Machine Learning
DSECL ZG565

Dr. Chetana Gavankar, Ph.D,
IIT Bombay-Monash University Australia
Chetana.gavankar@pilani.bits-pilani.ac.in

BITS Pilani
Pilani Campus

innovate achieve lead

BITS Pilani

Pilani Campus

Session 1
Date – 19th October 2019
Time – 9 am to 11 am

These slides are prepared by the instructor, with grateful acknowledgement of Prof. Sugato Ghosal, Prof. S.K. H. Islam from BITS Pilani and many others who made their course materials freely available online.

Session Content

- Objective of course
- Evaluation Plan
- What is Machine Learning?
- Application areas of Machine Learning
- Why Machine Learning is important?
- Design a Learning System
- Issues in Machine Learning

Objective of course

- Introduction to the basic concepts and techniques of Machine Learning
- Gain experience of doing independent study and research in the field of Machine Learning
- Develop skills of using recent machine learning software tools to evaluate learning algorithms and model selection for solving practical problems

What We'll Cover in this Course

- **Supervised learning**
  - Regression
  - Logistic regression
  - Bayesian learning
  - Decision Tree (Random Forest)
  - Neural networks
  - Support vector machines
- **Unsupervised learning**
  - Clustering
  - Dimensionality reduction
- **Applications**
- **Ensemble Learning**

Books

Text books and Reference book(s)

T1	Tom M. Mitchell: Machine Learning, The McGraw-Hill Companies, Inc. International Edition 1997
T2	Christopher M. Bhisop: Pattern Recognition & Machine Learning, Springer, 2006

R1	**C.J.C. BURGES: A Tutorial on Support Vector Machines for Pattern Recognition, Kluwer Academic Publishers, Boston.**

Evaluation Plan

Name	Type	Duration	Weight
Quiz-I (roughly after 6 sessions)	Online	2-Dec-2019 (Before midsem)	5%
Assignment-I (roughly after 3 sessions)	Take Home	15-Nov-2019 (Before midsem)	13%
Assignment-II	Take Home	31-Jan-2019 (after midsem)	12%
Mid-Semester Test (after 7 sessions)	Closed Book	1.5 Hrs	30%
Comprehensive Exam	Open Book	2.5 Hrs	40%

Please note there will be no change in submission dates for quiz and assignment

Lab Plan

Lab No.	Lab Objective
1	Linear Regression and Gradient Descent
2	Logistic Regression classifier
3	Single layer Back propagation NN
4	K-nearest Neighbour
5	SVM
6	K-means clustering

- **Labs not graded**
- **Webinars will be conducted for lab sessions**
- **Labs will be conducted in Python**

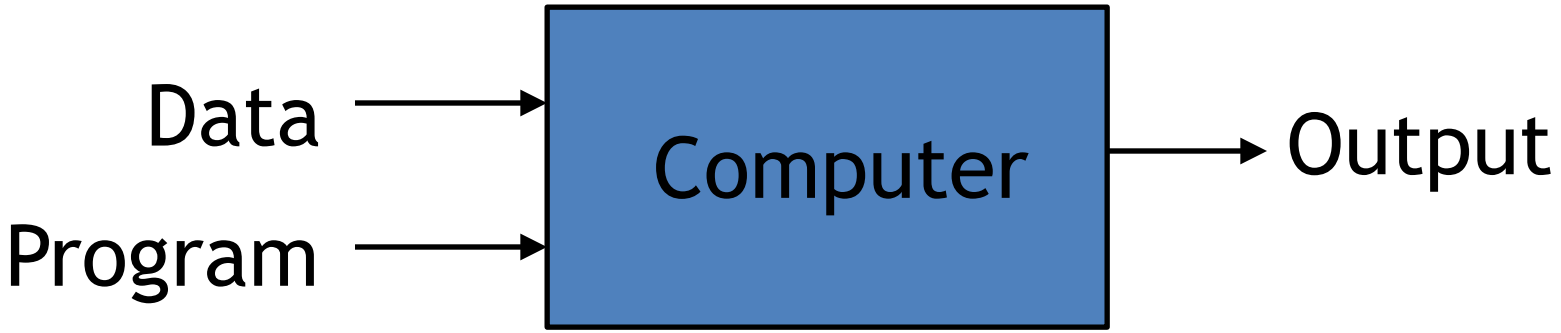
Machine Learning

- **Machine learning** is a scientific discipline that explores the construction and study of algorithms that can learn from data.
- Such algorithms operate by building a model based on inputs and using that to make predictions or decisions, rather than following only explicitly programmed instructions.

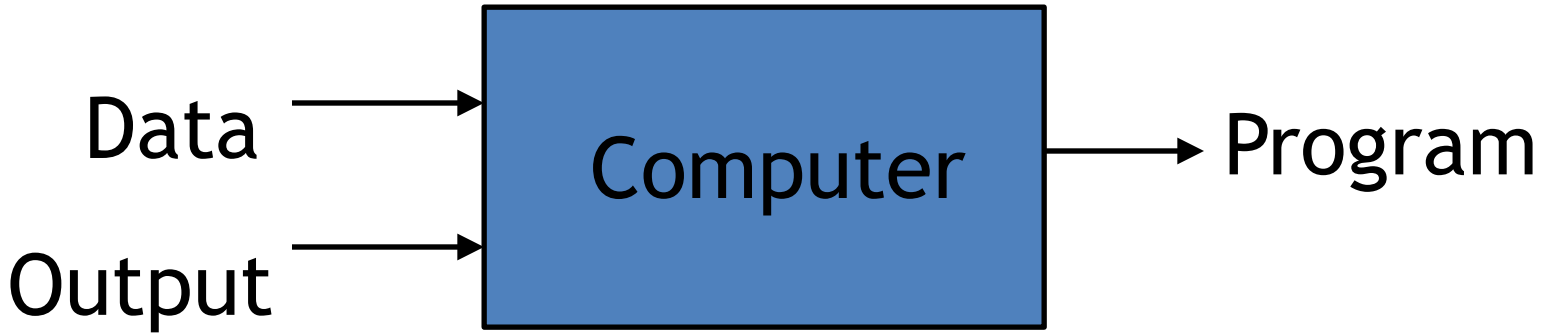
A Few Quotes

- “A breakthrough in machine learning would be worth ten Microsofts” (Bill Gates, Chairman, Microsoft)
- “Machine learning is the next Internet” (Tony Tether, Director, DARPA)
- “Web rankings today are mostly a matter of machine learning” (Prabhakar Raghavan, Dir. Research, Yahoo)
- “Machine learning is going to result in a real revolution” (Greg Papadopoulos, CTO, Sun)
- “Machine learning is today’s discontinuity” (Jerry Yang, CEO, Yahoo)

Traditional Programming



Machine Learning



Slide credit: Pedro Domingos

What is Machine Learning?

Definition by Tom Mitchell (1998):

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." Example: playing checkers.

E = the experience of playing many games of checkers

T = the task of playing checkers.

P = the probability that the program will win the next game.

What is Machine Learning?

- To have a learning problem, we must identify
  - The class of tasks
  - The measure of performance to be improved
  - Source of experience

Example of Learning Problems

A Checker Learning Problem

- **Task T:** Playing Checkers
- **Performance Measure P:** Percent of games won against opponents
- **Training Experience E:** To be selected ==> Games Played against itself

A handwriting recognition learning problem

- **Task T:** recognizing and classifying handwritten words within images
- **Performance measure P**: percent of words correctly classified
- **Training Experience E:** a database of handwritten words with given classifications

A robot driving learning problem

- **Task T:** driving on public four-lane highways using vision sensors
- **Performance measure P:** average distance travelled before an error (as judged by human)
- **Training experience E:** a sequence of images and steering commands recorded while observing a human driver

Why is Machine Learning Important?

- Some tasks cannot be defined well, except by examples.
- Relationships and correlations can be hidden within large amounts of data. Machine Learning may be able to find these relationships.
- Human designers often produce machines that do not work as well as desired in the environments in which they are used.

Why is Machine Learning Important ?

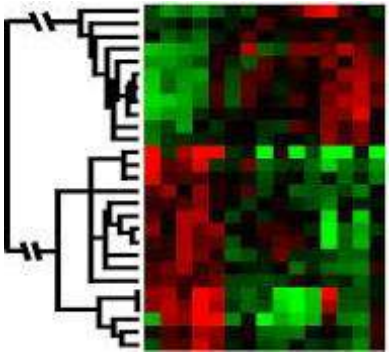
- The amount of knowledge available about certain tasks might be too large for explicit encoding by humans (e.g., medical diagnostic).
- New knowledge about tasks is constantly being discovered by humans. It may be difficult to continuously re-design systems “by hand”.

When Do We Use Machine Learning?

ML is used when:

- Human expertise does not exist (navigating on Mars)
- Humans can't explain their expertise (speech recognition)
- Models must be customized (personalized medicine)
- Models are based on huge amounts of data (genomics)



Learning isn't always useful:

- There is no need to "learn" to calculate payroll

Slide Credit: Eric Eaton

Applications of Machine Learning

- Learning to recognize spoken words (Lee, 1989; Waibel, 1989).
- Detect fraudulent use of credit cards or Learning to drive an autonomous vehicle (Pomerleau, 1989).
- Learning to classify new astronomical structures (Fayyad et al., 1995).

Applications of Machine Learning

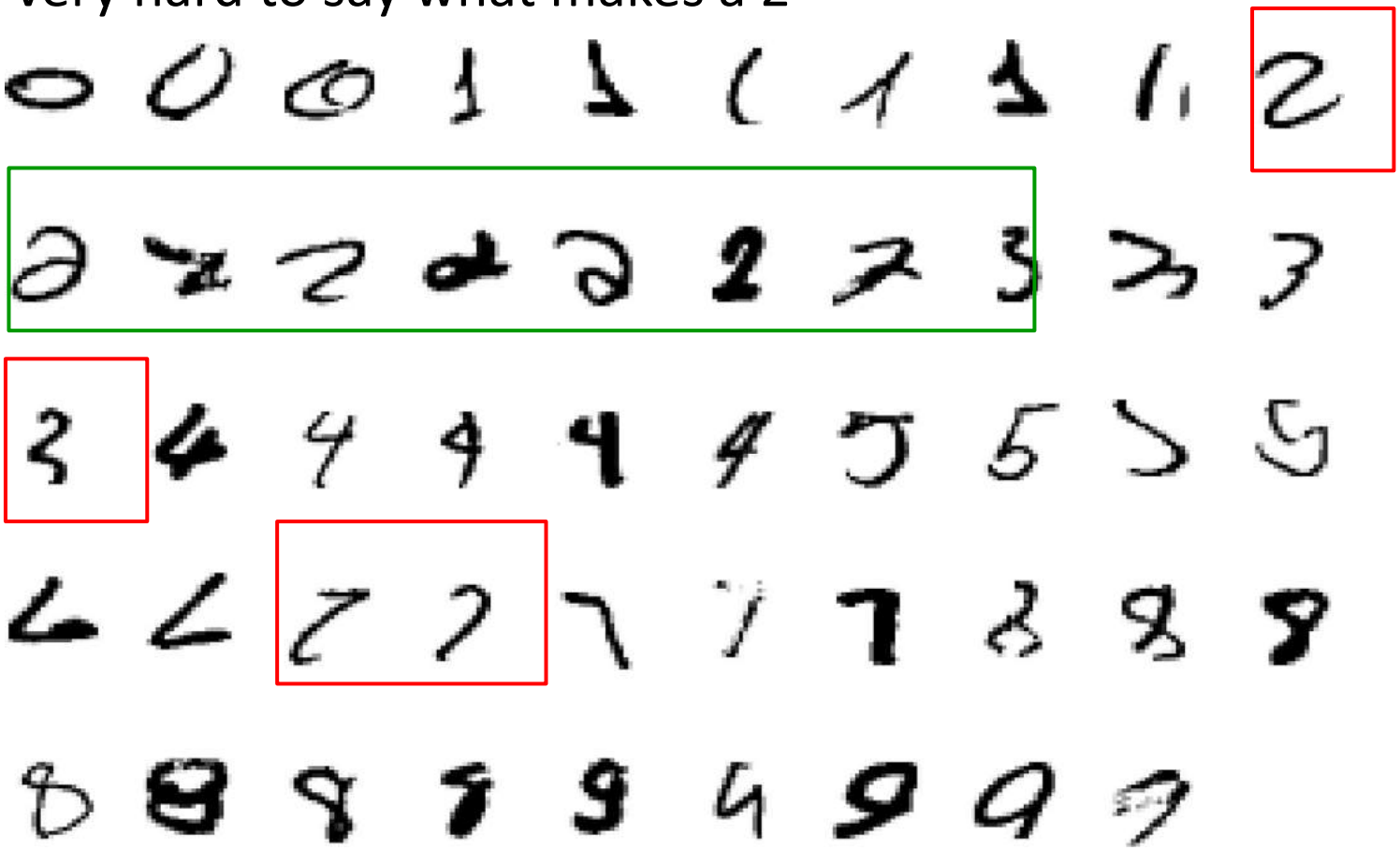
- Learning to play world-class backgammon (Tesauro 1992, 1995).
- Predict recovery rates of pneumonia patients (Copper et al. 1997)

Application Types: Classification

- Medical diagnosis
- Credit card applications or transactions
- Fraud detection in e-commerce
- Worm detection in network packets
- Spam filtering in email
- Recommended articles in a newspaper
- Recommended books, movies, music, or jokes
- Financial investments
- DNA sequences
- Spoken words
- Handwritten letters
- Astronomical images

Pattern recognition

It is very hard to say what makes a 2



Slide credit: Geoffrey Hinton

Application Domains

- Web search
- Computational biology
- Finance
- E-commerce
- Space exploration
- Robotics
- Information extraction
- Social networks
- Language Processing

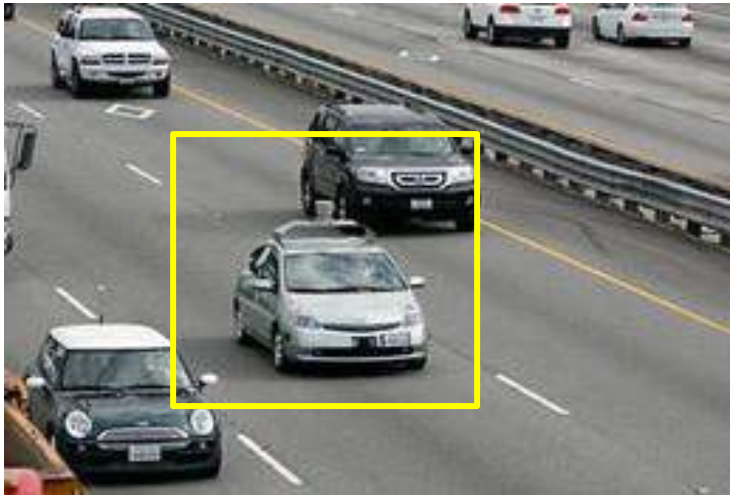
Many more emerging…

Slide credit: Pedro Domingos

State of the Art Applications of Machine Learning

In this course, you will learn principles that will help you understand and build some of these applications.

Autonomous Cars

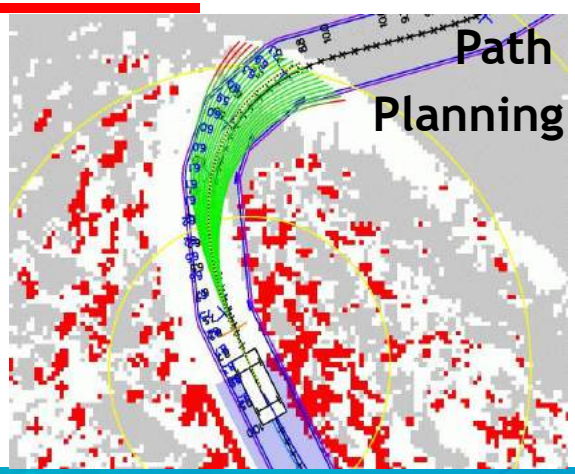


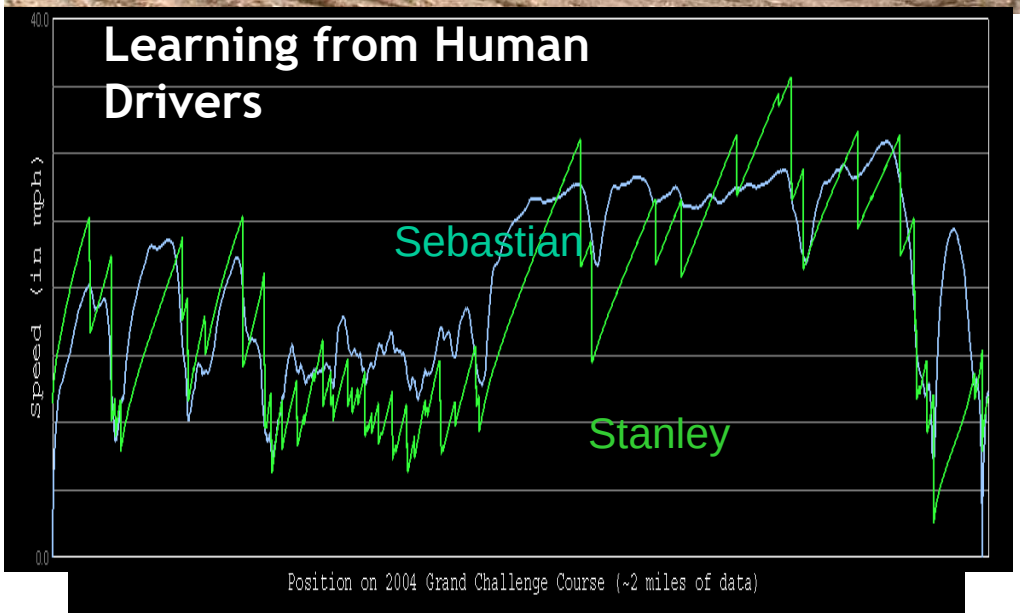
- Nevada made it legal for autonomous cars to drive on roads in June 2011
- As of 2013, four states (Nevada, Florida, California, and Michigan) have legalized autonomous cars

UPenn's Autonomous Car →

Slide credit: Eric Eaton

Autonomous Car Technology





Slide credit: Eric Eaton

Deep Learning in the Headlines

BUSINESS NEWS

MIT Technology Review

Is Google Cornering the Market on Deep Learning?

