from: <http://scholastic tutors.webs.com>

Example SVM – Φ is trivial

Let's substitute the values for $\tilde{S}_1$, $\tilde{S}_2$ and $\tilde{S}_3$ in the above equations.

$$\tilde{S}_1 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \quad \tilde{S}_2 = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \quad \tilde{S}_3 = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} = -1$$

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} = -1$$

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} = +1$$

- After simplification we get:

$$6\alpha_1 + 4\alpha_2 + 9\alpha_3 = -1$$

$$4\alpha_1 + 6\alpha_2 + 9\alpha_3 = -1$$

$$9\alpha_1 + 9\alpha_2 + 17\alpha_3 = +1$$

- Simplifying the above 3 simultaneous equations we get: $\alpha_1 = \alpha_2 = -3.25$ and $\alpha_3 = 3.5$.

Adopted from: <http://scholasticutors.webs.com>

Example SVM – Φ is trivial

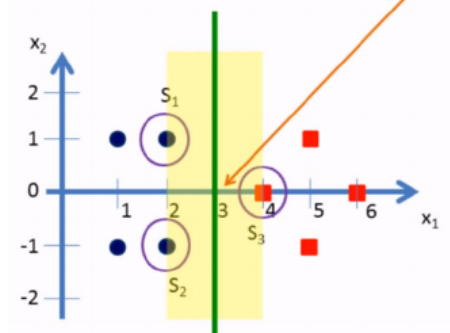
The hyper plane that discriminates the positive class from the negative class is give by:

$$\tilde{w} = \sum_i \alpha_i \tilde{S}_i$$

- Substituting the values we get:

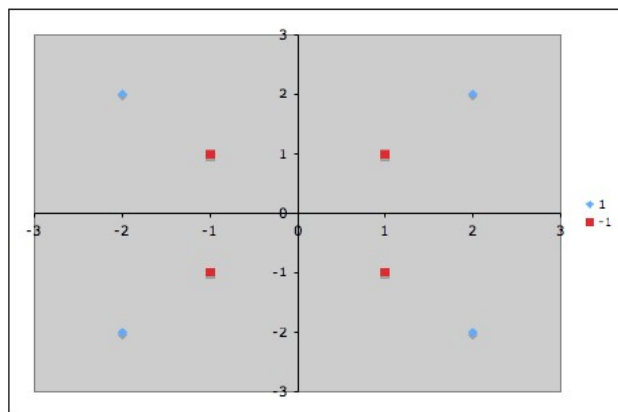
$$\begin{aligned}\tilde{w} &= \alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \\ \tilde{w} &= (-3.25) \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + (-3.25) \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + (3.5) \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -3 \end{pmatrix}\end{aligned}$$

- $y = wx + b$ with $w = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and offset $b = -3$.



Adopted from: <http://scholasticutors.webs.com>

Example SVM – Φ is non-trivial



Non-linear separable data points in R^2 .

Positive data, +1:

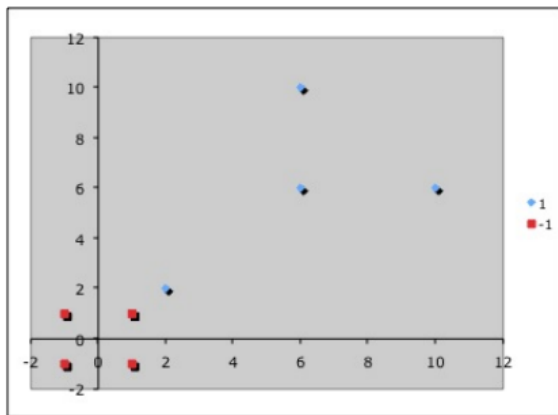
$$\left\{ \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ -2 \end{pmatrix}, \begin{pmatrix} -2 \\ -2 \end{pmatrix}, \begin{pmatrix} -2 \\ 2 \end{pmatrix} \right\}$$

Negative data, -1:

$$\left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix} \right\}$$

Adopted from: Dan Ventura, BYU – <http://axon.cs.byu.edu/Dan/>

Example SVM – Φ is non-trivial



Data separated in feature space

$$\Phi_1 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{cases} \begin{pmatrix} 4 - x_2 + |x_1 - x_2| \\ 4 - x_1 + |x_1 - x_2| \end{pmatrix} & \text{if } \sqrt{x_1^2 + x_2^2} > 2 \\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & \text{otherwise} \end{cases}$$

A. Input Space

Positive data, +1:

$$\left\{ \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ -2 \end{pmatrix}, \begin{pmatrix} -2 \\ -2 \end{pmatrix}, \begin{pmatrix} -2 \\ 2 \end{pmatrix} \right\}$$

Negative data, -1:

$$\left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix} \right\}$$

B. Feature Space

Positive data, +1:

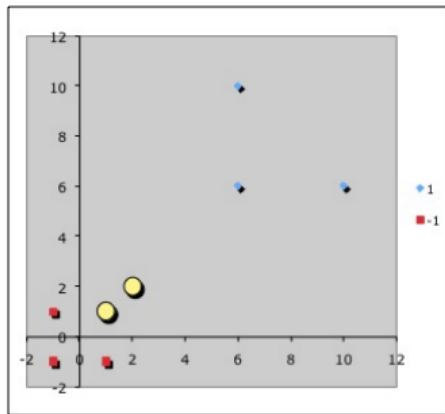
$$\left\{ \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 6 \\ 6 \end{pmatrix}, \begin{pmatrix} 2 \\ 6 \end{pmatrix} \right\}$$

Negative data, -1:

$$\left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix} \right\}$$

Adopted from: Dan Ventura, BYU – <http://axon.cs.byu.edu/Dan/>

Example SVM – Φ is non-trivial



Data separated in feature space
with support vectors in yellow

A. Feature Space

Positive data, +1:

$$\left\{ \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 6 \\ 6 \end{pmatrix}, \begin{pmatrix} 2 \\ 6 \end{pmatrix} \right\}$$

Negative data, -1:

$$\left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix} \right\}$$

B. Support Vectors

$$\left\{ s_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, s_2 = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \right\}$$

$$\alpha_1 \Phi_1(s_1) \cdot \Phi_1(s_1) + \alpha_2 \Phi_1(s_2) \cdot \Phi_1(s_1) = -1$$

$$\alpha_1 \Phi_1(s_1) \cdot \Phi_1(s_2) + \alpha_2 \Phi_1(s_2) \cdot \Phi_1(s_2) = +1$$

$$\alpha_1 \tilde{s}_1 \cdot \tilde{s}_1 + \alpha_2 \tilde{s}_2 \cdot \tilde{s}_1 = -1$$

$$\alpha_1 \tilde{s}_1 \cdot \tilde{s}_2 + \alpha_2 \tilde{s}_2 \cdot \tilde{s}_2 = +1$$

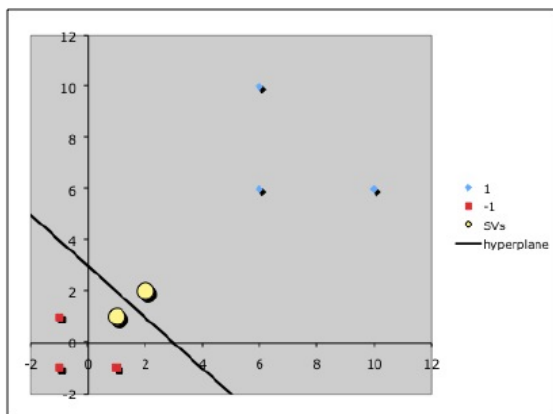
$$3\alpha_1 + 5\alpha_2 = -1$$

$$5\alpha_1 + 9\alpha_2 = +1$$

$$\alpha_1 = -7 \text{ and } \alpha_2 = 4.$$

Adopted from: Dan Ventura, BYU – <http://axon.cs.byu.edu/Dan/>

Example SVM – Φ is non-trivial



$$\begin{aligned}\tilde{w} &= \sum_i \alpha_i \tilde{s}_i \\ &= -7 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + 4 \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ 1 \\ -3 \end{pmatrix}\end{aligned}$$

$$w = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad b = -3.$$

Adopted from: Dan Ventura, BYU – <http://axon.cs.byu.edu/Dan/>

Classifying through SVM

$$f(x) = \sigma \left(\sum_i \alpha_i \Phi(s_i) \cdot \Phi(x) \right)$$

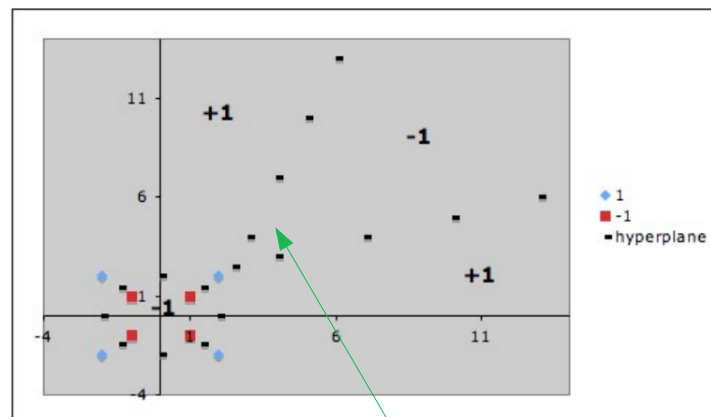
where $\sigma(z)$ returns the sign of z

we want to classify the point $x = (4, 5)$

$$\begin{aligned} f \begin{pmatrix} 4 \\ 5 \end{pmatrix} &= \sigma \left(-7\Phi_1 \begin{pmatrix} 1 \\ 1 \end{pmatrix} \cdot \Phi_1 \begin{pmatrix} 4 \\ 5 \end{pmatrix} + 4\Phi_1 \begin{pmatrix} 2 \\ 2 \end{pmatrix} \cdot \Phi_1 \begin{pmatrix} 4 \\ 5 \end{pmatrix} \right) \\ &= \sigma \left(-7 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} + 4 \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \right) \\ &= \sigma(-2) \end{aligned}$$