A cutting-edge corner of science is being wooed by Silicon Valley, to the dismay of some academics.

By Antonio Regalado on January 29, 2014

How much are a dozen deep-learning researchers worth? Apparently, more than $400 million.

This week, Google reportedly paid that much to acquire DeepMind Technologies, a startup based in

BloombergBusinessweek
Technology

Acquisitions

The Race to Buy the Human Brains Behind Deep Learning Machines

By Ashlee Vance | January 27, 2014

intelligence projects. "DeepMind is bona fide in terms of its research capabilities and depth," says Peter Lee, who heads Microsoft Research.

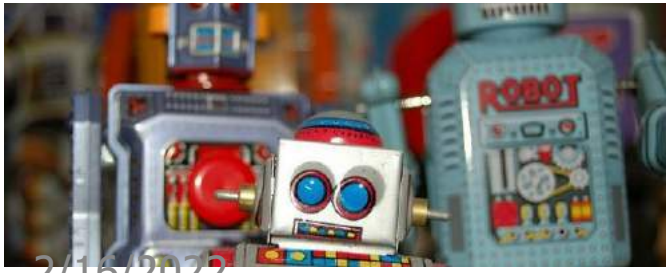
According to Lee, Microsoft, Facebook (FB), and Google find themselves in a battle for deep learning talent. Microsoft has gone from four full-time deep learning experts to 70 in the past three years. "We would have more if the talent was there to

WIRED GEAR SCIENCE ENTERTAINMENT BUSINESS SECURITY DESIG

INNOVATION INSIGHTS | community content | featured

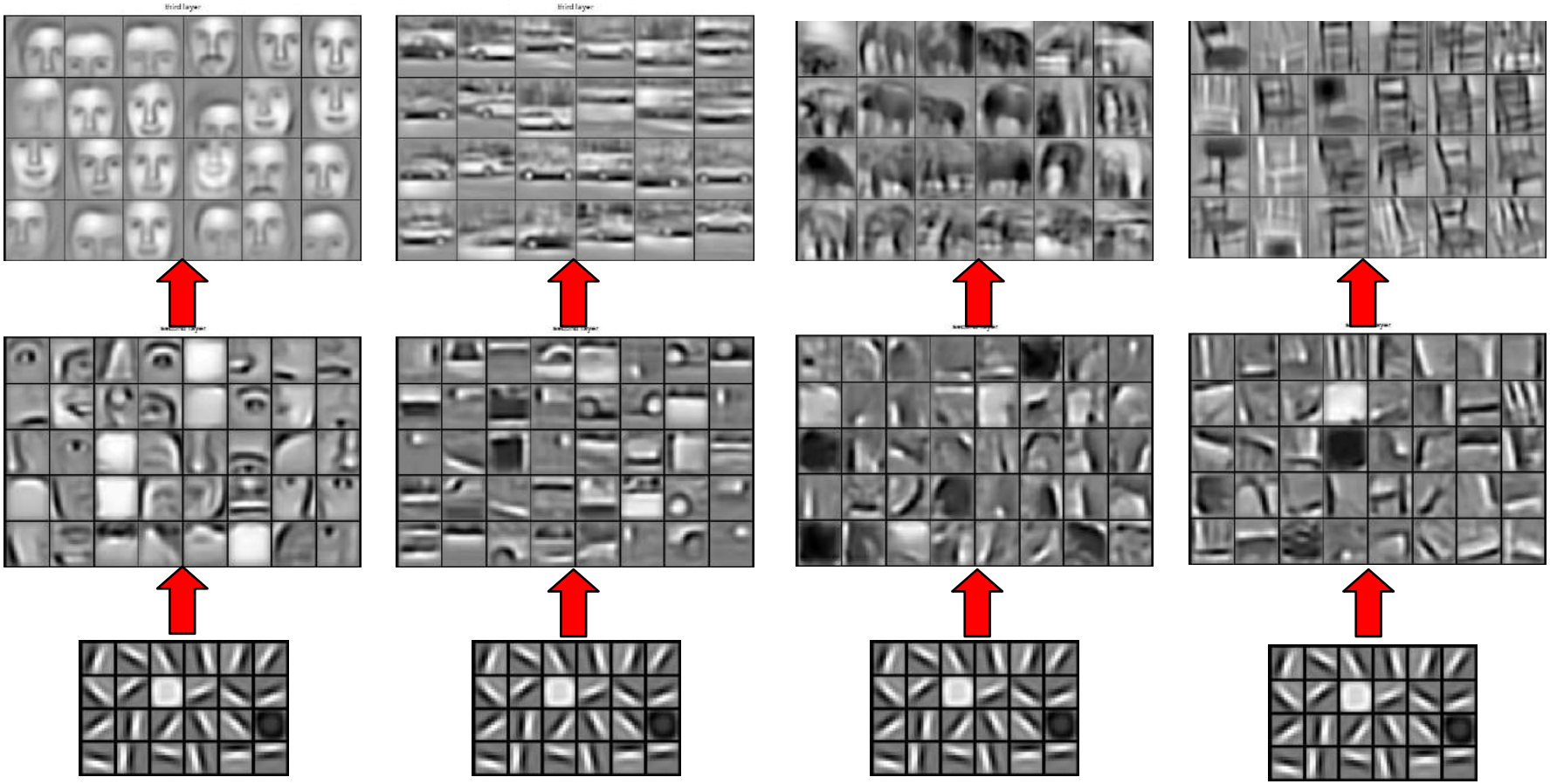
Deep Learning's Role in the Age of Robots

BY JULIAN GREEN, JETPAC 05.02.14 2:56 PM



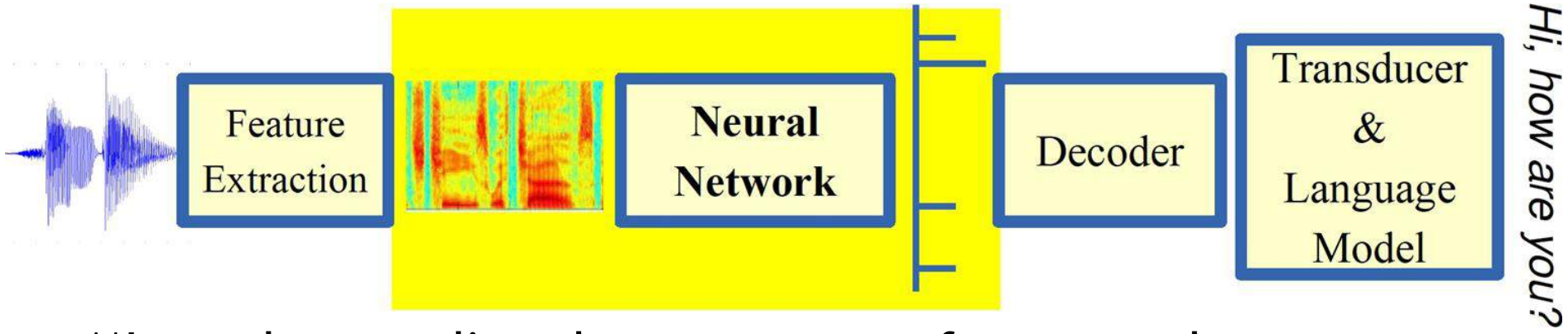
Slide credit: Eric Eaton

Learning of Object Parts

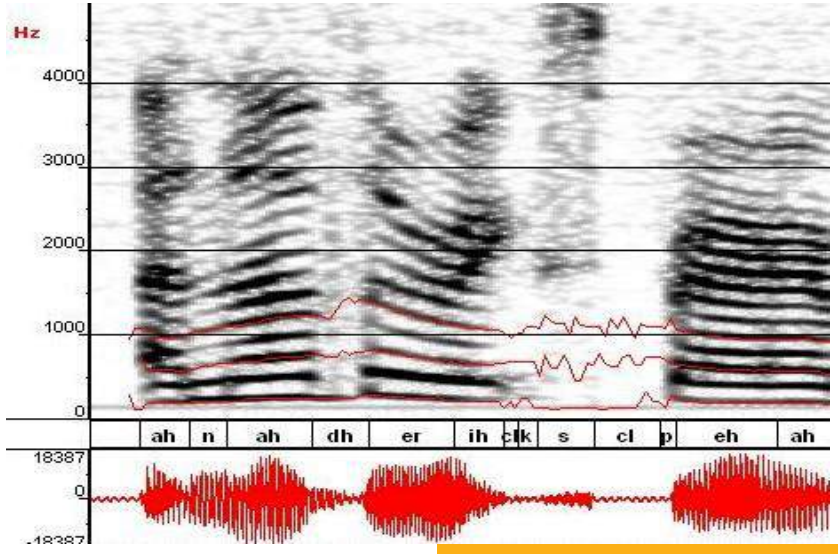


Slide credit: Andrew Ng

Automatic Speech Recognition



ML used to predict phoneme states from sound spectrogram Deep Learning Based Results



# Hidden Layers	1	2	4	8	10	12
Word Error Rate %	16.0	12.8	11.4	10.9	11.0	11.1

Baseline Gaussian Mixture Model based word error rate = 15.4%

[Zeiler et al. "On rectified linear units for speech recognition" ICASSP 2013]

Slide credit: Eric Eaton

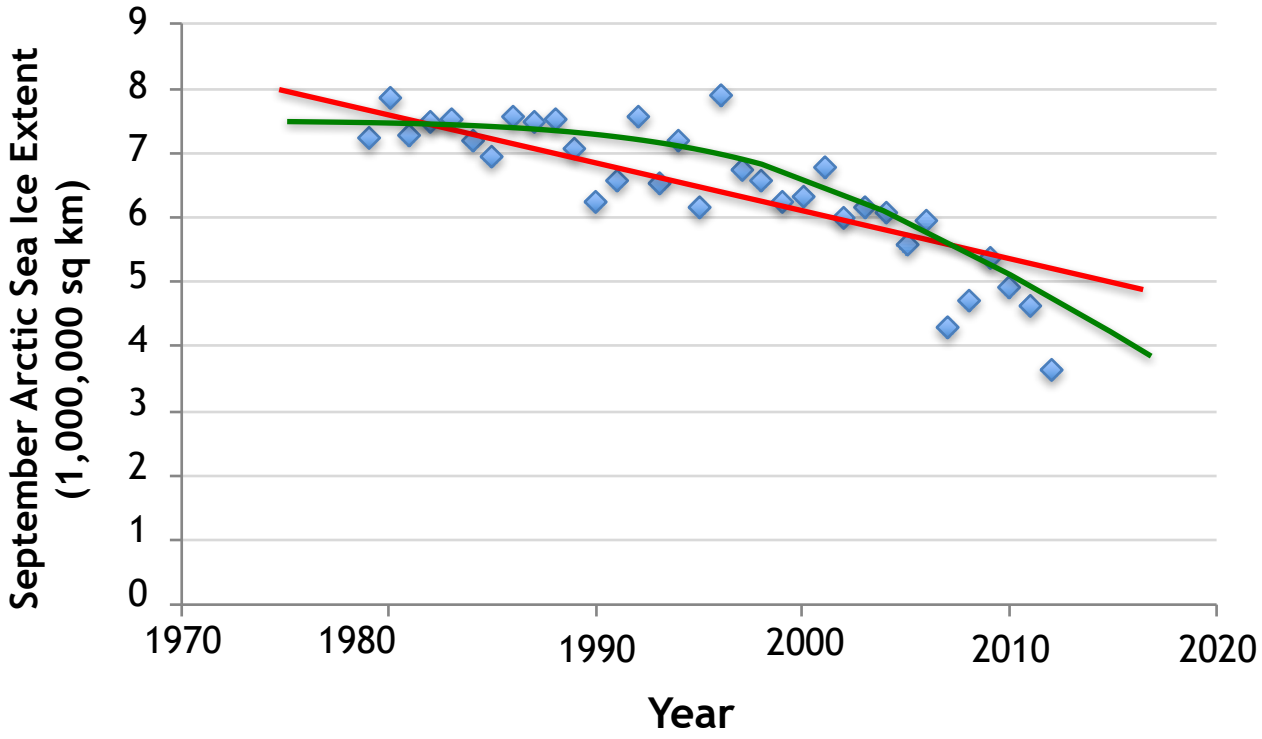
Types of Learning

- **Supervised (inductive) learning**
  - Given: training data, desired outputs (labels)
- **Unsupervised learning**
  - Given: training data (without desired outputs)
- **Semi-supervised learning**
  - Given: training data + a few desired outputs
- **Reinforcement learning**
  - Given: rewards from sequence of actions

Slide Credit: Eric Eaton

Supervised Learning: Regression

- Given $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Learn a function $f(x)$ to predict y given x

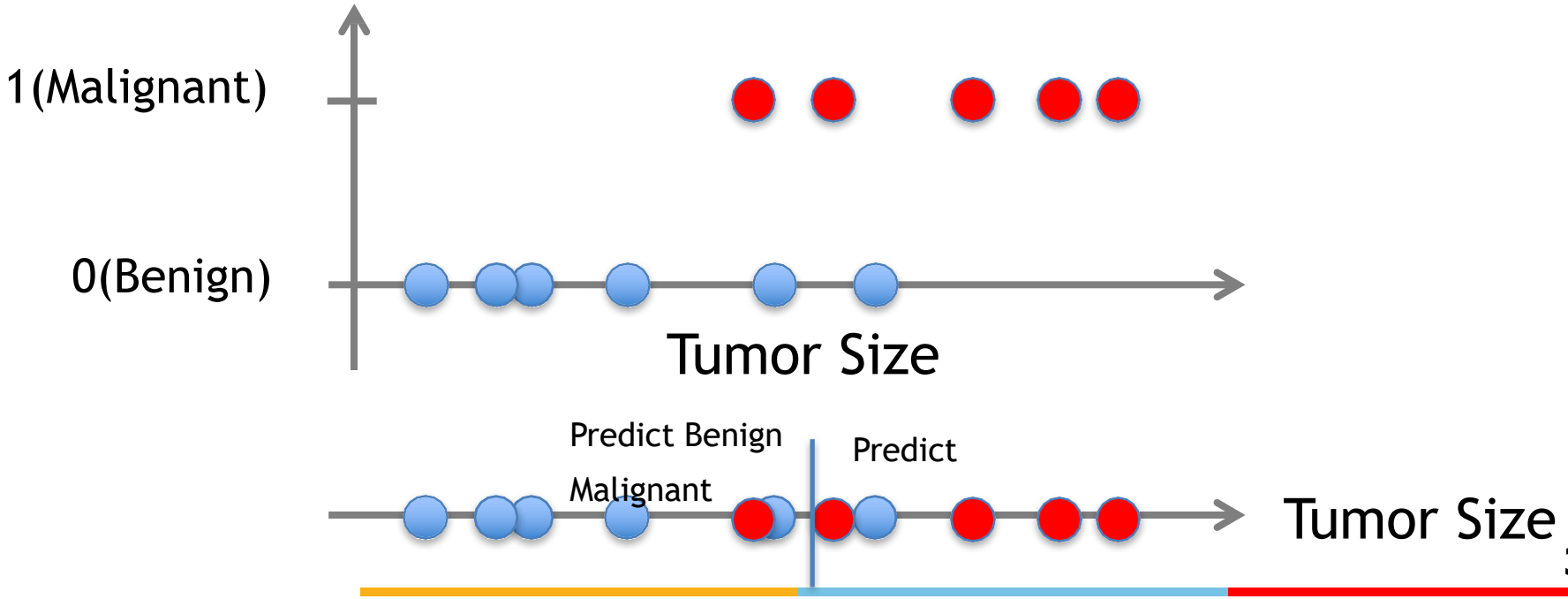


Data from G. Witt. Journal of Statistics Education, Volume 21, Number 1 (2013) Slide Credit: Eric Eaton

Supervised Learning: Classification

- Given (x_1, y_1), (x_2, y_2), …, (x_n, y_n)
- Learn a function $f(x)$ to predict y given x
  - y is categorical == classification

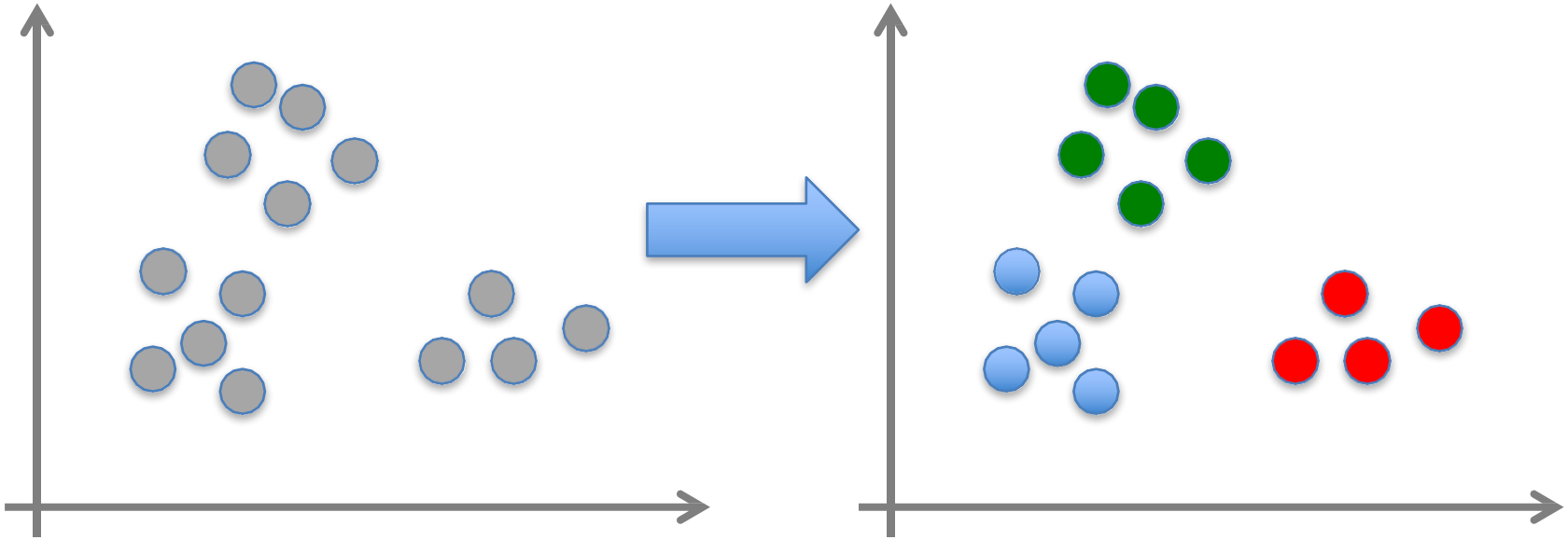
Breast Cancer (Malignant / Benign)



Slide Credit: Eric Eaton

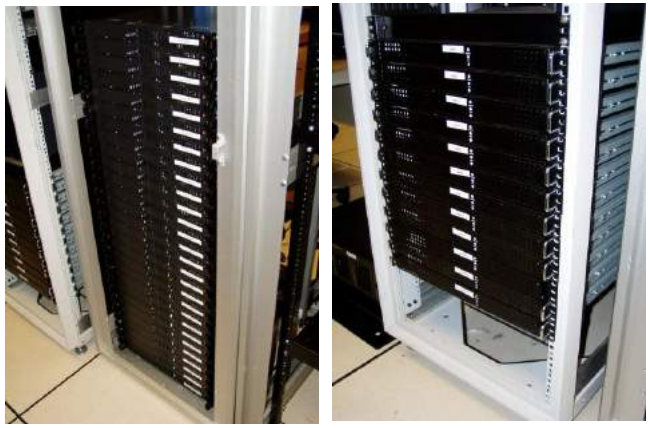
Unsupervised Learning

- Given $x_1, x_2, \ldots, x_n$ (without labels)
- Output hidden structure behind the x's
  - E.g., clustering

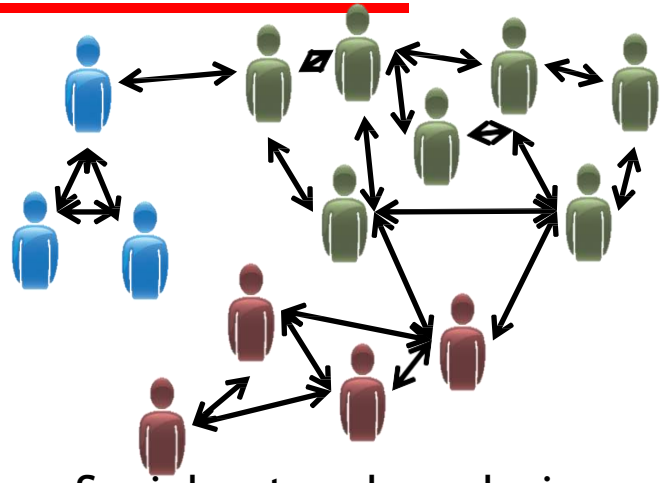


Slide Credit: Eric Eaton

Unsupervised Learning



Organize computing clusters



Social network analysis

Market segmentation

Astronomical data analysis

Slide credit: Andrew Ng

Reinforcement Learning

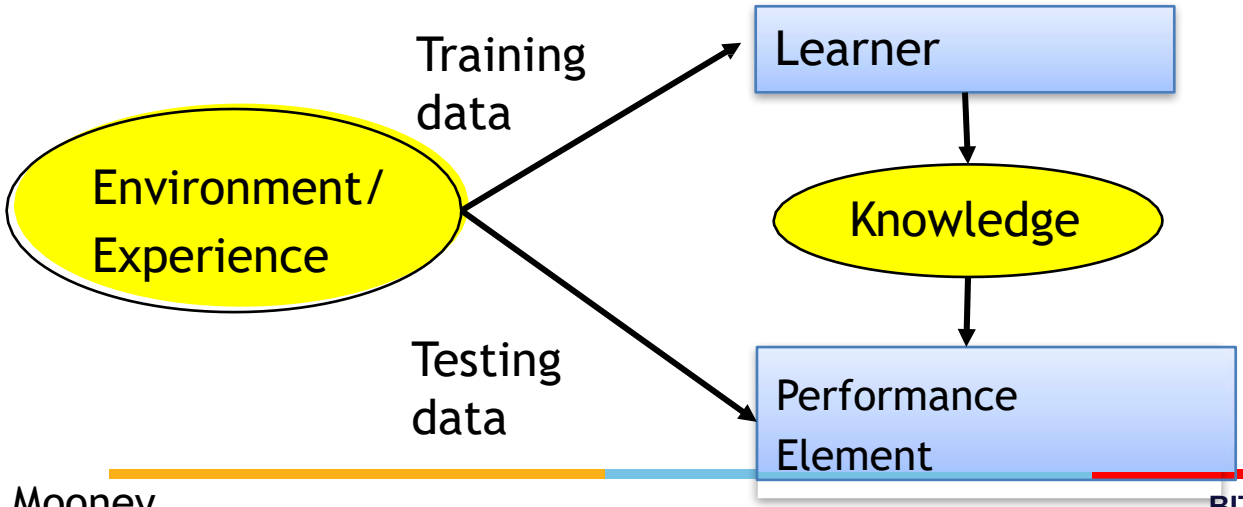
- Given a sequence of states and actions with (delayed) rewards, output a policy
  - Policy is a mapping from states → actions that tells you what to do in a given state
- Examples:
  - Credit assignment problem
  - Game playing
  - Robot in a maze

Slide Credit: Eric Eaton

Design a Learning System

Designing a Learning System

- Choose the training experience(data)
- Choose exactly what is to be learned
  - i.e. the ***target function***
- Choose how to represent the target function
- Choose a learning algorithm to infer the target function from the experience



Slide Credit: Ray Mooney

Designing a Learning System: An Example

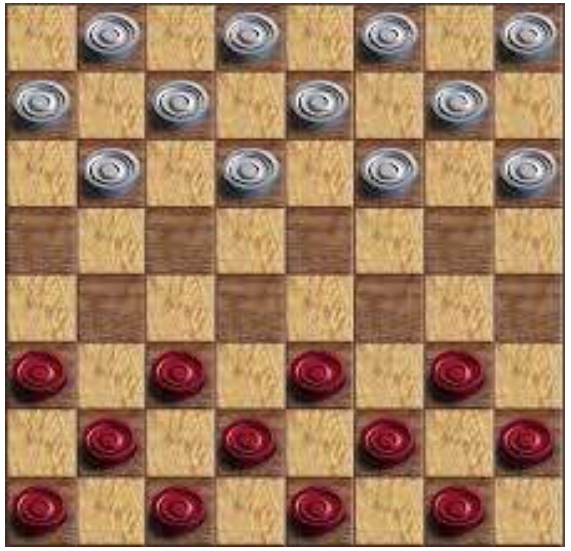
1. Problem Description (Ex: Playing checkers)
2. Choosing the Training Experience (data expressed as features)
3. Choosing the Target Function to be learnt (Ex: deciding next board position
4. Choosing a Representation for the Target Function (design a function as linear etc)
5. Choosing a Function Approximation Algorithm (parameters learning using loss function)
6. Final Design

Choosing the training experience

- The first problem to choose the type of training experience from which our system will learn.
  - ✓ The type of training experience available can have a significant impact on success or failure of the learner.
  - ✓ One key attribute is whether the training experience provides ***direct*** or ***indirect*** feedback regarding the choices made by the performance system.

Choosing the training experience

- In learning to play checkers, the system might learn from **direct training** examples consisting of individual checkers board states and the correct move for each.
- Alternatively, it might have available only **indirect information** consisting of the move sequences and final outcomes of various games played.



Choosing the training experience

- In **indirect training**, information about the correctness of specific moves early in the game must be inferred indirectly from the fact that the game was eventually won or lost.
- The learner faces an additional problem of **credit assignment**, or determining the degree to which each move in the sequence deserves credit or blame for the final outcome.
- Credit assignment can be a particularly difficult problem because the game can be lost even when early moves are optimal, if these are followed later by poor moves. Hence, learning from direct training feedback is typically easier than learning from indirect feedback.

Choosing the training experience

- A second important attribute of the training experience is the degree to which the learner controls the sequence of training examples.
  - the learner might rely on the teacher to select informative board states and to provide the correct move for each.
  - the learner might itself propose board states that it finds particularly confusing and ask the teacher for the correct move.

Choosing the training experience

- the learner may have complete control over both the board states and (indirect) training classifications, as it does when it learns by playing against itself with no teacher present.
  - the learner may choose between experimenting with novel board states that it has not yet considered, or sharpen its skill by playing minor variations of lines of play it currently finds most promising.

Choosing the training experience

- A third important attribute of the training experience is how well it represents the distribution of examples over which the final system performance ***P*** must be measured.
  - learning is most reliable when the training examples follow a distribution similar to that of future test examples.
  - the performance metric ***P*** is the percent of games the system wins in the world tournament. If its training experience ***E*** consists only of games played against itself, there is an obvious danger that this training experience might not be fully representative of the distribution of situations over which it will later be tested.

Choosing the training experience

- For example, the learner might never encounter certain crucial board states that are very likely to be played by the human checkers champion.
- it is often necessary to learn from a distribution of examples that is somewhat different from those on which the final system will be evaluated
- one distribution of examples will not necessary lead to strong performance over some other distribution.

Choosing the Target Function

- *ChooseMove*, however, is difficult to learn.
- An easier and related target function to learn is function

V: B --> R, which assigns a numerical score to each board. The better the board positions B, the higher the score R.

- If the system can successfully learn such a target function ***V,*** then it can easily use it to select the best move from any current board position.
- This can be accomplished by generating the successor board state produced by every legal move, then using ***V*** to choose the best successor state and therefore the best legal move.

Choosing the Target Function

- Let us therefore define the target value ***V(b)*** for an arbitrary board state ***b*** in ***B***, as follows:
  1. If ***b*** is a final board state that is won, then ***V(b) = 100***
  2. If ***b*** is a final board state that is lost, then ***V(b) = -100***
  3. If ***b*** is a final board state that is draw, then ***V(b) = 0***
  4. If ***b*** is a not a final state in the game, then ***V(b) = V(b')***, where ***b'*** is the best final board state that can be achieved starting from ***b*** and playing optimally until the end of the game.

Choosing the Target Function

- This recursive definition specifies a value of ***V(b)*** for every board state ***b.***
- It is not usable by player - it is not efficiently computable - it is a ***nonoperational definition.***
- The goal of learning in this case is to discover an ***operational description of V*** i.e., a description that can be used by the checkers - playing program to evaluate states and select moves within realistic time bounds.

Choosing the Target Function

- Reduced the learning task to the problem of discovering an operational description of the ideal target function ***V*** – it is difficult to learn ***V*** perfectly.
- We often expect learning algorithms to acquire only some ***approximation*** to the target function -- is called ***function approximation.*** [The actual function can often not be learned and must be approximated]

Choosing a Representation for the Target Function

- **Expressiveness versus Training set size**
  - More expressive the representation of the target function, the closer to the "truth" we can get.
  - More expressive the representation, the more training examples are necessary to choose among the large number of "representable" possibilities.
- **Example of a representation:**
  - x1/x2 = # of black/red pieces on the board
  - x3/x4 = # of black/red king on the board
  - x5/x6 = # of black/red pieces threatened by red/black

$$\hat{V}(b) = w0+w1.x1+w2.x2+w3.x3+w4.x4+w5.x5+w6.x6$$

wi's are adjustable or "learnable" coefficients

Choosing a Representation for the Target Function

- w0 through w6 are numerical coefficients, or weights, to be chosen by the learning algorithm.
- Learned values for the weights w1 through w6 will determine the relative importance of the various board features in determining the value of the board, whereas the weight w0 will provide an additive constant to the board value.

Choosing a Representation for the Target Function

Partial design of a checkers learning program:

- Task T: playing checkers
- Performance measure P: percent of games won in the world tournament
- Training experience E: games played against itself
- *Target function:* $V : Board \rightarrow \Re$
- *Target function representation*

$$\hat{V}(b) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5 + w_6 x_6$$

Choosing a Function Approximation Algorithm

- **Generating Training Examples of the form <b,V_{train}(b)>** [e.g. <x1=3, x2=0, x3=1, x4=0, x5=0, x6=0, +100 (=blacks won)]
  - x1= # of black pieces on the board
  - x2 = # of red pieces on the board
  - x3 = # of black king on the board
  - x4 = # of red king on the board
  - x5 = # of black pieces threatened by red
  - x6 = # of red pieces threatened by black

Choosing a Function Approximation Algorithm

- **Estimating the training Values**
  - only training information available to learner is whether the game was eventually won or lost.
  - we require training examples that assign specific scores to specific board states.
  - easy to assign a value to board states that correspond to the end of the game,
  - less obvious how to assign training values to the more numerous intermediate board states that occur before the game's end.
  - fact that the game was eventually won or lost does not necessarily indicate that every board state along the game path was necessarily good or bad.

Choosing a Function Approximation Algorithm

- **Estimating the training Values**
  - Despite the ambiguity inherent in estimating training values for intermediate board states.
  - one simple approach is to assign the training value of $V_{train}(b)$ for **any intermediate board** state **b** to be $\hat{V}$**(Successor(b))**, where $\hat{V}$ is the learner's current approximation to **V** and where **Successor(b)** denotes the next board state following **b**
  - This rule for estimating training values can be summarized as

$$V_{train}(b) \leftarrow \hat{V}(Successor(b))$$

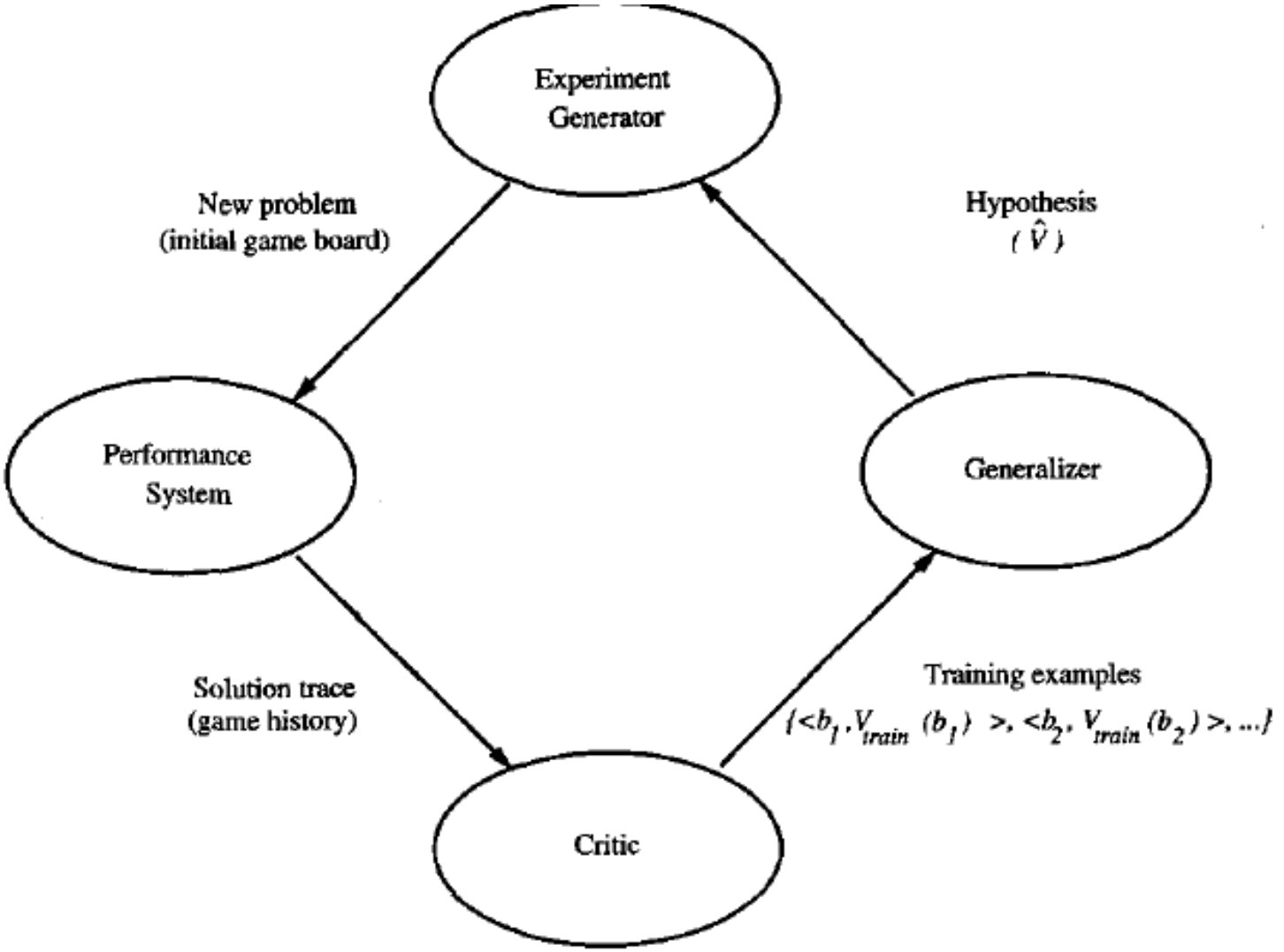
Choosing a Function Approximation Algorithm

- **Adjust the weights**
  - Specify the learning algorithm for choosing the weights **wi** to best fit the set of training examples {**<b,V_{train}(b)>** }.
  - One common approach is to define the best hypothesis, or set of weights, as that which minimizes the squared error ***E*** between the training values and the values predicted by the hypothesis $\hat{V}$ **.**

$$E \equiv \sum_{\langle b, V_{train}(b)\rangle \in \text{ training examples}} (V_{train}(b) - \hat{V}(b))^2$$

  - Thus, we seek the weights, or equivalently the $\hat{V}$, that minimize ***E*** for the observed training examples.