Adopted from: Dan Ventura, BYU – <http://axon.cs.byu.edu/Dan/>

Classifying through SVM



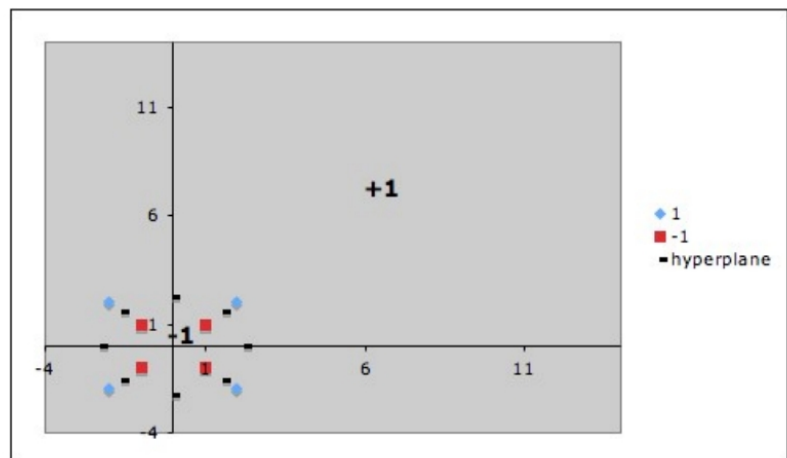
The decision surface in input space corresponding to Φ_1 . Note the singularity. We would classify $x = (4, 5)$ as negative.

Adopted from: Dan Ventura, BYU – <http://axon.cs.byu.edu/Dan/>

The Kernel Trick

Our Φ_1 preserved the number of dimensions.
However, it is often that to effectively separate data,
the feature space is higher dimension from input space.

$$\Phi_2 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \frac{(x_1^2 + x_2^2) - 5}{3} \end{pmatrix}$$



The decision surface in the input space corresponding to Φ_2 .

Adopted from: Dan Ventura, BYU – <http://axon.cs.byu.edu/Dan/>

The Kernel Trick

$$\Phi_2 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \frac{(x_1^2 + x_2^2) - 5}{3} \end{pmatrix}$$

A. Feature Space

Positive data, +1:

$$\left\{ \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ -2 \\ 1 \end{pmatrix}, \begin{pmatrix} -2 \\ -2 \\ 1 \end{pmatrix}, \begin{pmatrix} -2 \\ 2 \\ 1 \end{pmatrix} \right\} \quad \left| \quad \left\{ \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 6 \\ 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 6 \\ 6 \\ 6 \end{pmatrix}, \begin{pmatrix} 2 \\ 2 \\ 6 \end{pmatrix} \right\}$$

Negative data, -1:

$$\left\{ \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \\ -1 \end{pmatrix} \right\} \quad \left| \quad \left\{ \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix} \right\}$$

B. Support Vectors

All vectors are support
vectors

$$\left\{ s_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, s_2 = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \right\}$$

Adopted from: Dan Ventura, BYU – <http://axon.cs.byu.edu/Dan/>

The Kernel Trick

$$\begin{aligned} &\alpha_1 \Phi_2(s_1) \cdot \Phi_2(s_{i=1to4}) + \alpha_2 \Phi_2(s_2) \cdot \Phi_2(s_{i=1to4}) + \\ &\alpha_3 \Phi_2(s_3) \cdot \Phi_2(s_{i=1to4}) + \alpha_4 \Phi_2(s_4) \cdot \Phi_2(s_{i=1to4}) + \\ &\alpha_5 \Phi_2(s_5) \cdot \Phi_2(s_{i=1to4}) + \alpha_6 \Phi_2(s_6) \cdot \Phi_2(s_{i=1to4}) + \\ &\alpha_7 \Phi_2(s_7) \cdot \Phi_2(s_{i=1to4}) + \alpha_8 \Phi_2(s_8) \cdot \Phi_2(s_{i=1to4}) = +1 \end{aligned}$$

$$\begin{aligned} &\alpha_1 \Phi_2(s_1) \cdot \Phi_2(s_{i=5to8}) + \alpha_2 \Phi_2(s_2) \cdot \Phi_2(s_{i=5to8}) + \\ &\alpha_3 \Phi_2(s_3) \cdot \Phi_2(s_{i=5to8}) + \alpha_4 \Phi_2(s_4) \cdot \Phi_2(s_{i=5to8}) + \\ &\alpha_5 \Phi_2(s_5) \cdot \Phi_2(s_{i=5to8}) + \alpha_6 \Phi_2(s_6) \cdot \Phi_2(s_{i=5to8}) + \\ &\alpha_7 \Phi_2(s_7) \cdot \Phi_2(s_{i=5to8}) + \alpha_8 \Phi_2(s_8) \cdot \Phi_2(s_{i=5to8}) = -1 \end{aligned}$$

$$\begin{aligned} &+9\alpha_1 + 1\alpha_2 - 7\alpha_3 + 1\alpha_4 + 3\alpha_5 - 1\alpha_6 - 5\alpha_7 - 1\alpha_8 = +1 \\ &+1\alpha_1 + 9\alpha_2 + 1\alpha_3 - 7\alpha_4 - 1\alpha_5 + 3\alpha_6 - 1\alpha_7 - 5\alpha_8 = +1 \\ &-7\alpha_1 + 1\alpha_2 + 9\alpha_3 + 1\alpha_4 - 5\alpha_5 - 1\alpha_6 + 3\alpha_7 - 1\alpha_8 = +1 \\ &+1\alpha_1 - 7\alpha_2 + 1\alpha_3 + 9\alpha_4 - 1\alpha_5 - 5\alpha_6 - 1\alpha_7 + 3\alpha_8 = +1 \\ &+3\alpha_1 - 1\alpha_2 - 5\alpha_3 - 1\alpha_4 + 3\alpha_5 + 1\alpha_6 - 1\alpha_7 + 1\alpha_8 = -1 \\ &-1\alpha_1 + 3\alpha_2 - 1\alpha_3 - 5\alpha_4 + 1\alpha_5 + 3\alpha_6 + 1\alpha_7 - 1\alpha_8 = -1 \\ &-5\alpha_1 - 1\alpha_2 + 3\alpha_3 - 1\alpha_4 - 1\alpha_5 + 1\alpha_6 + 3\alpha_7 + 1\alpha_8 = -1 \\ &-1\alpha_1 - 5\alpha_2 - 1\alpha_3 + 3\alpha_4 + 1\alpha_5 + 1\alpha_6 + 1\alpha_7 + 3\alpha_8 = -1 \end{aligned}$$

8 x 8 matrix is not invertible.

By taking two support vectors at a time, one positive and one negative classification,

$$\alpha_{i(+1)} = \frac{1}{3}, \quad \alpha_{i(-1)} = -\frac{2}{3}$$

$$\tilde{w} = \sum_{i=1}^8 \alpha_i \tilde{s}_i = \begin{pmatrix} 0 \\ 0 \\ 4 \end{pmatrix}$$

The discriminating feature is x3

$$f(x) = \sigma(x_3)$$

Adopted from: Dan Ventura, BYU – <http://axon.cs.byu.edu/Dan/>

MCTE 512 - ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING FOR ENGINEERS

BY RODRIGO S. JAMISOLA JR.

BOTSWANA INTERNATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY, PALAPYE, BOTSWANA

LECTURE SLIDES ADOPTED FROM *MACHINE LEARNING* ©TOM M. MITCHELL, MCGRAW HILL, 1997 WITH SOME MODIFICATION
30 NOVEMBER 2020

ARTIFICIAL NEURAL NETWORKS

- Threshold units
- Gradient descent
- Multilayer networks
- Backpropagation
- Hidden layer representations

THE NUERON

Biological Neuron

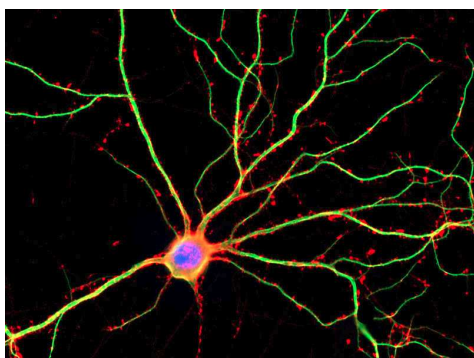
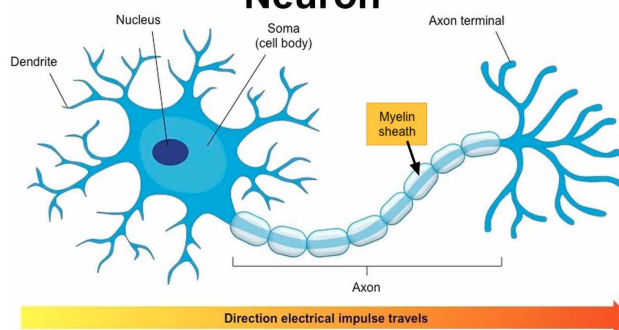


Diagram of a Neuron



CONNECTIONIST MODELS

Consider humans:

- Neuron switching time $\sim .001$ second
 - Number of neurons $\sim 10^{10}$
 - Connections per neuron $\sim 10^{4-5}$
 - Scene recognition time $\sim .1$ second
 - 100 inference steps doesn't seem like enough
- much parallel computation

Properties of artificial neural nets (ANN's):

- Many neuron-like threshold switching units
- Many weighted interconnections among units
- Highly parallel, distributed process
- Emphasis on tuning weights automatically

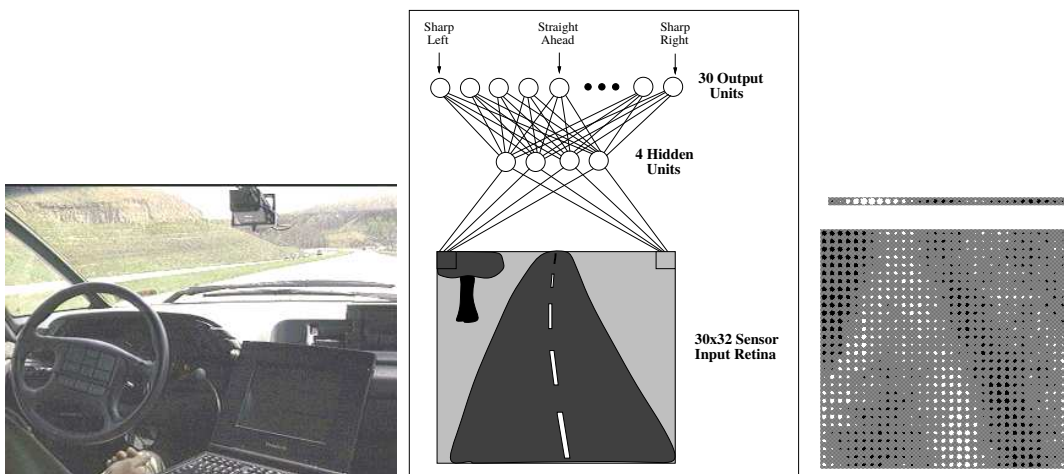
WHEN TO CONSIDER NEURAL NETWORKS

- Input is high-dimensional discrete or real-valued (e.g. raw sensor input)
- Output is discrete or real valued
- Output is a vector of values
- Possibly noisy data
- Form of target function is unknown
- Human readability of result is unimportant

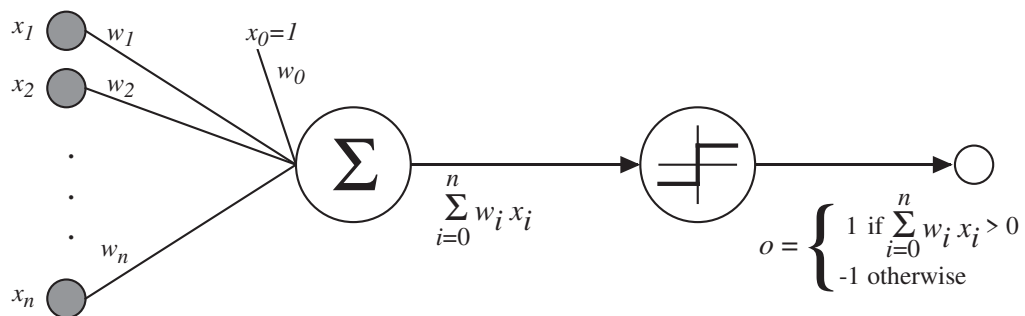
Examples:

- Speech phoneme recognition [Waibel]
- Image classification [Kanade, Baluja, Rowley]
- Financial prediction
- Self-driving cars
- Video frames image classification

ALVINN DRIVES 70 MPH ON HIGHWAYS



PERCEPTRON

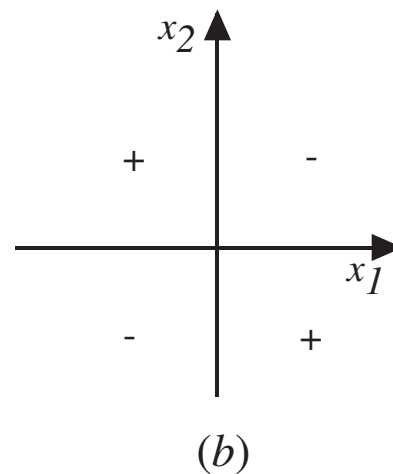
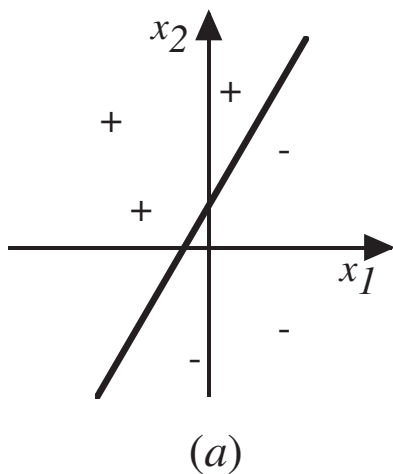


$$o(x_1, \dots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \dots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Sometimes we'll use simpler vector notation:

$$o(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ -1 & \text{otherwise.} \end{cases}$$

DECISION SURFACE OF A PERCEPTRON



Represents some useful functions

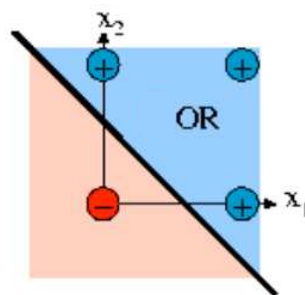
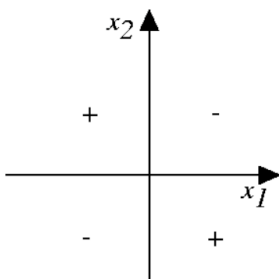
- What weights represent $g(x_1, x_2) = AND(x_1, x_2)$?

But some functions not representable

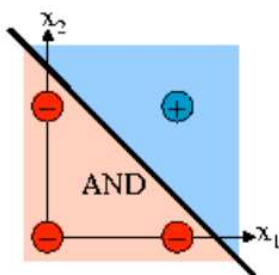
- e.g., not linearly separable
- Therefore, we'll want networks of these...

CONCEPT SPACE AND LINEAR SEPARABILITY

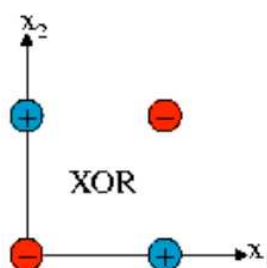
Linearly Separable



Linearly Separable



Not Linearly Separable



PERCEPTRON TRAINING RULE

$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = \eta(t - o)x_i$$

Where:

- $t = c(\vec{x})$ is target value
- o is perceptron output
- η is small constant (e.g., .1) called *learning rate*

Can prove it will converge

- If training data is linearly separable
- and η sufficiently small

GRADIENT DESCENT

To understand, consider simpler *linear unit*, where

$$o = w_0 + w_1x_1 + \cdots + w_nx_n$$

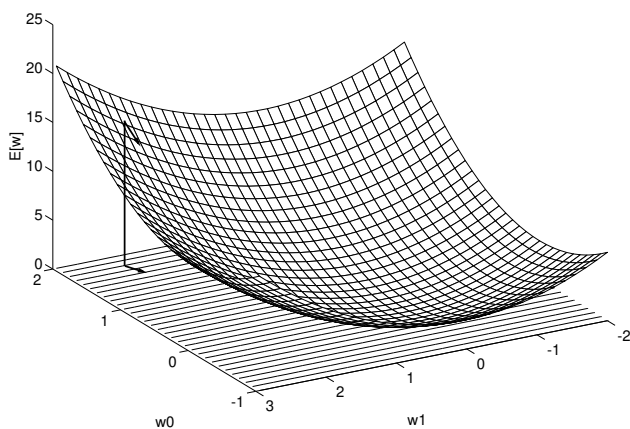
Let's learn w_i 's that minimize the squared error

$$E[\vec{W}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

Where D is set of training examples

GRADIENT DESCENT

Gradient



$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

GRADIENT DESCENT

$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_d (t_d - o_d)^2 \\&= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\&= \frac{1}{2} \sum_d 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\&= \sum_d (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x}_d) \\ \frac{\partial E}{\partial w_i} &= \sum_d (t_d - o_d)(-x_{i,d})\end{aligned}$$