Final Design for Checkers Learning



Final Design for Checkers Learning

- **The Performance Module (performance)** : Takes as input a new board and outputs a trace of the game it played against itself.
- **The Critic (data generation)** : Takes as input the trace of a game and outputs a set of training examples of the target function
- **The Generalizer (learner)**: Takes as input training examples and outputs a hypothesis which estimates the target function.
- **The Experiment Generator (task):** Takes as input the current hypothesis (currently learned function) and outputs a new problem (an initial board state) for the performance system to explore

Issues in Machine Learning

- What algorithms are available for learning a concept? How well do they perform?
- How much training data is sufficient to learn a concept with high confidence?
- When is it useful to use prior knowledge?
- Are some training examples more useful than others?
- What are the best tasks for a system to learn?
- What is the best way for a system to represent its knowledge?

ML in a Nutshell

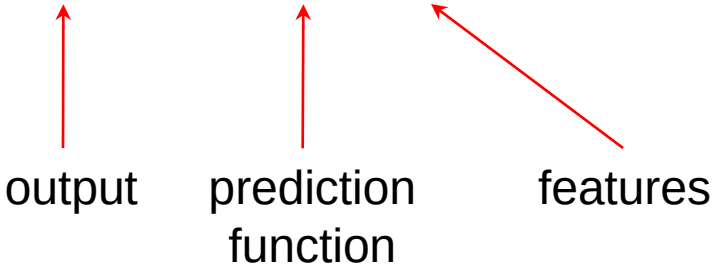
- Tens of thousands of machine learning algorithms
  - Hundreds new every year

- Every ML algorithm has three components
  - **Representation**
  - **Optimization**
  - **Evaluation**

Slide credit: Pedro Domingos

Regression

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



- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

Slide credit: L. Lazebnik

Regression Example

- Apply a prediction function to a feature representation of the image to get the desired output:

f() = “apple”

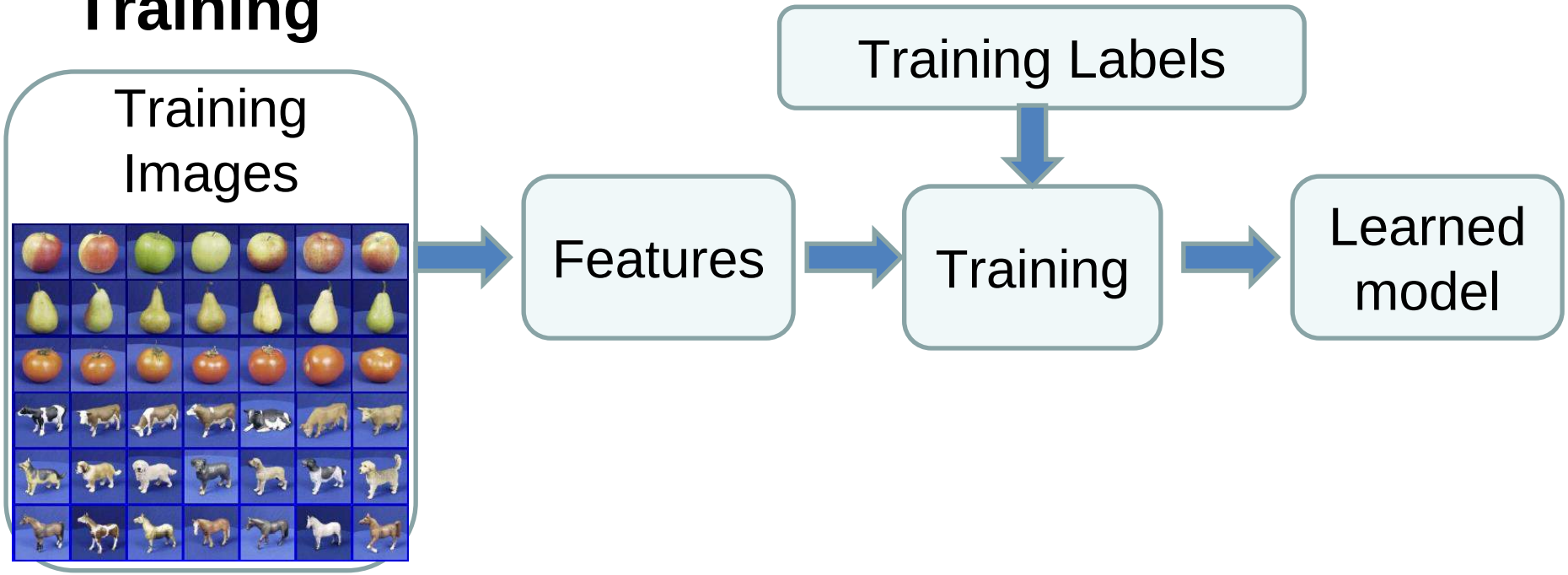
f() = “tomato”

f() = “cow”

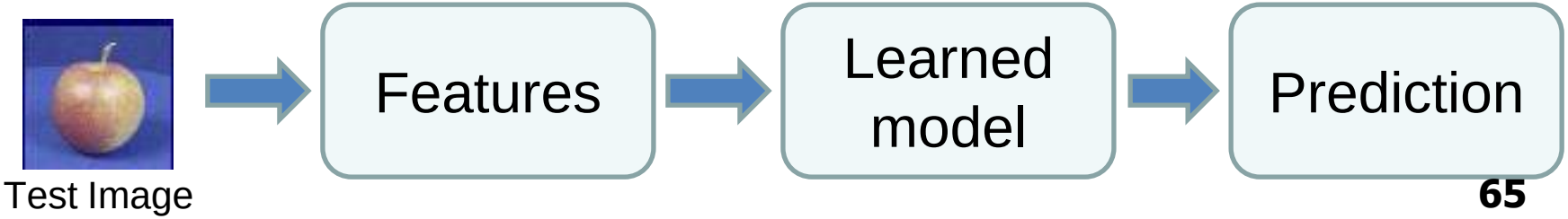
Slide credit: L. Lazebnik

Classification

Training



Testing



Slide credit: D. Hoiem and L. Lazebnik

Function Representations

- Numerical functions
  - Linear regression
  - Neural networks
  - Support vector machines
- Symbolic functions
  - Decision trees
  - Rules in propositional logic
  - Rules in first-order predicate logic
- Instance-based functions
  - Nearest-neighbor
  - Case-based
- Probabilistic Graphical Models
  - Naïve Bayes
  - Bayesian networks
  - Hidden-Markov Models (HMMs)
  - Probabilistic Context Free Grammars (PCFGs)
  - Markov networks

Slide credit: Ray Mooney

Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- etc.

Slide credit: Pedro Domingos

Evaluating Performance

- If *y* is discrete:
  - Accuracy: # correctly classified / # all test examples
  - Precision/recall
    - True Positive, False Positive, True Negative, False Negative
    - **P**recision = TP / (TP + FP) = # predicted true pos / # predicted pos
    - **R**ecall = TP / (TP + FN) = # predicted true pos / # true pos
  - F-measure

    = 2PR / (P + R)

- Want evaluation metric to be in some range, e.g. [0 1]
  - 0 = worst possible classifier, 1 = best possible classifier

Evaluating Performance

- If *y* is continuous:
  - Sum-of-Squared-Differences (SSD) error between predicted and true *y*:

$$\mathbf{E} = \sum_{\mathbf{i}=1}^{\mathbf{n}} (\mathbf{f}(\mathbf{x_i}) - \mathbf{y_i})^2$$

Training vs Testing

- What do we want?
  - High accuracy on training data?
  - No, high accuracy on *unseen/new/test data!*
  - Why is this tricky?
- Training data
  - Features (x) and labels (y) used to learn mapping f
- Test data
  - Features used to make a prediction
  - Labels only used to see how well we've learned f!!!
- Validation data
  - Held-out set of the *training data*
  - Can use both features and labels to tune *parameters* of the model we're learning

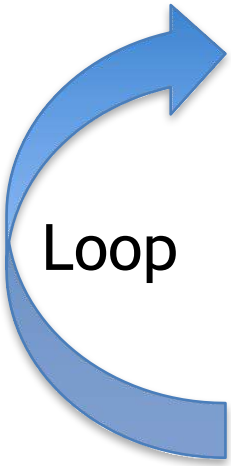
Slide Credit: Adriana

Training vs. Test Distribution

- We generally assume that training and test examples are independently drawn from the same overall distribution of data
  - We call this “i.i.d” which stands for “independent and identically distributed”

Slide credit: Ray Mooney

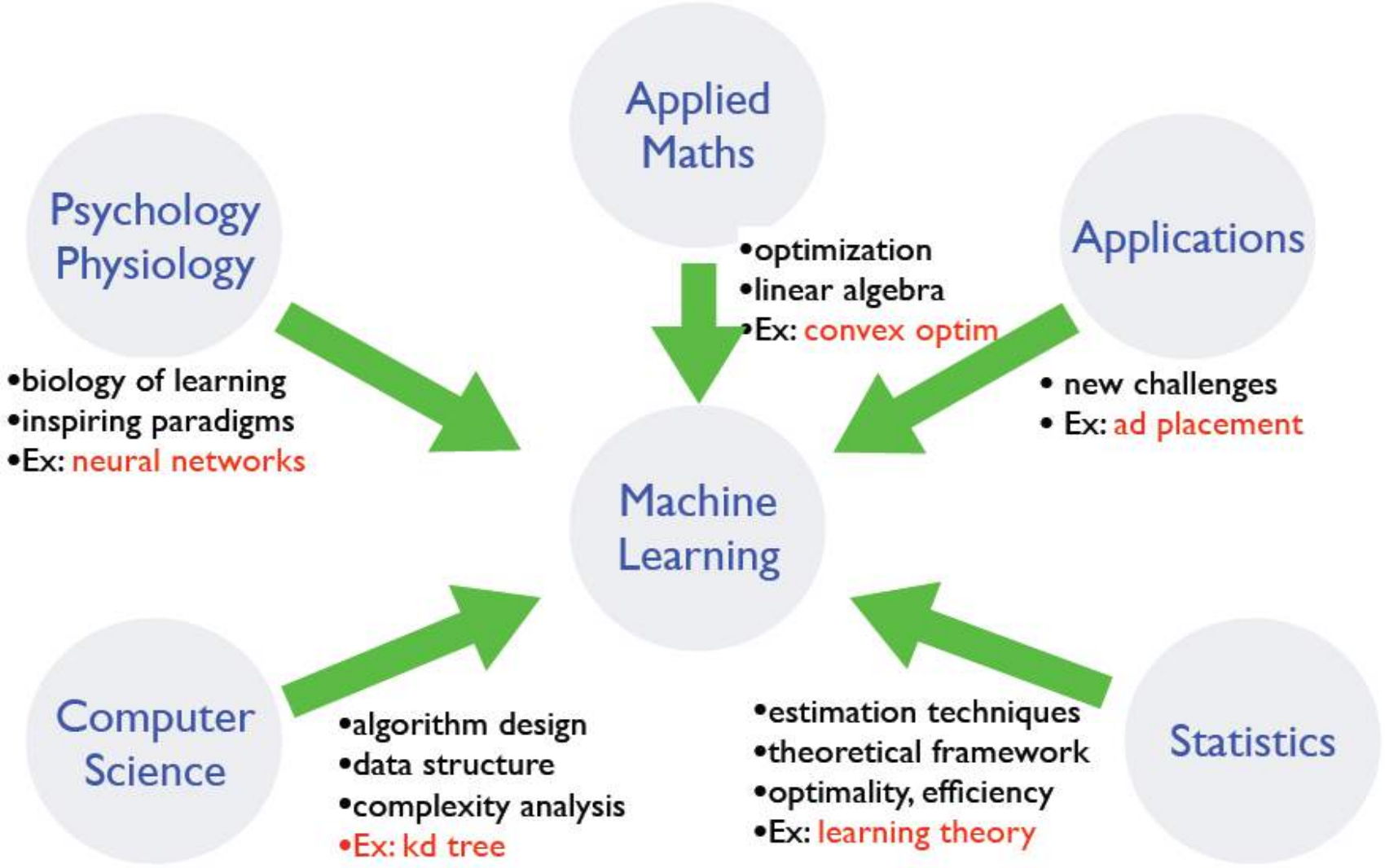
ML in Practice



- Understand domain, prior knowledge, and goals
- Data integration, selection, cleaning, pre-processing, etc.
- Learn models
- Interpret results
- Consolidate and deploy discovered knowledge

Based on a slide by Pedro Domingos

Where does ML fit in?



Slide credit: Dhruv Batra, Fei Sha

Recommended Readings

- Tom M. Mitchell, Machine Learning, The McGraw-Hill Companies, Inc. International Edition 1997. **[Ch. 1]**
- **http://www.cs.princeton.edu/courses/archive/spr08/cos511/ [Web]**

Thank You

Machine Learning
DSECL ZG565

Dr. Chetana Gavankar, Ph.D,
IIT Bombay-Monash University Australia
Chetana.gavankar@pilani.bits-pilani.ac.in

BITS Pilani
Pilani Campus

innovate achieve lead

BITS Pilani

Pilani Campus

Lecture No. – 2 | Math Preliminaries

Date – 02/11/2019

Time – 9:00 AM – 11:00 AM

These slides are prepared by the instructor, with grateful acknowledgement of Prof. Sugato Ghosal, Prof. S.K. H. Islam from BITS Pilani and many others who made their course materials freely available online.

Session Content

- Linear Algebra Review
- Calculus Review
- Probability Theory (Ref: 1.2)
- Decision Theory (Ref: 1.5)
- Information Theory (Ref: 1.6)

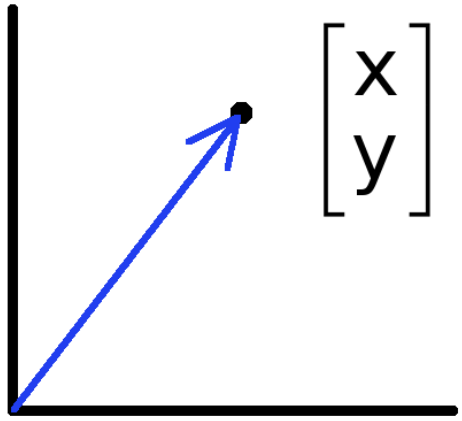
Linear algebra Review

See http://cs229.stanford.edu/section/cs229-linalg.pdf for more

Vectors and Matrices

- Collections of ordered numbers that represent movements in space, scaling factors, word counts, movie ratings, pixel brightness, etc.
- Vector is a mathematical quantity that has magnitude and direction

Vectors



- Vectors can represent an offset in 2D or 3D space
- Points are just vectors from the origin

- Data can also be treated as a vector
- Such vectors don't have a geometric interpretation, but calculations like "distance" still have value

Vector

- A column vector $\mathbf{v} \in \mathbb{R}^{n \times 1}$ where

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

- A row vector $\mathbf{v}^T \in \mathbb{R}^{1 \times n}$ where

$$\mathbf{v}^T = \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}$$

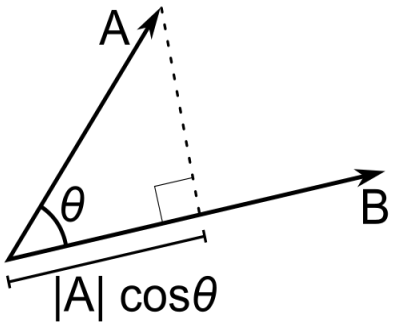
T denotes the transpose operation

Inner Product

- Multiply corresponding entries of two vectors and add up the result

$$\mathbf{x}^T\mathbf{y} = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} x_i y_i \qquad \text{(scalar)}$$

- If B is a unit vector, then A·B gives the length of A which lies in the direction of B (projection)



(if B is unit-length hence norm is 1)

Norms

- **Norm** is a function that assigns a strictly positive *length* or *size* to each vector in a vector space—except for the zero vector

- **L^1 norm -** One-dimensional vector spaces

$$\|\boldsymbol{x}\|_1 := \sum_{i=1}^{n} |x_i|$$

- **L^2 norm -** *n*-dimensional Euclidean space **R^n**,

$$\|\boldsymbol{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}$$

- **L^p norm** - Let p ≥ 1 be a real number. The p norm of vector x=(x_1, x_2,…x_n)

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

Matrix

- A matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is an array of numbers with size $m \downarrow$ by $n \rightarrow$, i.e. m rows and n columns.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & & & & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{bmatrix}$$

- If $m = n$, we say that $\mathbf{A}$ is square.