GRADIENT DESCENT

GRADIENT-DESCENT (*training_examples*, η)

Each training example is a pair of the form $\langle \vec{x}, t \rangle$ where

- $\vec{x}$ is the vector of input values
- t is the target output value, and
- η is the learning rate (e.g., .05)

1. Initialize each w_i to some small random value
2. Until the termination condition is met, Do
 - 2.1 Initialize each Δw_i to zero.
 - 2.2 For each $\langle \vec{x}, t \rangle$ in *training_examples*, Do
 - 2.2.1 Input the instance $\vec{x}$ to the unit and compute the output o
 - 2.2.2 For each linear unit weight w_i , Do

$$\Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i$$

- 2.3 For each linear unit weight w_i , Do

$$w_i \leftarrow w_i + \Delta w_i$$

SUMMARY

Perceptron training rule guaranteed to succeed if

- Training examples are linearly separable
- Sufficiently small learning rate η

Linear unit training rule uses gradient descent

- Guaranteed to converge to hypothesis with minimum squared error
- Given sufficiently small learning rate η
- Even when training data contains noise
- Even when training data not separable by H

INCREMENTAL (STOCHASTIC) GRADIENT DESCENT

Batch mode Gradient Descent:

Do until satisfied

1. Compute the gradient $\nabla E_D[\vec{W}]$
2. $\vec{W} \leftarrow \vec{W} - \eta \nabla E_D[\vec{W}]$

Incremental mode Gradient Descent:

Do until satisfied

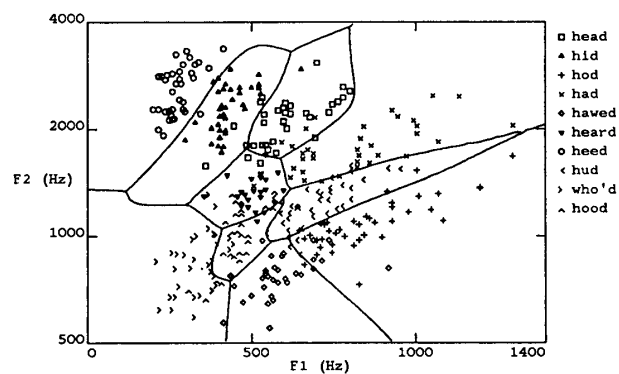
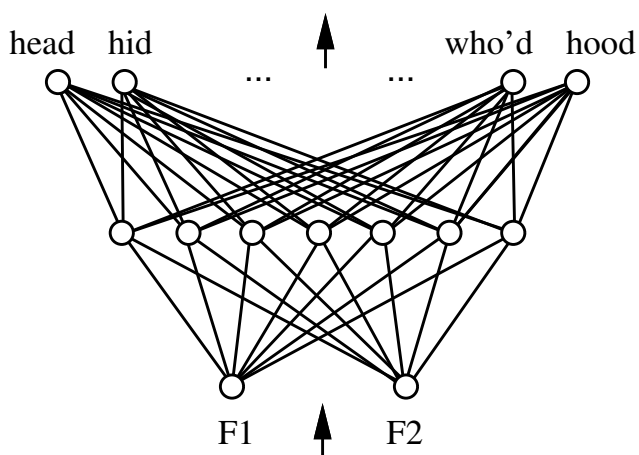
- For each training example d in D
 1. Compute the gradient $\nabla E_d[\vec{W}]$
 2. $\vec{W} \leftarrow \vec{W} - \eta \nabla E_d[\vec{W}]$

$$E_D[\vec{W}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

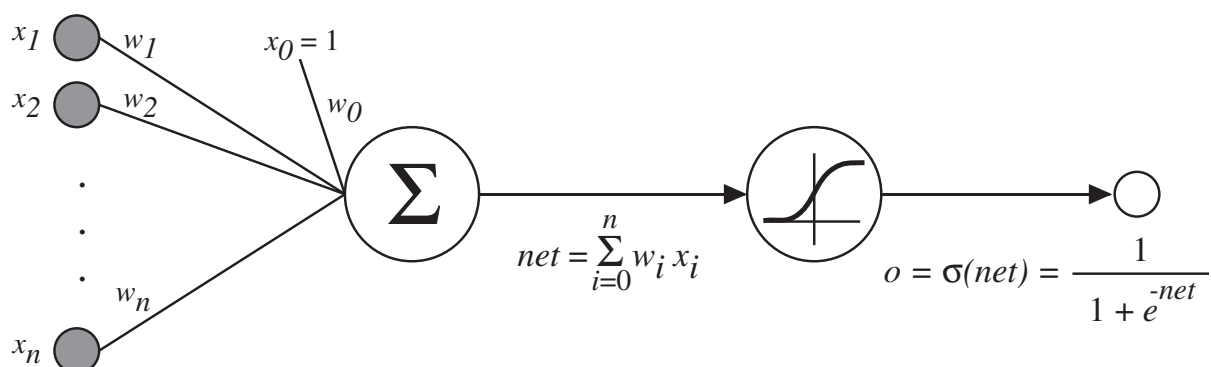
$$E_d[\vec{W}] \equiv \frac{1}{2} (t_d - o_d)^2$$

Incremental Gradient Descent can approximate *Batch Gradient Descent* arbitrarily closely if η made small enough

MULTILAYER NETWORKS OF SIGMOID UNITS



SIGMOID UNIT



$\sigma(x)$ is the sigmoid function

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

Nice property: $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$

We can derive gradient decent rules to train

- One sigmoid unit
- *Multilayer networks* of sigmoid units → Backpropagation

ERROR GRADIENT FOR A SIGMOID UNIT

$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \\&= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\&= \frac{1}{2} \sum_d 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\&= \sum_d (t_d - o_d) \left(-\frac{\partial o_d}{\partial w_i} \right) \\&= - \sum_d (t_d - o_d) \frac{\partial o_d}{\partial net_d} \frac{\partial net_d}{\partial w_i}\end{aligned}$$

ERROR GRADIENT FOR A SIGMOID UNIT

But we know:

$$\frac{\partial o_d}{\partial net_d} = \frac{\partial \sigma(net_d)}{\partial net_d} = o_d(1 - o_d)$$

$$\frac{\partial net_d}{\partial w_i} = \frac{\partial (\vec{w} \cdot \vec{x}_d)}{\partial w_i} = x_{i,d}$$

So:

$$\frac{\partial E}{\partial w_i} = - \sum_{d \in D} (t_d - o_d) o_d (1 - o_d) x_{i,d}$$

BACKPROPAGATION ALGORITHM

1. Initialize all weights to small random numbers.
2. Until satisfied, Do
 - For each training example, Do
 - 2.1 Input the training example to the network and compute the network outputs
 - 2.2 For each output unit k