Matrix Operations

- Addition

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} a+1 & b+2 \\ c+3 & d+4 \end{bmatrix}$$

  - Can only add a matrix with matching dimensions, or a scalar.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + 7 = \begin{bmatrix} a+7 & b+7 \\ c+7 & d+7 \end{bmatrix}$$

- Scaling

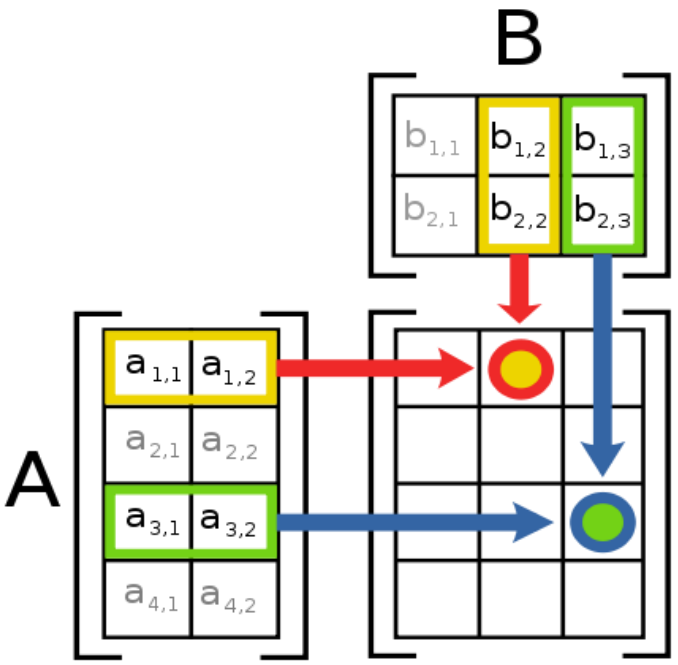
$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times 3 = \begin{bmatrix} 3a & 3b \\ 3c & 3d \end{bmatrix}$$

Matrix Multiplication

- Let X be an *a*x*b* matrix, Y be an *b*x*c* matrix
- Then Z = X*Y is an *a*x*c* matrix
- Second dimension of first matrix, and first dimension of first matrix have to be the same, for matrix multiplication to be possible

Matrix Multiplication

- The product AB is:



- Each entry in the result is (that row of A) dot product with (that column of B)

Different types of product

- $\boldsymbol{x}, \boldsymbol{y}$ = column vectors (nx1)
- $\boldsymbol{X}, \boldsymbol{Y}$ = matrices (mxn)
- x, y = scalars (1x1)

- $\boldsymbol{x}^T \boldsymbol{y} = \boldsymbol{x} \cdot \boldsymbol{y}$ = inner product (1xn x nx1 = scalar)
- $\boldsymbol{x} \otimes \boldsymbol{y} = \boldsymbol{x}\, \boldsymbol{y}^T$ = outer product (nx1 x 1xn = matrix)

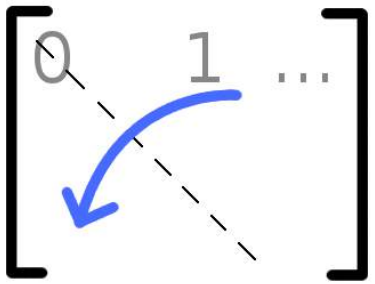
- $\boldsymbol{X} * \boldsymbol{Y}$ = matrix product
- $\boldsymbol{X} .* \boldsymbol{Y}$ = element-wise product

Inverse

- Given a matrix **A**, its inverse $\mathbf{A^{-1}}$ is a matrix such that $\mathbf{AA^{-1} = A^{-1}A = I}$
- E.g. $\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{3} \end{bmatrix}$
- Inverse does not always exist. If $\mathbf{A^{-1}}$ exists, **A** is *invertible* or *non-singular*. Otherwise, it's *singular*.

Matrix Operations

- Transpose – flip matrix, so row 1 becomes column 1



$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \\ 4 & 5 \end{bmatrix}^T = \begin{bmatrix} 0 & 2 & 4 \\ 1 & 3 & 5 \end{bmatrix}$$

- A useful identity:

$$(ABC)^T = C^T B^T A^T$$

Matrix Rank

- Column/row rank
  - col-rank(A)=no. of independent columns
  - row-rank(A)=no. of independent rows
- Column rank always equals row rank
- Matrix rank $\text{rank}(\mathbf{A}) \triangleq \text{col-rank}(\mathbf{A}) = \text{row-rank}(\mathbf{A})$
- If a matrix is not full rank, inverse doesn't exist
  - Inverse also doesn't exist for non-square matrices

Matrix Operation Properties

- Matrix addition is commutative and associative
  - A + B = B + A
  - A + (B + C) = (A + B) + C
- Matrix multiplication is associative and distributive but *not* commutative
  - A(B*C) = (A*B)C
  - A(B + C) = A*B + A*C
  - A*B != B*A

Linear independence

- Suppose we have a set of vectors $\mathbf{v_1}, \ldots, \mathbf{v_n}$
- If we can express $\mathbf{v}_1$ as a linear combination of the other vectors $\mathbf{v}_2 \ldots \mathbf{v}_n$, then $\mathbf{v}_1$ is linearly *dependent* on the other vectors.
  - The direction $\mathbf{v}_1$ can be expressed as a combination of the directions $\mathbf{v}_2 \ldots \mathbf{v}_n$. (E.g. $\mathbf{v}_1 = .7\ \mathbf{v}_2 - .7\ \mathbf{v}_4$)
- If no vector is linearly dependent on the rest of the set, the set is linearly *independent*.
  - Common case: a set of vectors $\mathbf{v_1}, \ldots, \mathbf{v_n}$ is always linearly independent if each vector is perpendicular to every other vector (and non-zero)

ingular Value Decomposition (SVD)

- There are several computer algorithms that can "factor" a matrix, representing it as the product of some other matrices
- The most useful of these is the Singular Value Decomposition
- **Singular value decomposition** (**SVD**) is a factorization of a real or complex matrix
- Represents any matrix **A** as a product of three matrices: $\mathbf{U\Sigma V^T}$

https://en.wikipedia.org/wiki/Singular_value_decomposition

Singular Value Decomposition (SVD)

- In general, if **A** is *m* x *n*, then **U** will be *m* x *m*, **Σ** will be *m* x *n*, and $\mathbf{V^T}$ will be *n* x *n*.

$$\underset{U}{\begin{bmatrix} -.39 & -.92 \\ -.92 & .39 \end{bmatrix}} \times \underset{\Sigma}{\begin{bmatrix} 9.51 & 0 & 0 \\ 0 & .77 & 0 \end{bmatrix}} \times \underset{V^T}{\begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix}} = \underset{A}{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}}$$

Singular Value Decomposition (SVD)

For square, positive semi-definite matrix A

$$\mathbf{U\Sigma V^T = A}$$

- Where **U** and **V** are rotation matrices, and **Σ** is a scaling matrix.

$$\underset{U}{\begin{bmatrix} -.40 & .916 \\ .916 & .40 \end{bmatrix}} \times \underset{\Sigma}{\begin{bmatrix} 5.39 & 0 \\ 0 & 3.154 \end{bmatrix}} \times \underset{V^T}{\begin{bmatrix} -.05 & .999 \\ .999 & .05 \end{bmatrix}} = \underset{A}{\begin{bmatrix} 3 & -2 \\ 1 & 5 \end{bmatrix}}$$

- U and V are unitary matrices, i.e., $UU^T = U^TU = I$ (identity matrix)
- Orthogonal matrix is a square matrix whose columns and rows are orthogonal unit vectors (i.e., orthonormal vectors), i.e. $Q^TQ=QQ^T=\mathbf{I}$, where I is the identity matrix.
- Matrix Q is orthogonal if its transpose is equal to its inverse: $Q^T=Q^{-1}$.

https://www.youtube.com/watch?v=NsNNI_-JPUY

Positive-semidefinite matrix

- Symmetric ($M = M^T$) $n * n$ real matrix M is said to be **positive-semidefinite** if the scalar $x^T M x$ is positive or zero for every non-zero column vector x of n real numbers

(domain of x = 1*n, domain of M= n*n and x^T = n*1)

- When interpreting $M x$ as the output of matrix, M, that is acting on an input, x, the property of **positive-semidefinite** implies that the output always is positive or zero ; inner product with the input

$$M \text{ positive semi-definite} \iff x^T M x \geq 0 \text{ for all } x \in \mathbb{R}^n$$

Calculus review

Differentiation

The derivative provides us information about the rate of change of a function.

The derivative of a function is also a function.

Example:

The derivative of the acceleration function is the velocity function.

Derivative = rate of change

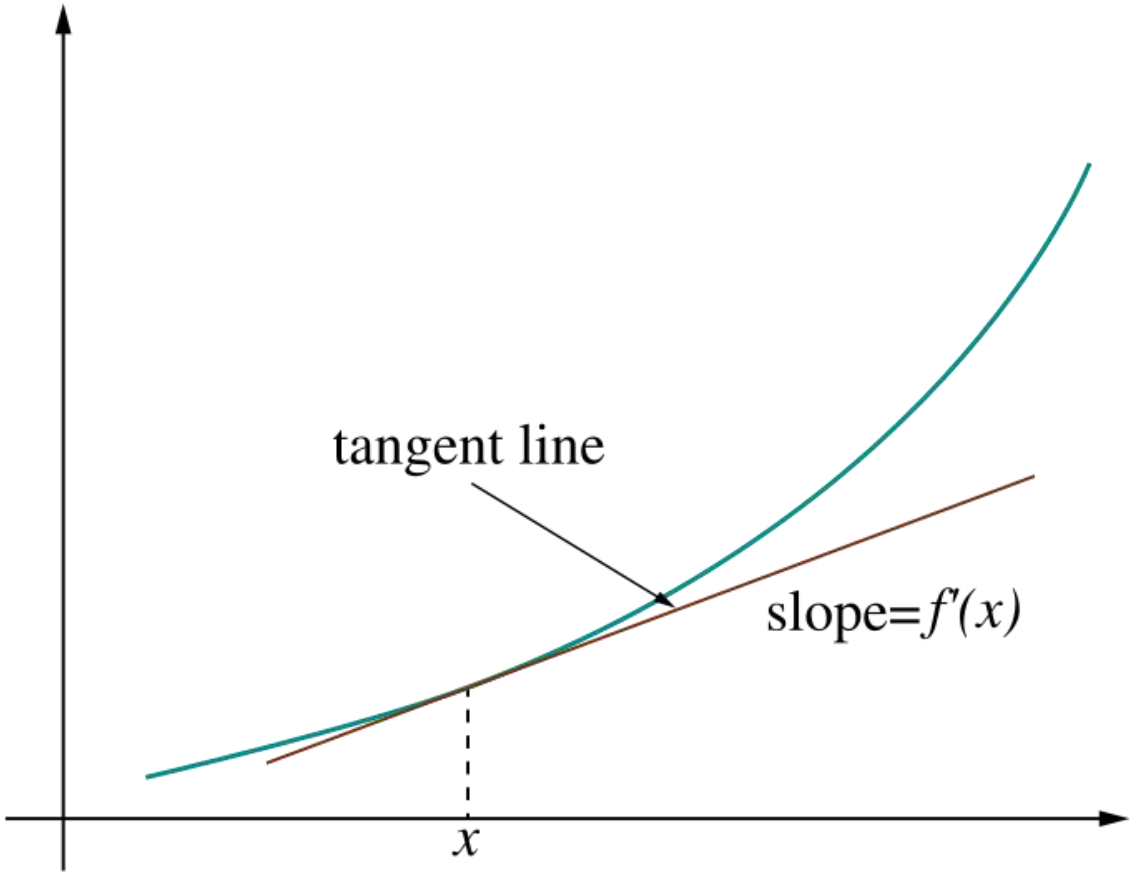


Image: Wikipedia

Derivative = rate of change

- Linear function y = mx + b
- Slope $$m = \frac{\text{change in } y}{\text{change in } x} = \frac{\Delta y}{\Delta x},$$

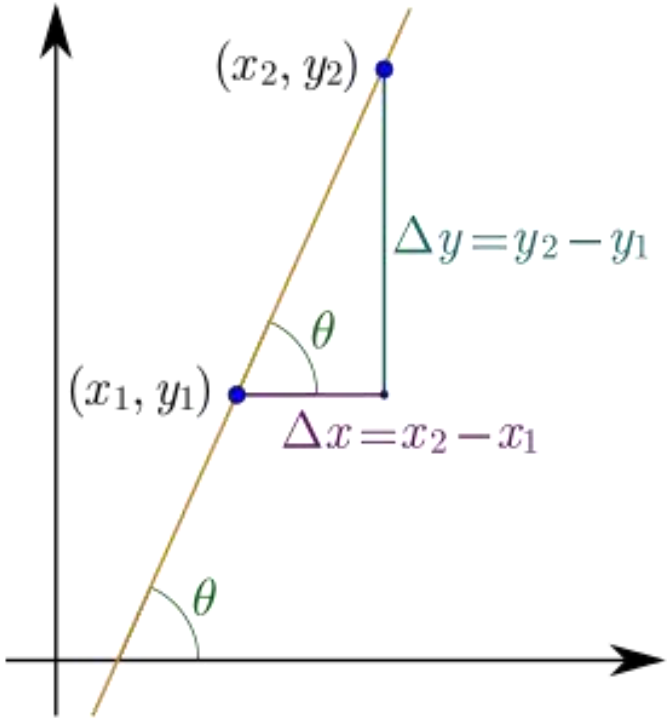


Image: Wikipedia

Ways to Write the Derivative

Given the function f(x), we can write its derivative in the following ways:

- f'(x)
- $\frac{d}{dx}$f(x)

The derivative of x is commonly written dx.

Differentiation Formulas

The following are common differentiation formulas:

- The derivative of a constant is 0.

$$\frac{d}{du}c = 0$$

- The derivative of a sum is the sum of the derivatives.

$$\frac{d}{du}(f(u) + g(u)) = f'(u) + g'(u)$$

More Formulas

- The derivative of *u* to a constant power:

$$\frac{d}{du} u^n = n * u^{n-1}$$

- The derivative of *e*:

$$\frac{d}{du} e^u = e^u$$

- The derivative of *log*:

$$\frac{d}{du} \log(u) = \frac{1}{u}$$

Product and Quotient

The product rule and quotient rules are commonly used in differentiation.

- Product rule:

$$\frac{d}{du}(f(u)*g(u)) = f(u)g'(u) + g(u)f'(u)$$

- Quotient rule:

$$\frac{d}{du}\left(\frac{f(u)}{g(u)}\right) = \frac{g(u)f'(u) - f(u)g'(u)}{g^2(u)}$$

Chain Rule

The chain rule allows you to combine any of the differentiation rules we have already covered.