$$\delta_k \leftarrow y_k(1 - y_k)(t_k - y_k)$$

- 2.3 For each hidden unit h

$$\delta_j \leftarrow z_j(1 - z_j) \sum_{k \in \text{outputs}} w_{kj} \delta_k$$

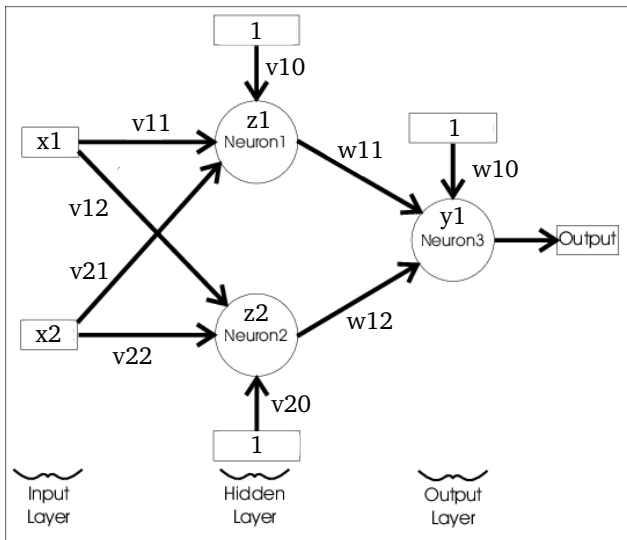
- 2.4 Update each network weight w_{kj}

$$w_{kj} \leftarrow w_{kj} + \Delta w_{kj}$$

- 2.5 Update each network weight v_{ji}

$$v_{ji} \leftarrow v_{ji} + \Delta v_{ji}$$

BACKPROPAGATION EQUATIONS



$$z_1 = \frac{1}{1 + e^{-v_1^T x}}, \quad z_2 = \frac{1}{1 + e^{-v_2^T x}}$$

$$y_1 = \frac{1}{1 + e^{-w_1^T x}}$$

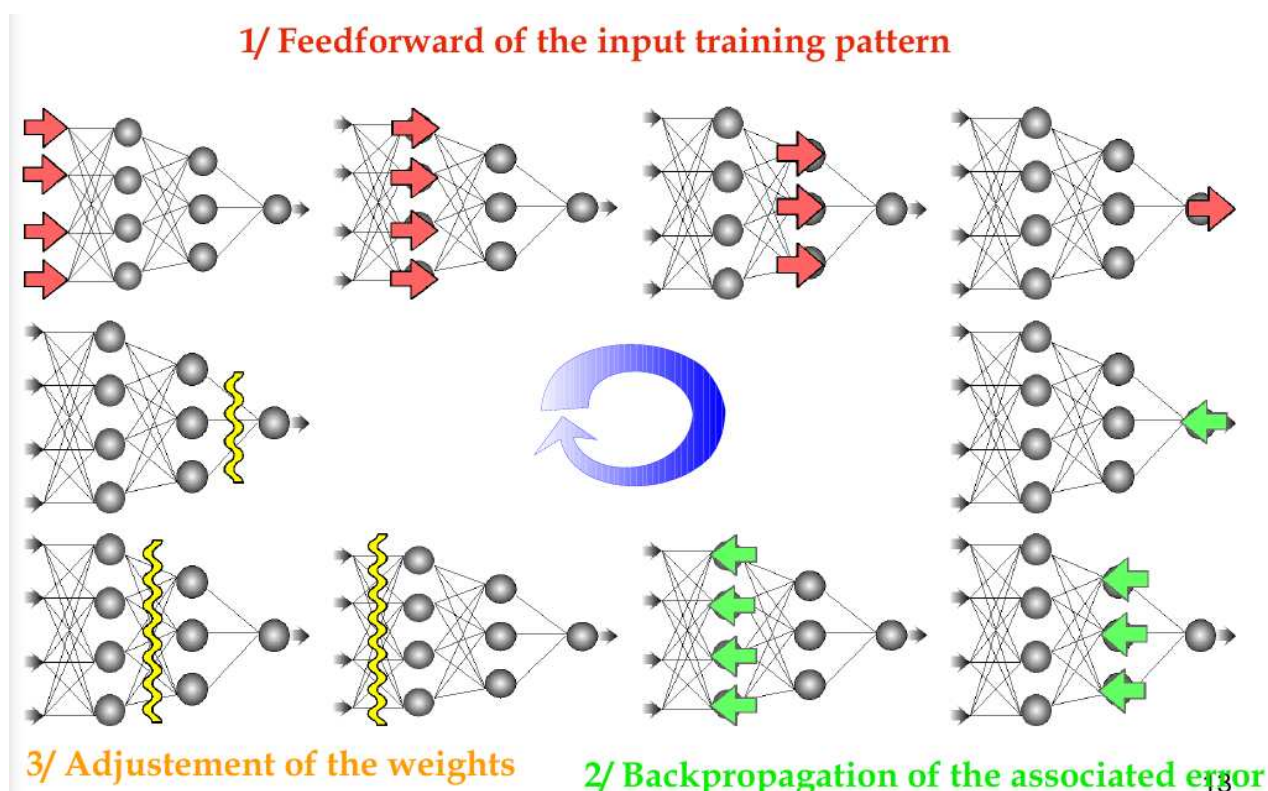
$$\begin{aligned} w_{kj} &\leftarrow w_{kj} + \Delta w_{kj} \\ &\leftarrow w_{kj} + \alpha \delta_k z_j \\ &\leftarrow w_{kj} + \alpha y_k (1 - y_k) (t_k - y_k) z_j \end{aligned}$$

$$\begin{aligned} v_{ji} &\leftarrow v_{ji} + \Delta v_{ji} \\ &\leftarrow v_{ji} + \alpha \delta_j x_i \\ &\leftarrow v_{ji} + \alpha z_j (1 - z_j) \sum \delta_k w_{kj} x_i \\ &\leftarrow v_{ji} + \alpha z_j (1 - z_j) \sum y_k (1 - y_k) (t_k - y_k) w_{kj} x_i \end{aligned}$$

BACKPROPAGATION TRAINING UPDATE

- x : input training vector
- t : output target vector
- δ_k : error correction of w_{kj} corresponding to y_k
- δ_j : error correction of v_{ji} corresponding to z_j
- α : learning rate
- v_{j0} : weight for bias of hidden unit j
- w_{k0} : weight for bias of hidden unit k
- i : index for input data
- j : index for hidden layer
- k : index for output layer

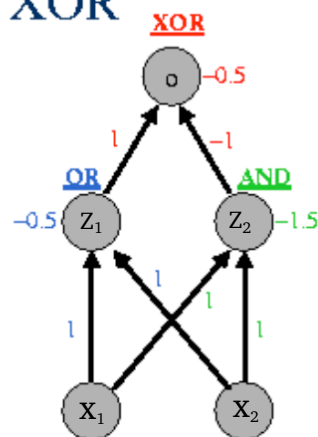
BACKPROPAGATION TRAINING UPDATE



XOR EXAMPLE

Boolean XOR

input x1	input x2	output
0	0	0
0	1	1
1	0	1
1	1	0



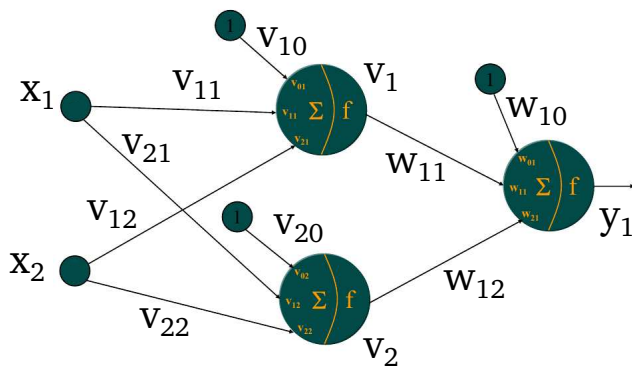
$$z_1 = \frac{1}{1 + e^{-v_1^T x}}, \quad z_2 = \frac{1}{1 + e^{-v_2^T x}}$$

$$y_1 = \frac{1}{1 + e^{-w_1^T x}}$$

$$\begin{aligned} w_{kj} &\leftarrow w_{kj} + \Delta w_{kj} \\ &\leftarrow w_{kj} + \alpha \delta_k z_j \\ &\leftarrow w_{kj} + \alpha y_k (1 - y_k) (t_k - y_k) z_j \end{aligned}$$

$$\begin{aligned} v_{ji} &\leftarrow v_{ji} + \Delta v_{ji} \\ &\leftarrow v_{ji} + \alpha \delta_j x_i \\ &\leftarrow v_{ji} + \alpha z_j (1 - z_j) \sum \delta_k w_{kj} x_i \\ &\leftarrow v_{ji} + \alpha z_j (1 - z_j) \sum y_k (1 - y_k) (t_k - y_k) w_{kj} x_i \end{aligned}$$

XOR EXAMPLE



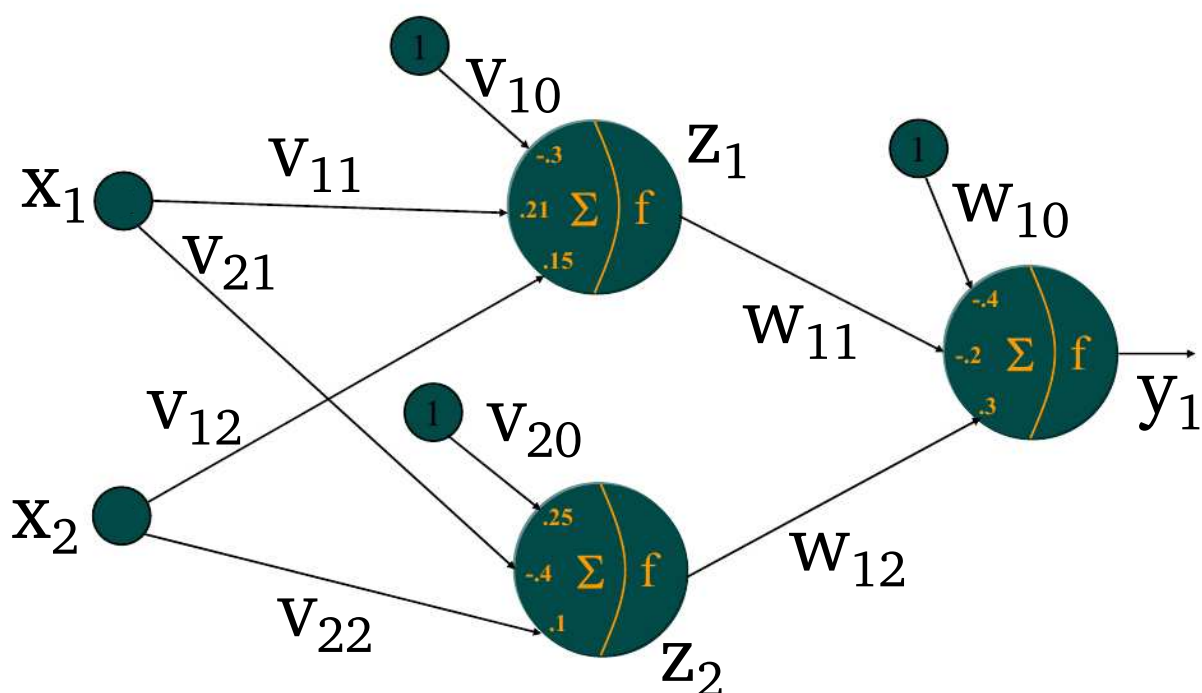
1. Initialize all weights to small random numbers.
2. For each training example, Do
 - Input the training example to the network and compute the network outputs
 - For each output unit k

$$w_{kj} \leftarrow w_{kj} + \alpha y_k (1 - y_k) (t_k - y_k) z_j$$

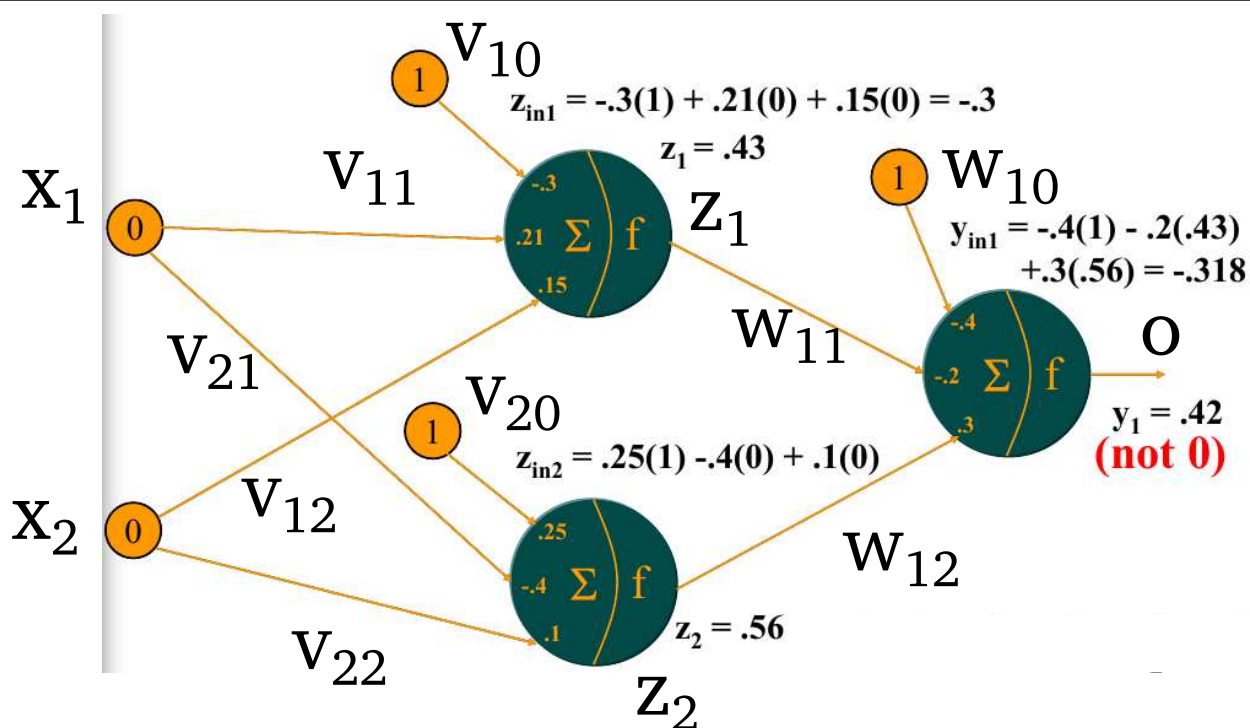
$$v_{ji} \leftarrow v_{ji} + \alpha z_j (1 - z_j) \sum y_k (1 - y_k) (t_k - y_k) w_{kj} x_i$$

XOR EXAMPLE - STEP 1

Initialize weights to some small random values.



XOR EXAMPLE - STEP 2 FEEDFORWARD



$$z_{in1} = v_1^T X \quad \left| \quad y_{in1} = w^T z \quad \left| \quad z_1 = \frac{1}{1 + e^{-z_{in1}}} \quad \left| \quad y_1 = \frac{1}{1 + e^{-y_{in1}}}$$

XOR EXAMPLE - STEP 2 FEEDFORWARD

Weights vectors

$$v_1 = [v_{10}, v_{11}, v_{12}]^T, v_2 = [v_{20}, v_{21}, v_{22}]^T,$$

$$w_1 = [w_{10}, w_{11}, w_{12}]^T$$

Table of random initial values:

Weight	Value
v_{10}	-0.3
v_{11}	0.21
v_{12}	0.15
v_{20}	0.25
v_{21}	-0.4
v_{22}	0.1
w_{10}	-0.4
w_{11}	-0.2
w_{12}	0.3

Table of Data

x_1	x_2	Output
0	0	0
0	1	1
1	0	1
1	1	0

$$\begin{aligned}
 z_{in1} &= v_1^T x \\
 &= v_{10}(1) + v_{11}(x_1) + v_{12}(x_2) \\
 &= -0.3(1) + 0.21(0) + 0.15(0) \\
 &= -0.3
 \end{aligned}$$

$$\begin{aligned}
 z_1 &= \frac{1}{1+e^{-z_{in1}}} \\
 &= \frac{1}{1+e^{-(-0.3)}} \\
 &= 0.43
 \end{aligned}$$

XOR EXAMPLE - STEP 2 FEEDFORWARD

$$\begin{aligned}z_{in2} &= v_2^T X \\&= v_{20}(1) + v_{21}(x_1) + v_{22}(x_2) \\&= 0.25(1) + -0.4(0) + 0.1(0) \\&= 0.25\end{aligned}$$

$$\begin{aligned}z_2 &= \frac{1}{1+e^{-z_{in2}}} \\&= \frac{1}{1+e^{-0.25}} \\&= 0.56\end{aligned}$$

$$\begin{aligned}y_1 &= \frac{1}{1+e^{-y_{in1}}} \\&= \frac{1}{1+e^{0.318}} \\&= 0.42\end{aligned}$$

$$\begin{aligned}y_{in1} &= w_1^T X \\&= w_{10}(1) + w_{11}(z_1) + w_{12}(z_2) \\&= -0.4(1) + -0.2(0.43) + 0.3(0.56) \\&= -0.318\end{aligned}$$

XOR EXAMPLE - STEP 3 BACKPROPAGATE

Output Layer with $k = 1$:

$$\delta_k = y_k(1 - y_k)(t_k - y_k)$$

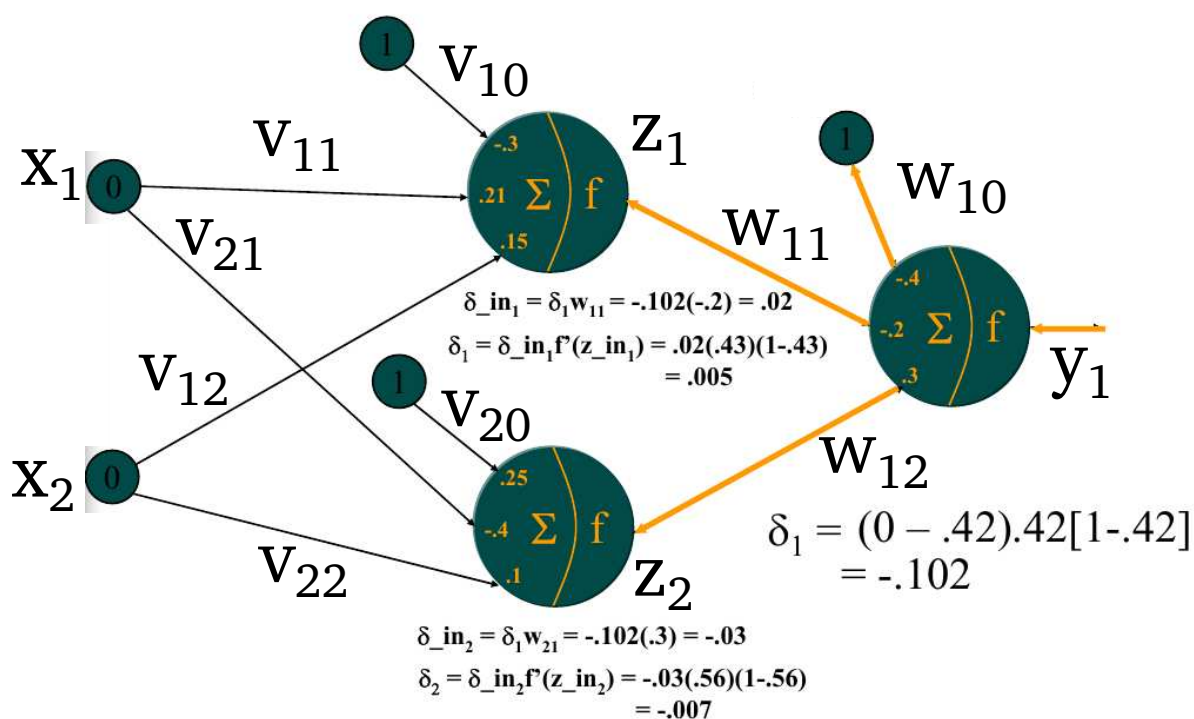
Update the weights of the output with $j = 1, 2$:

$$\begin{aligned} w_{kj} &\leftarrow w_{kj} + \alpha \delta_k z_j \\ &\leftarrow w_{kj} + \alpha y_k(1 - y_k)(t_k - y_k) z_j \end{aligned}$$