- First, do the derivative of the outside and then do the derivative of the inside.

$$\frac{d}{du} f(g(u)) = \quad f'(g(u))^* g'(u)^* du$$

Try These

$f(z) = z + 11$

$s(y) = 4ye^{2y}$

$g(y) = 4y^3 + 2y$

$p(x) = \dfrac{\log(x^2)}{x}$

$h(x) = e^{3x}$

$q(z) = (e^z - z)^3$

Solutions

$f'(z) = 1$

$s'(y) = 8ye^{2y} + 4e^{2y}$

$g'(y) = 12y^2 + 2$

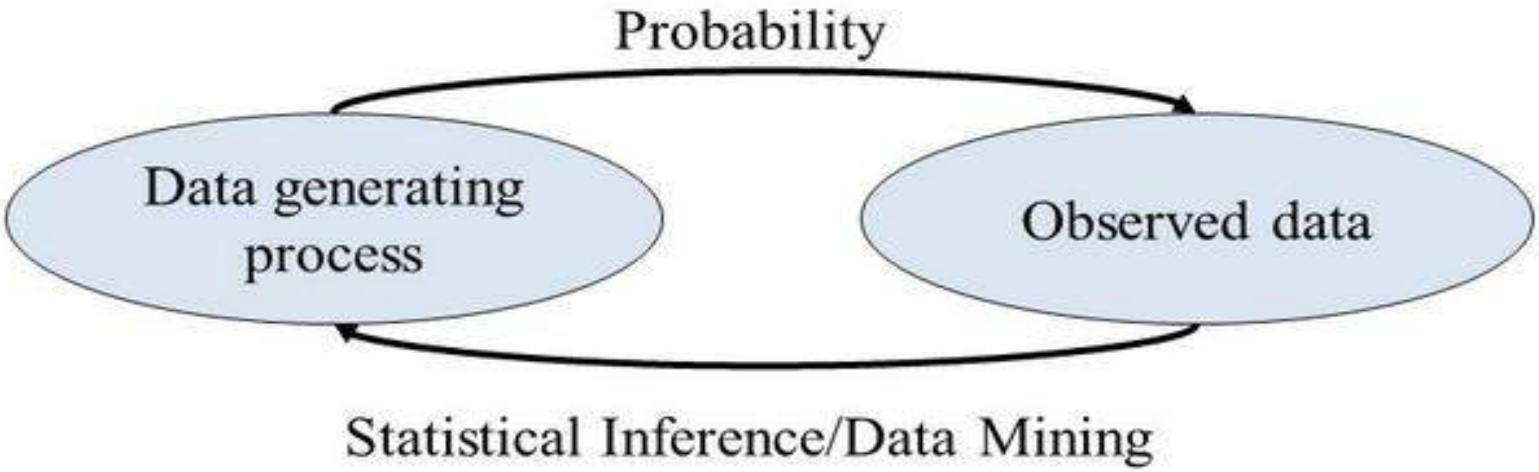
$p'(x) = \dfrac{2 - \log(x^2)}{x^2}$

$h'(x) = 3e^{3x}$

$q'(z) = 3(e^z - z)^2(e^z - 1)$

Probability Review

Probability Theory

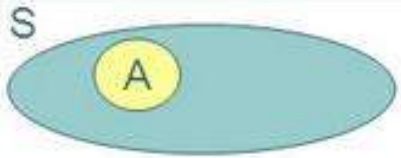


- **Probability Theory**
  - Given a data generating process, what are the properties of the outcome?
- **Statistical Inference**
  - Given the outcome, what can we say about the process that generated the data?
  - How can we generalize these observations and make predictions about future outcomes?

Probability Theory

Let A be an event, then we denote

$P(A)$ the probability for A



It always hold that $0 \leq P(A) \leq 1$ $P(\emptyset) = 0$ $P(S) = 1$

Consider an experiment which has N equally likely outcomes, and let exactly n of these events correspond to the event A. Then

$$P(A) = \frac{n}{N} = \frac{\text{\# successful outcomes}}{\text{\# possible outcomes}}$$

Example:
Rolling a dice

$P(\text{even number})$

$$= \frac{3}{6} = \frac{1}{2}$$

Random Variable

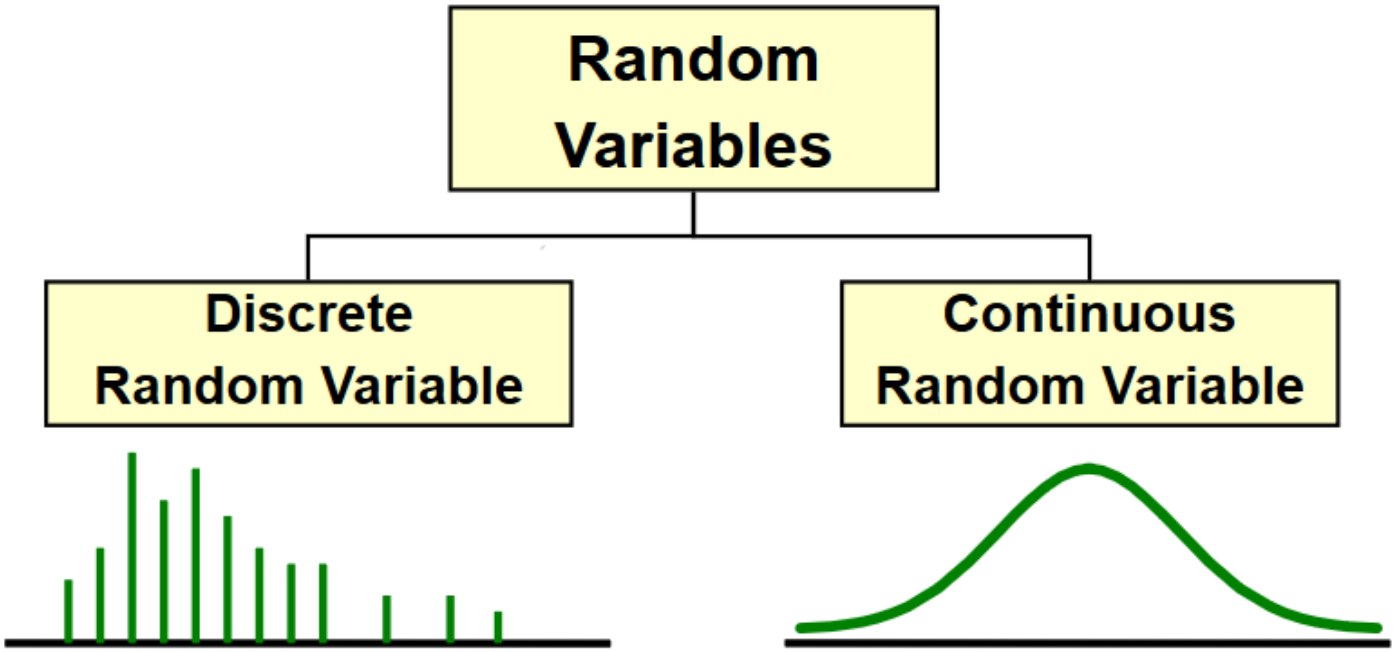
- A **random variable**, usually written X, is a **variable** whose possible values are numerical outcomes of a **random** phenomenon or experiment.

Examples

- ✔ X = number of heads when the experiment is flipping a coin 20 times.
- ✔ C = the daily change in a stock price.
- ✔ R = the number of kilometers per litter you get on your car during a family vacation.

Random Variable

Represents a possible numerical value from a random event



Random Variable

innovate achieve lead

Discrete Random Variable

- one that takes on a ***countable*** number of values
- usually count data [Number of]
- list **all** possible outcomes without missing any of them

Example:

- ✔ X = sum of values on the roll of two dice:
  X has to be either 2, 3, 4, …, or 12.

- ✔ Y = number of students in MTech DSE:
  Y has to be 60,65,70 ……

Random Variable

innovate achieve lead

Continuous Random Variable

- Variable that takes on an uncountable number of values
- Usually measurement data [time, weight, distance, etc]
- You can never list all possible outcomes even if you had an infinite amount of time

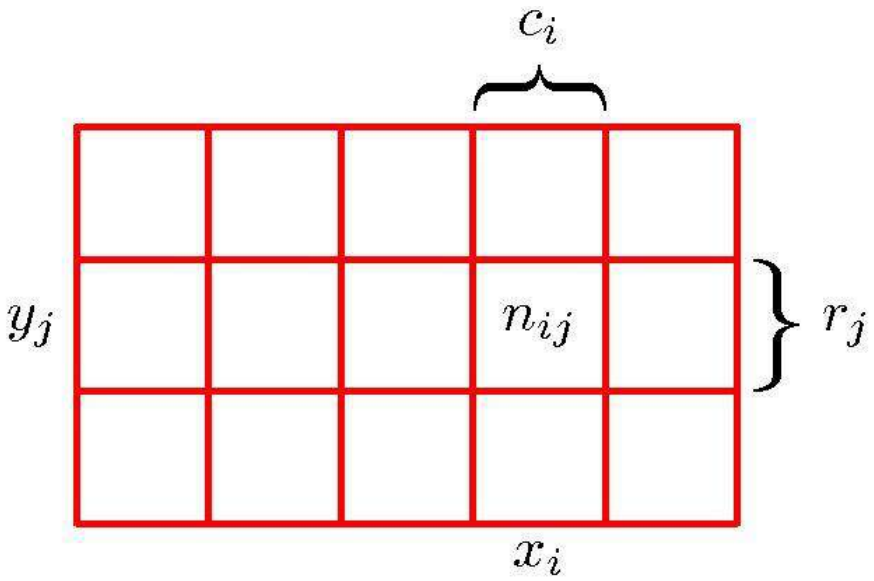
Example:

X = time it takes you to drive home from work place: X > 0, might be 30.1 minutes measured to the nearest tenth but in reality the actual time is 30.10000001……. minutes?)

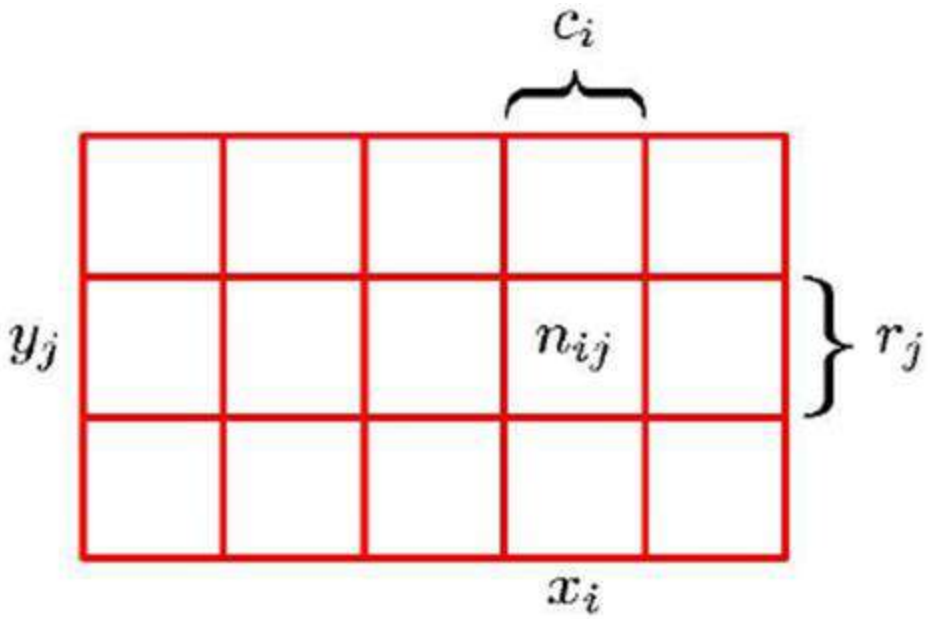
Exercise: try to list all possible numbers between 0 and 1

Probability Theory

- X and Y are two discrete random variables.
- X can take any of the values x_i, i = 1, . . . , M, and Y can take the values y_j, j = 1, . . . , L.
- Let a total of N trials in which we sample both of the variables X and Y, and let the number of such trials in which X = x_i and Y = y_j be n_{ij} .
- Let the number of trials in which X takes the value x_i be denoted by c_i, and similarly let the number of trials in which Y takes the value y_j be denoted by r_j .



Probability Theory



Marginal Probability

$$p(X = x_i) = \frac{c_i}{N}.$$

Joint Probability

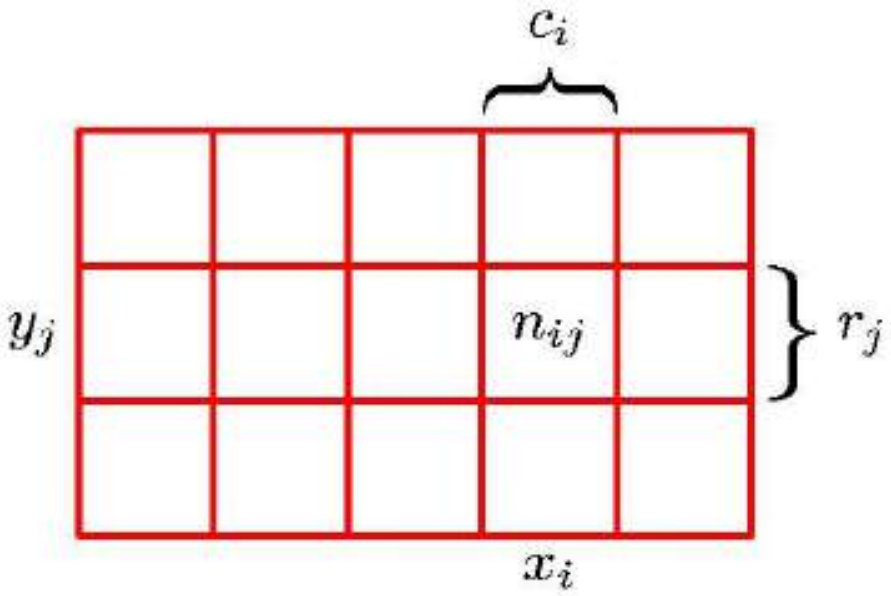
$$p(X = x_i, Y = y_j) = \frac{n_{ij}}{N}$$

Conditional Probability

$$p(Y = y_j | X = x_i) = \frac{n_{ij}}{c_i}$$

Here we are implicitly considering the limit $N \to \infty$.

Probability Theory



Sum Rule

$$p(X = x_i) = \frac{c_i}{N} = \frac{1}{N}\sum_{j=1}^{L} n_{ij}$$

$$= \sum_{j=1}^{L} p(X = x_i, Y = y_j)$$

Product Rule

$$p(X = x_i, Y = y_j) = \frac{n_{ij}}{N} = \frac{n_{ij}}{c_i} \cdot \frac{c_i}{N}$$

$$= p(Y = y_j | X = x_i)p(X = x_i)$$

Probability Theory

Notation

- A **random variable X** represents outcomes or states of the world.
- We will write p(x) to mean Probability(X = x)
- **Sample space**: the space of all possible outcomes (may be discrete, continuous, or mixed)
- p(x) is the **probability mass (density) function**
  - Assigns a number to each point in sample space
  - Non-negative, sums (integrates) to 1
  - Intuitively: how often does x occur, how much do we believe in x.

Probability Theory

Joint Probability Distribution

- Prob(X=x, Y=y)
  - "Probability of X=x and Y=y"
  - p(x, y)

Conditional Probability Distribution

- Prob(X=x|Y=y)
  - "Probability of X=x given Y=y"
  - p(x|y) = p(x,y)/p(y)

Probability Theory

The Rules of Probability

- Sum Rule (marginalization/summing out):

$$p(x) = \sum_{y} p(x, y)$$

$$p(x_1) = \sum_{x_2} \sum_{x_3} \dots \sum_{x_N} p(x_1, x_2, \dots, x_N)$$

- Product/Chain Rule:

$$p(x, y) = p(y \mid x)\, p(x)$$

$$p(x_1, \dots, x_N) = p(x_1)\, p(x_2 \mid x_1) \dots p(x_N \mid x_1, \dots, x_{N-1})$$

Probability Theory

from the definition of conditional distributions:

$$P(A|B)P(B) = P(A, B) = P(B|A)P(A)$$

Hence:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

is known as Bayes rule.

example:

$$P(\text{“taking a shower”}|\text{“wet”}) = P(\text{“wet”}|\text{“taking a shower”})\frac{P(\text{“taking a shower”})}{P(\text{“wet”})}$$

$$P(\text{reason}|\text{observation}) = P(\text{observation}|\text{reason})\frac{P(\text{reason})}{P(\text{observation})}$$

Probability Theory

innovate achieve lead

Bayes rule - Example

if patient has meningitis, then very often a stiff neck is observed

if patient has meningitis, then very often a stiff neck is observed

$P(S|M) = 0.8$ (can be easily determined by counting)

observation: 'I have a stiff neck! Do I have meningitis?' (is it reasonable to be afraid?)

Probability Theory

Bayes rule - Example

if patient has meningitis, then very often a stiff neck is observed
$P(S|M) = 0.8$ (can be easily determined by counting)

observation: 'I have a stiff neck! Do I have meningitis?' (is it reasonable to be afraid?)
$P(M|S) = ?$

we need to now: $P(M) = 0.0001$ (one of 10000 people has meningitis)
and $P(S) = 0.1$ (one out of 10 people has a stiff neck).

Probability Theory

if patient has meningitis, then very often a stiff neck is observed
$P(S|M) = 0.8$ (can be easily determined by counting)

observation: 'I have a stiff neck! Do I have meningitis?' (is it reasonable to be afraid?)
$P(M|S) = ?$

we need to now: $P(M) = 0.0001$ (one of 10000 people has meningitis) and $P(S) = 0.1$ (one out of 10 people has a stiff neck).

then:

$$P(M|S) = \frac{P(S|M)P(M)}{P(S)} = \frac{0.8 \times 0.0001}{0.1} = 0.0008$$

Keep cool. Not very likely

Probability Densities

Continuous Probability Distribution

Let X be a continuous rv. Then a *probability distribution or probability density function (pdf)* of X is a function $f(x)$ such that for any two numbers a and b,

$$P(a \le X \le b) = \int_a^b f(x)dx$$

The graph of f is the *density curve*.