At the hidden layer (we remove the $\sum$ to get incremental mode) with $i = 1, \dots, 4$:

$$\begin{aligned} v_{ji} &\leftarrow v_{ji} + \alpha \delta_j x_i \\ &\leftarrow v_{ji} + \alpha z_j(1 - z_j) \delta_k w_{kj} x_i \\ &\leftarrow v_{ji} + \alpha z_j(1 - z_j) y_k(1 - y_k)(t_k - y_k) w_{kj} x_i \end{aligned}$$

XOR EXAMPLE - STEP 3 BACKPROPAGATE δ_k, δ_j



XOR EXAMPLE - STEP 3 BACKPROPAGATE δ_k, δ_j

Output Layer $\delta_k, k = 1$:

$$\delta_k = y_k(1 - y_k)(t_k - y_k)$$

$$\begin{aligned}\delta_1 &= y_1(1 - y_1)(t_1 - y_1) \\ &= 0.42(1 - 0.42)(0 - 0.42) \\ &= -0.102\end{aligned}$$

Hidden Layer $\delta_j, j = 1$:

$$\delta_j = z_j(1 - z_j)\delta_k w_{kj}$$

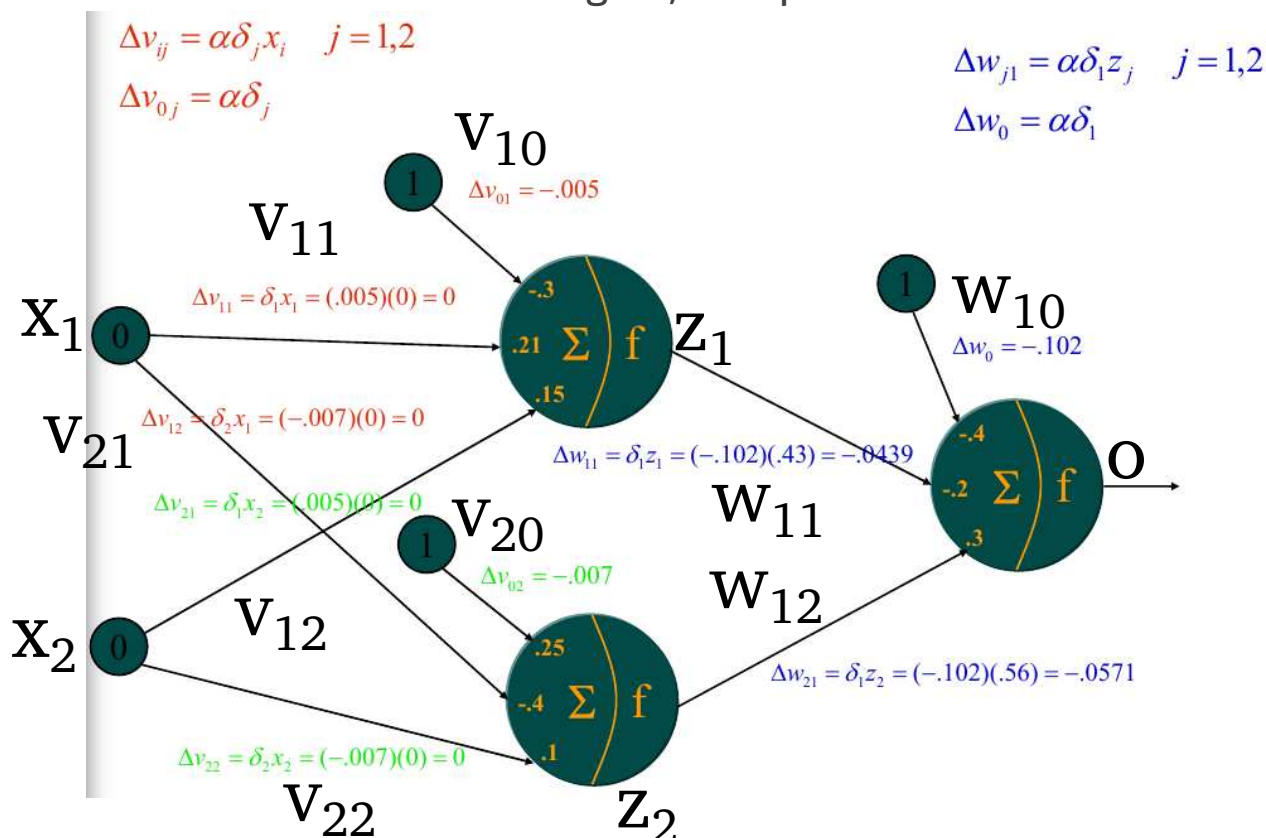
$$\begin{aligned}\delta_1 &= z_1(1 - z_1)\delta_1 w_{11} \\ &= 0.43(1 - 0.43)(-0.102)(-0.2) \\ &= 0.005\end{aligned}$$

Hidden Layer $\delta_j, j = 2$:

$$\begin{aligned}\delta_2 &= z_2(1 - z_2)\delta_1 w_{12} \\ &= 0.56(1 - 0.56)(-0.102)(0.3) \\ &= -0.007\end{aligned}$$

XOR EXAMPLE - STEP 3 BACKPROPAGATE ΔW_{kj} , ΔV_{ji}

Calculate the incremental weights, first pass:



XOR EXAMPLE - STEP 3 BACKPROPAGATE ΔW_{kj}

Output Layer $k = 1$, $\alpha = 1$

$$\Delta W_{kj} = \alpha \delta_k z_j$$

■ $j = 0$

$$\begin{aligned}\Delta W_{10} &= \alpha \delta_1 z_0 \\ &= 1(-0.102)(1) \\ &= -0.102\end{aligned}$$

■ $j = 1$

$$\begin{aligned}\Delta W_{11} &= \alpha \delta_1 z_1 \\ &= 1(-0.102)(0.43) \\ &= -0.0439\end{aligned}$$

■ $j = 2$

$$\begin{aligned}\Delta W_{12} &= \alpha \delta_1 z_2 \\ &= 1(-0.102)(0.56) \\ &= -0.0571\end{aligned}$$