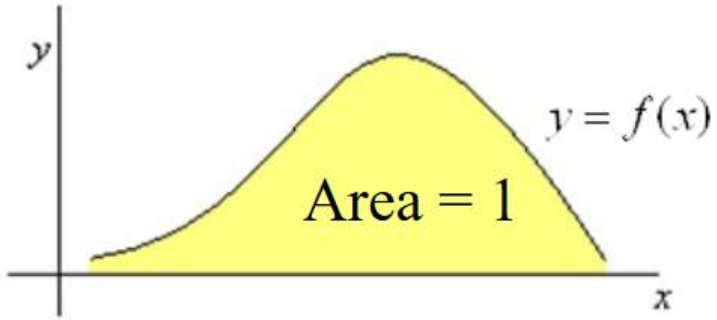
Probability Densities

Probability Density Function

For $f(x)$ to be a pdf

1. $f(x) > 0$ for all values of x.

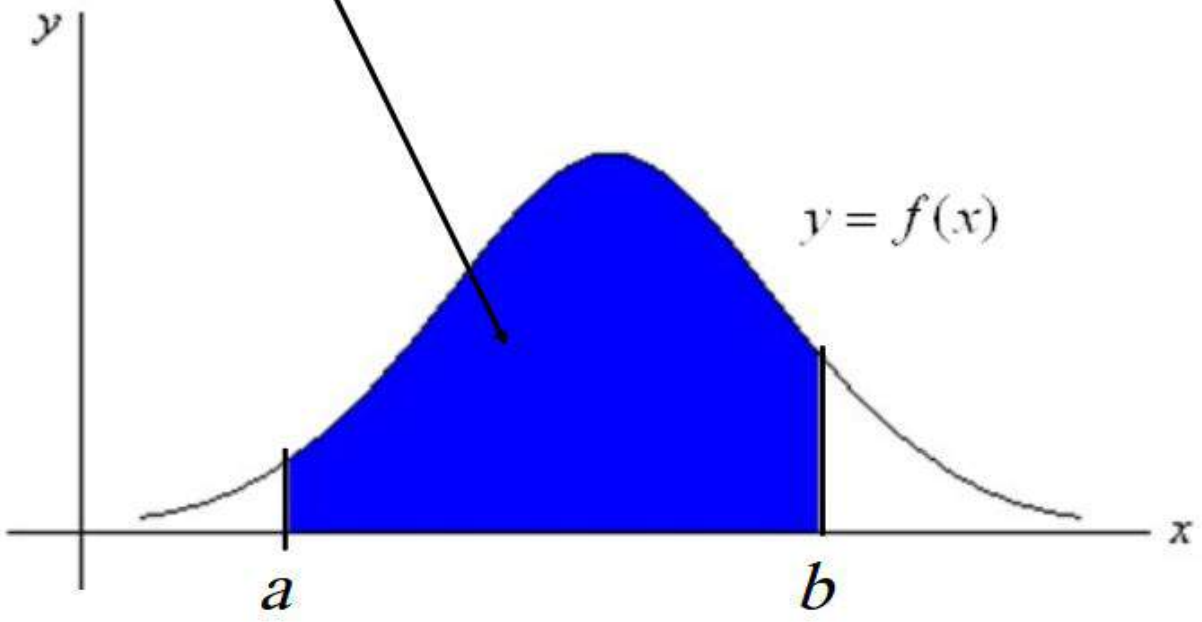
2. The area of the region between the graph of f and the $x-$ axis is equal to 1.



Probability Densities

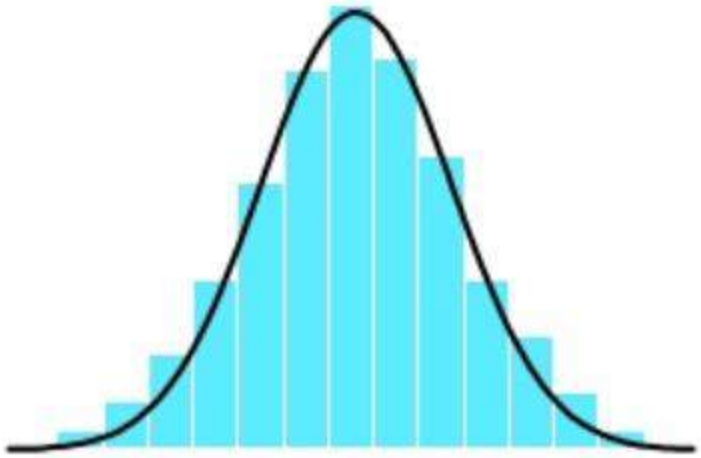
Probability Density Function

$P(a \leq X \leq b)$ is given by the area of the shaded region.



Gaussian Distribution

- The commonest and the most useful continuous distribution.
- A symmetrical probability distribution where most results are located in the middle and few are spread on both sides.
- It has the shape of a bell.
- Can entirely be described by its mean and standard deviation.



Gaussian Distribution

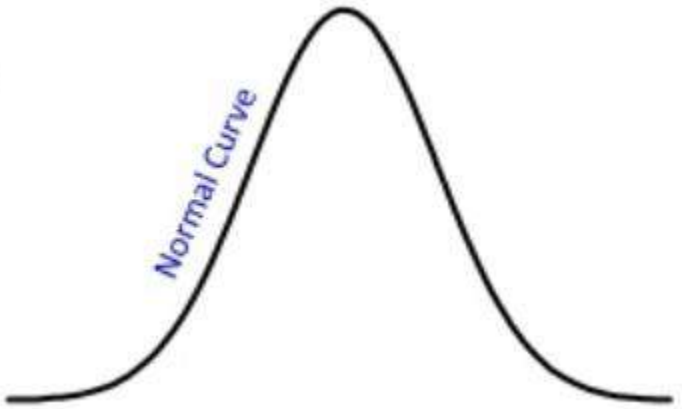
Examples:

- The body temperature for healthy humans.
- The heights and weights of adults.
- The thickness and dimensions of a product.
- IQ and standardized test scores.
- Quality control test results.
- Errors in measurements.

Gaussian Distribution

Normal Curve:

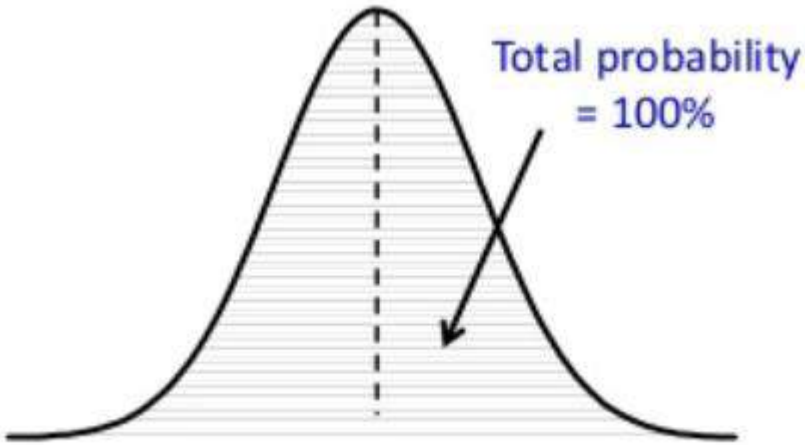
- A graphical representation of the normal distribution.
- It is determined by the mean and the standard deviation.
- It a symmetric unimodal bell-shaped curve.
- Its tails extending infinitely in both directions.
- The wider the curve, the larger the standard deviation and the more variation exists in the process.
- The spread of the curve is equivalent to six times the standard deviation of the process.



Gaussian Distribution

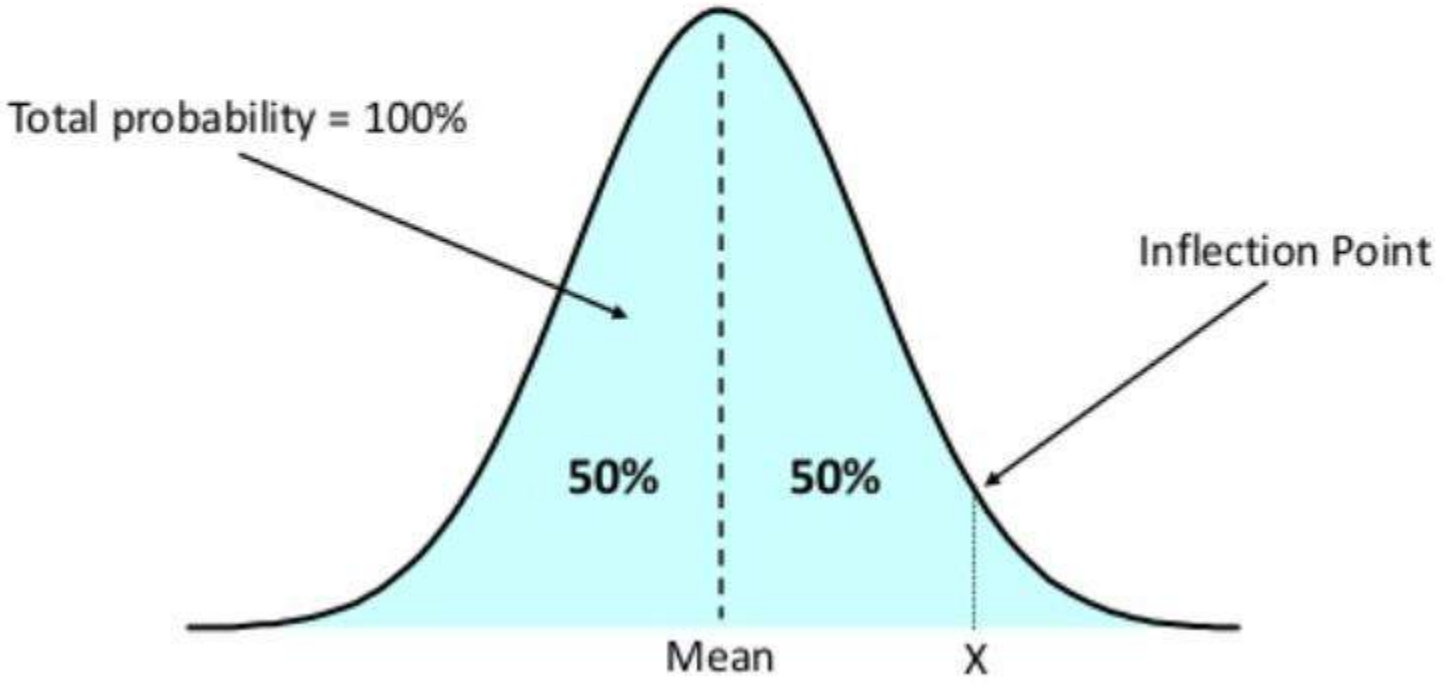
Normal Curve:

- Helps calculating the probabilities for normally distributed populations.
- The probabilities are represented by the area under the normal curve.
- The total area under the curve is equal to **100%** (or **1.00**).
- This represents the population of the observations.
- We can get a rough estimate of the probability above a value, below a value, or between any two values.



Gaussian Distribution

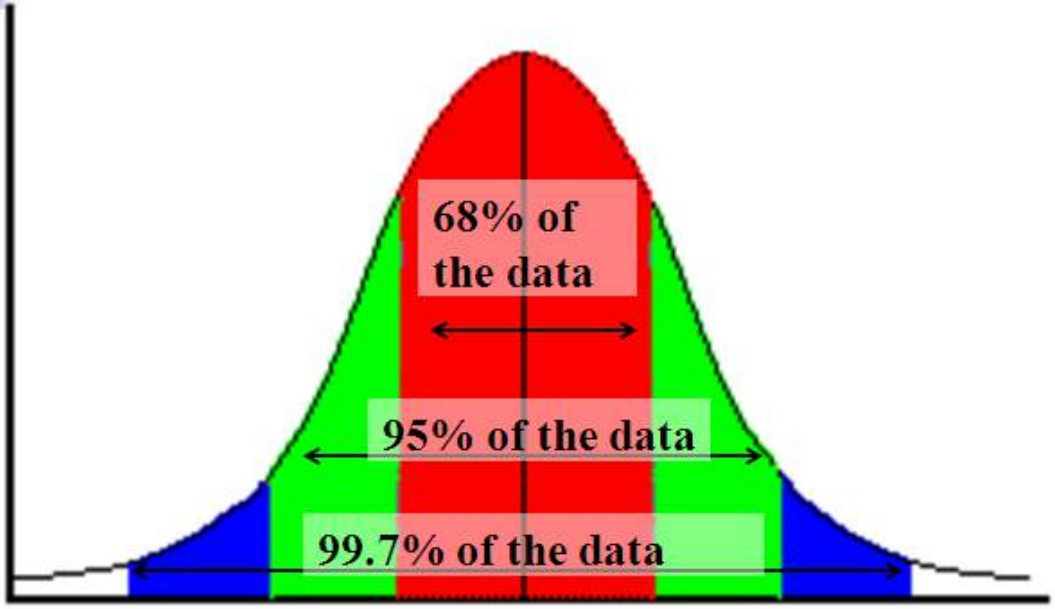
- Since the normal curve is symmetrical, **50 percent** of the data lie on each side of the curve.



Gaussian Distribution

Empirical Rule:

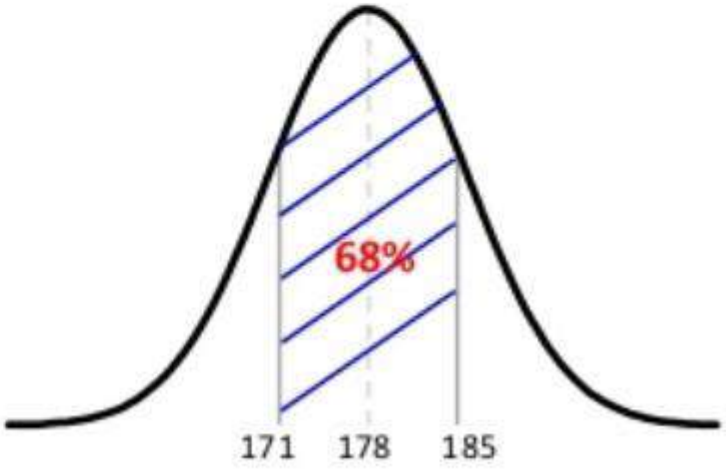
- For any normally distributed data:
  - **68%** of the data fall within **1** standard deviation of the mean.
  - **95%** of the data fall within **2** standard deviations of the mean.
  - **99.7%** of the data fall within **3** standard deviations of the mean.



Gaussian Distribution

- Suppose that the heights of a sample men are normally distributed.
- The mean height is **178** cm and a standard deviation is **7** cm.

- **We can generalize that:**
  - **68%** of population are between **171** cm and **185** cm.
  - This might be a generalization, but it's true if the data is normally distributed.



Mean, Variance & Standard Deviation

- The mean of a discrete random variable is the ***weighted average*** of all of its values. The weights are the probabilities.

- This parameter is also called the expected value of X and is represented by E(X).

$$E(X) = \mu = \sum_{all\ x} xP(x)$$

- The variance is

$$V(X) = \sigma^2 = \sum_{all\ x} (x-\mu)^2 P(x)$$

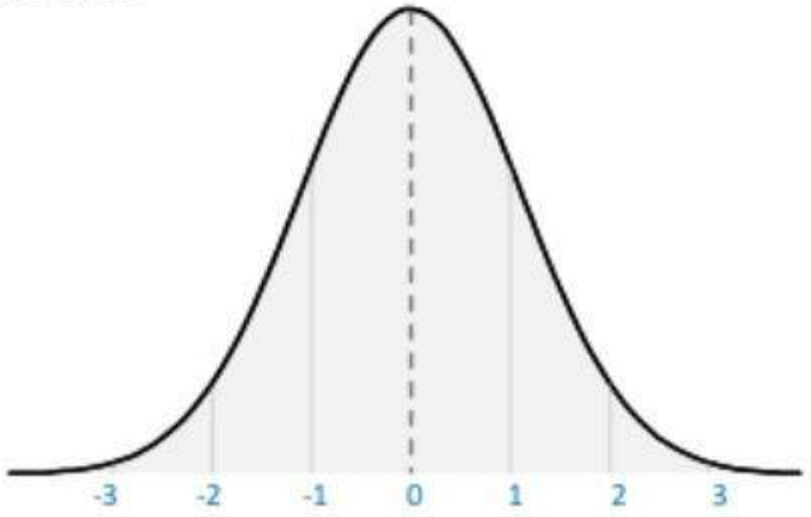
- The standard deviation is

$$\sigma = \sqrt{\sigma^2}$$

Gaussian Distribution

Standard Normal Distribution:

- A common practice to convert any normal distribution to the standardized form and then use the standard normal table to find probabilities.
- The **Standard Normal Distribution** (Z distribution) is a way of standardizing the normal distribution.
- It always has a mean of **0** and a standard deviation of **1**.