XOR EXAMPLE - STEP 3 BACKPROPAGATE Δv_{ji}

Hidden Layer $j = 1, 2$

$$\Delta v_{ji} = \alpha \delta_j x_i$$

■ $j = 2$

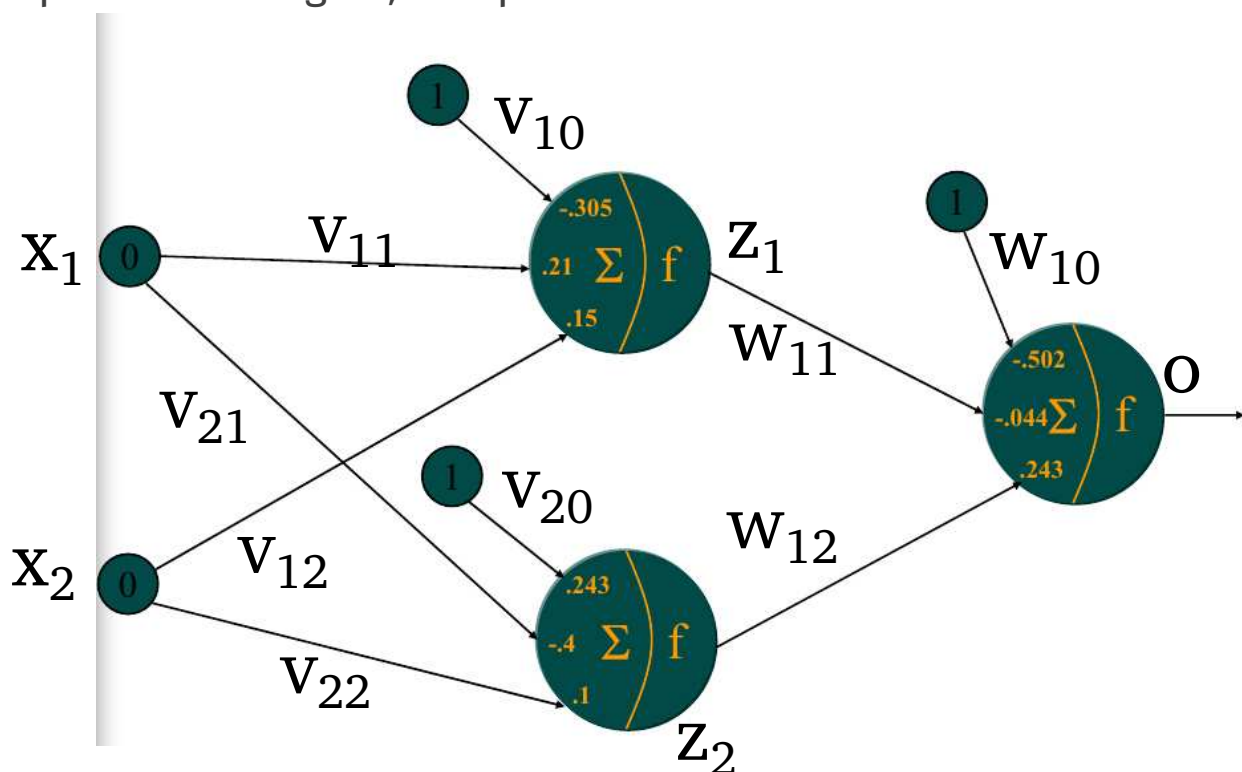
■ $j = 1$

$$\begin{aligned}\Delta v_{10} &= \alpha \delta_1 x_0 \\ &= (1)(0.005)(1) \\ &= 0.005 \text{ (different)} \\ \Delta v_{11} &= \alpha \delta_1 x_1 \\ &= (1)(0.005)(0) \\ &= 0 \\ \Delta v_{12} &= \alpha \delta_1 x_2 \\ &= (1)(0.005)(0) \\ &= 0\end{aligned}$$

$$\begin{aligned}\Delta v_{20} &= \alpha \delta_2 x_0 \\ &= (1)(-0.007)(1) \\ &= -0.007 \\ \Delta v_{21} &= \alpha \delta_2 x_1 \\ &= (1)(-0.007)(0) \\ &= 0 \\ \Delta v_{22} &= \alpha \delta_2 x_2 \\ &= (1)(-0.007)(0) \\ &= 0\end{aligned}$$

XOR EXAMPLE - STEP 4 UPDATE WEIGHTS

Update the weights, first pass:



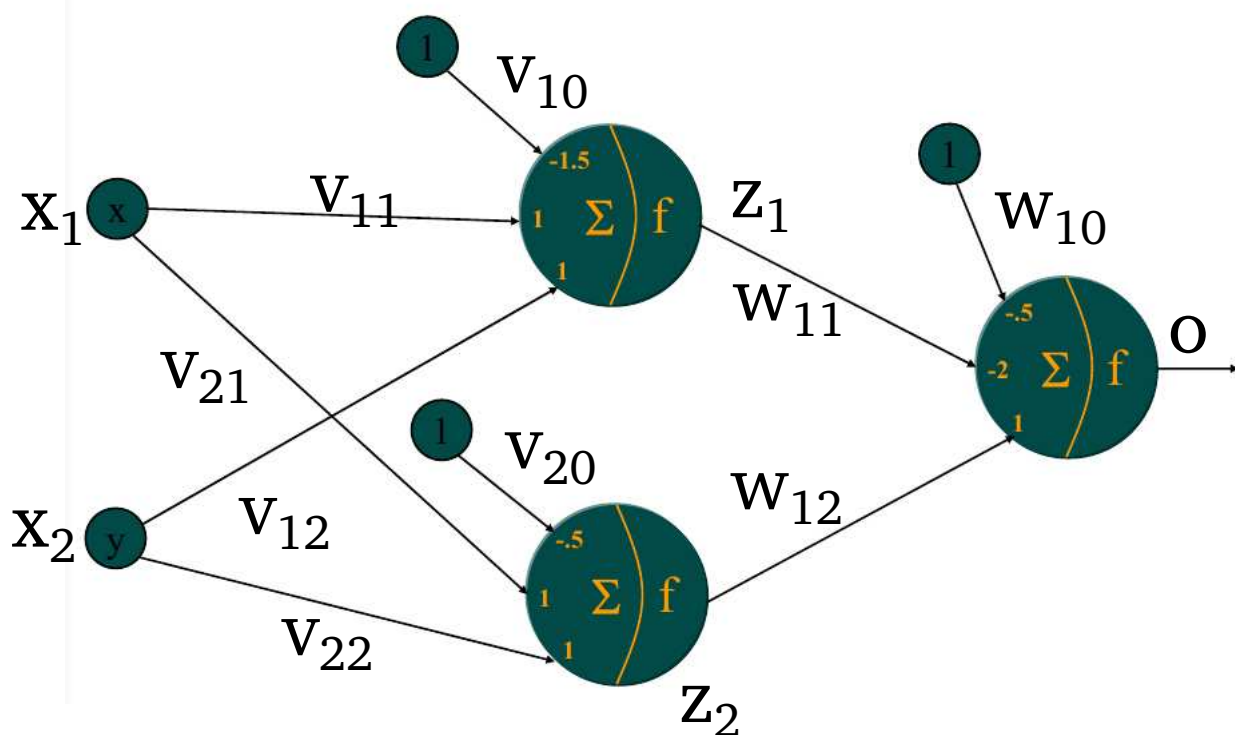
XOR EXAMPLE - STEP 4 UPDATE WEIGHTS

Update the weights, first pass:

$$\begin{aligned}
 & \blacksquare v_{ji}, j = 1, 2 \\
 & \quad \blacktriangleright j = 1 \\
 & \blacksquare w_{kj}, k = 1 \\
 & \quad w_{10} = w_{10} + \Delta w_{10} \\
 & \quad \quad = -0.4 + (-0.102) \\
 & \quad \quad = -0.502 \\
 & \quad w_{11} = w_{11} + \Delta w_{11} \\
 & \quad \quad = -0.2 + -0.0439 \\
 & \quad \quad = -0.243 \text{ (different)} \\
 & \quad w_{12} = w_{12} + \Delta w_{12} \\
 & \quad \quad = 0.3 + -0.0571 \\
 & \quad \quad = 0.243 \\
 & \quad v_{10} = v_{10} + \Delta v_{10} \\
 & \quad \quad = -0.3 + 0.005 = -0.295 \\
 & \quad \quad \quad \text{(different)} \\
 & \quad v_{11} = v_{11} + 0. \Delta v_{11} \\
 & \quad \quad = 0.21 + 0.0 = 0.21 \\
 & \quad v_{12} = v_{12} + \Delta v_{12} \\
 & \quad \quad = 0.15 + 0.0 = 0.15 \\
 & \quad \blacktriangleright j = 2 \\
 & \quad v_{20} = v_{20} + \Delta v_{20} \\
 & \quad \quad = 0.25 + (-0.007) = 0.243 \\
 & \quad v_{21} = v_{21} + 0. \Delta v_{21} \\
 & \quad \quad = -0.4 + 0.0 = -0.4 \\
 & \quad v_{22} = v_{22} + \Delta v_{22} \\
 & \quad \quad = 0.1 + 0.0 = 0.1
 \end{aligned}$$

XOR EXAMPLE - STEP 5 MAXIMUM ITERATIONS

After about 500 iterations:



Abstract

This book is a collection of all laboratory exercises for the course Artificial Intelligence and Machine Learning for Engineers, a fifth-year level course for the program Bachelor of Engineer in Mechatronics and Industrial Instrumentation in Botswana International University of Science and Technology, Palapye, Botswana. The exercises were designed to enhance the understanding of the students to complement the topics discussed in the lectures. Most of the exercises require the students to perform manual computations in order to enhance their theoretical understanding. This book includes the course syllabus as well as lecture slides for the entire semester. Interested students can attend virtual class discussions uploaded on YouTube via the channel of the author.

About the Author



Rodrigo S. Jamisola, Jr. received his B.S. degree in Mechanical Engineering from the University of the Philippines-Diliman, M.E. degree (research-based) in Mechanical Engineering from the National University of Singapore in 2001, M.Sc. degree in Electrical and Computer Engineering from Colorado State University in 2006, and Ph.D. degree in Electronics and Communications Engineering from De La Salle University-Manila in 2009. He joined De La Salle University as an Asst. Professor in 2008 and Toyota Motor Philippines as R&D Manager in 2011. He was a Post-doctoral Research Fellow at Daegu-Gyeongbuk Institute of Science and Technology in South Korea, and was a Sr. Postdoctoral Fellow at Italian Institute of Technology (IIT) in Genova, Italy. He is now an Associate Professor at Botswana International University of Science and Technology in Palapye, Botswana. His current research projects include Development of Drones for Surveillance and Delivery, Machine Learning Modelling of Addiction, and Exoskeleton for Enhanced Mobility and Rehabilitation.

ISBN: 978-99968-935-0-6