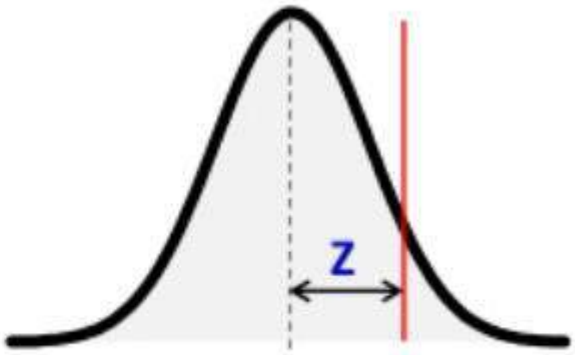
Gaussian Distribution

Standard Normal Distribution:

- Any normally distributed data can be converted to the standardized form using the formula:

$$Z = (X - \mu) / \sigma$$

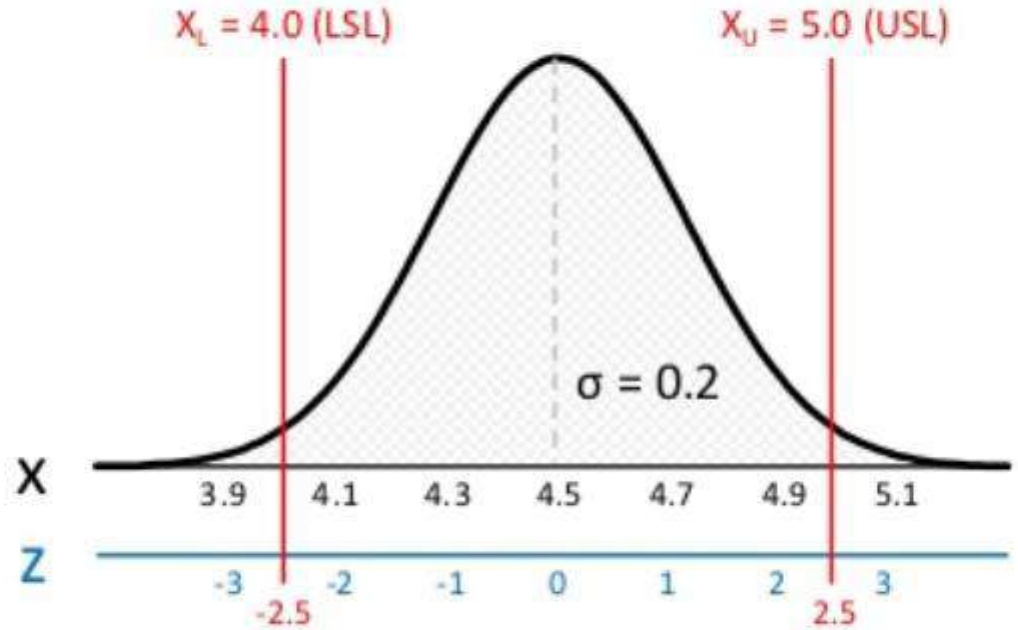
- where:
  - 'X' is the data point in question.
  - 'Z' (or **Z-score**) is a measure of the number of standard deviations of that data point from the mean.



Gaussian Distribution

Standard Normal Distribution:

- Converting from 'X' to 'Z':



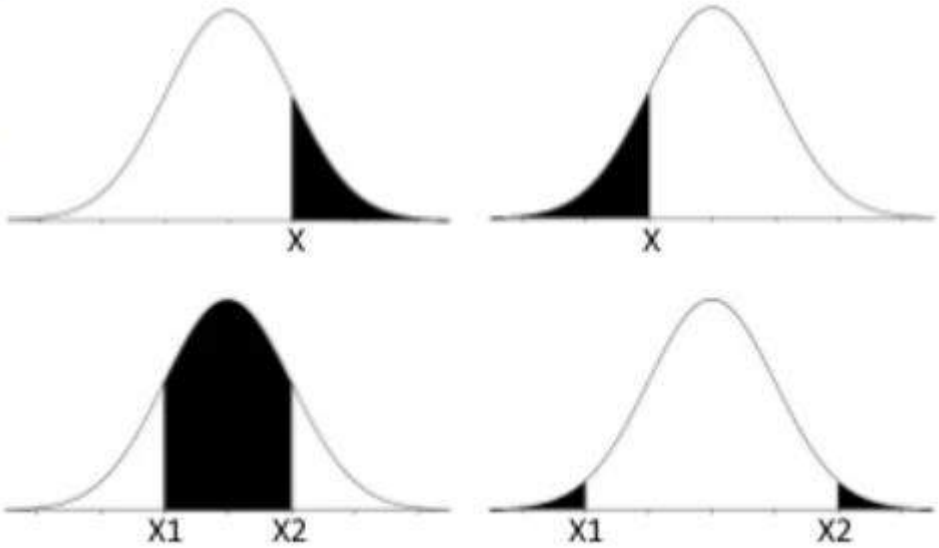
The specification limits at 4.0 and 5.0mm respectively, and the corresponding z values

The X-scale is for the actual values and the Z-scale is for the standardized values

Gaussian Distribution

Standard Normal Distribution:

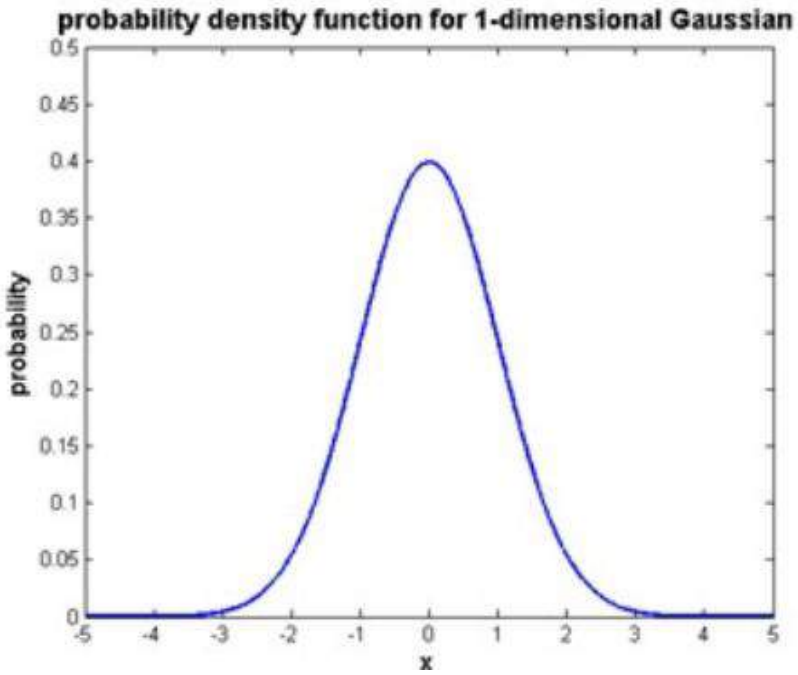
- You can then use this information to determine the area under the normal distribution curve that is:
  - To the right of your data point.
  - To the left of the data point.
  - Between two data points.
  - Outside of two data points.



Gaussian Distribution

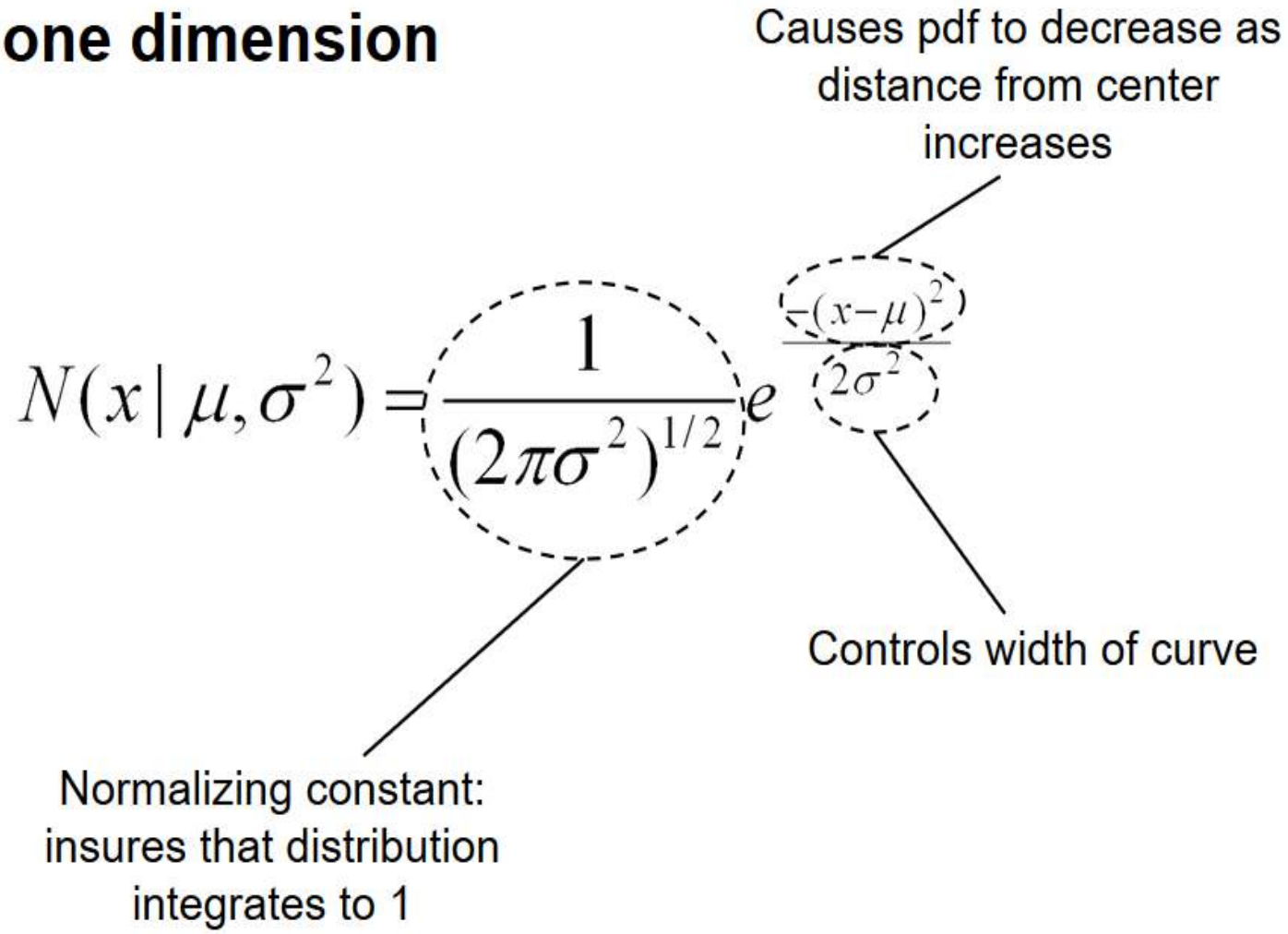
In one dimension

$$N(x \mid \mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

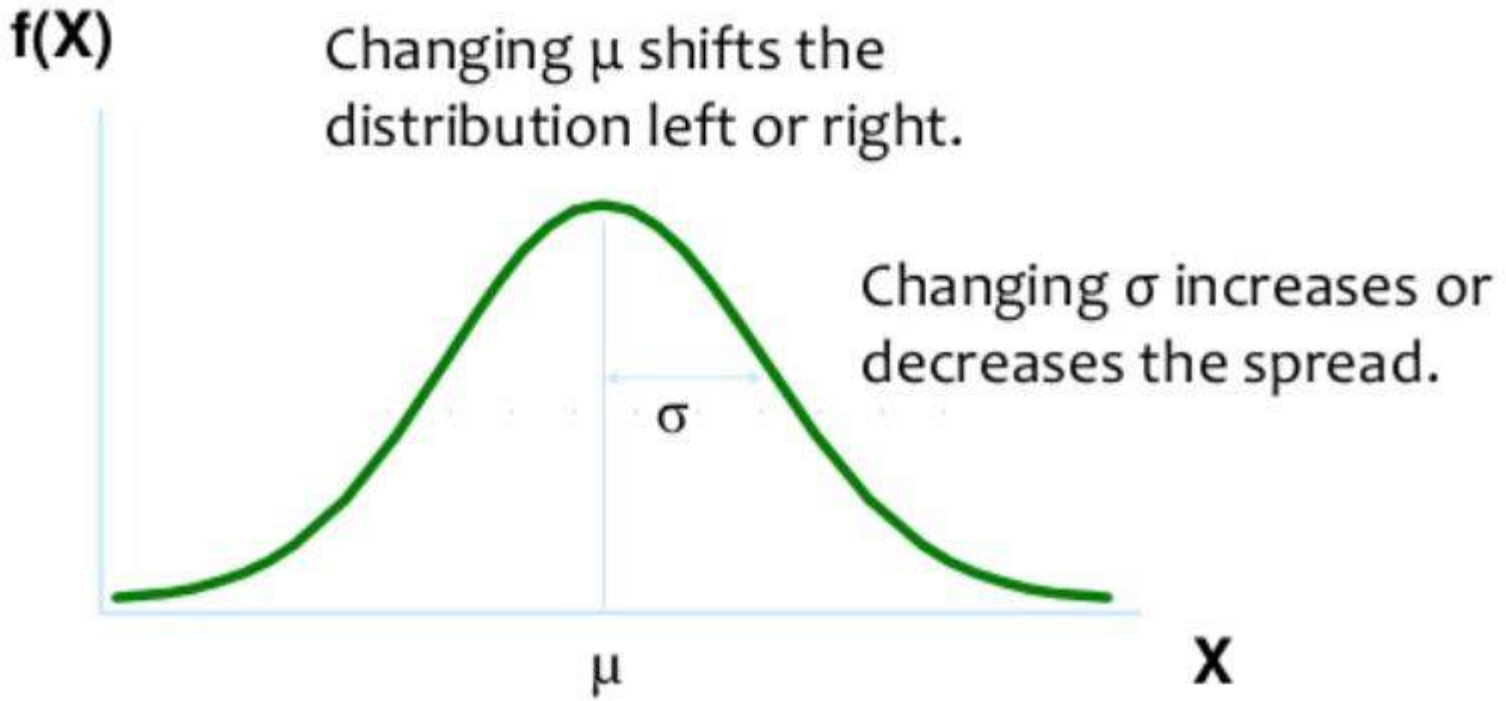


Gaussian Distribution

In one dimension

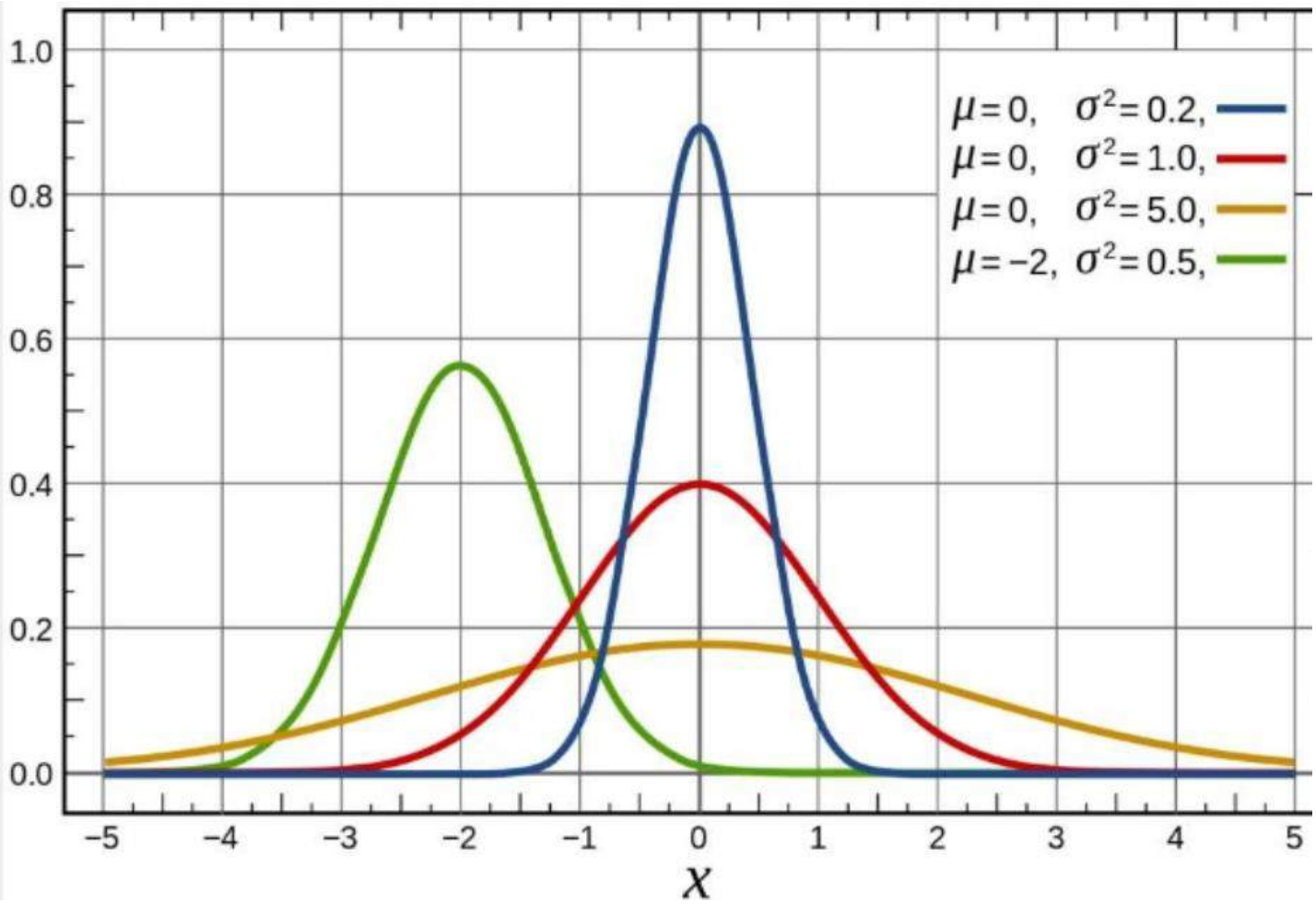


Gaussian Distribution



The normal curve is not a single curve but a family of curves, each of which is determined by its mean and standard deviation.

Gaussian Distribution



Multivariate Gaussian Distribution

In *d* dimensions

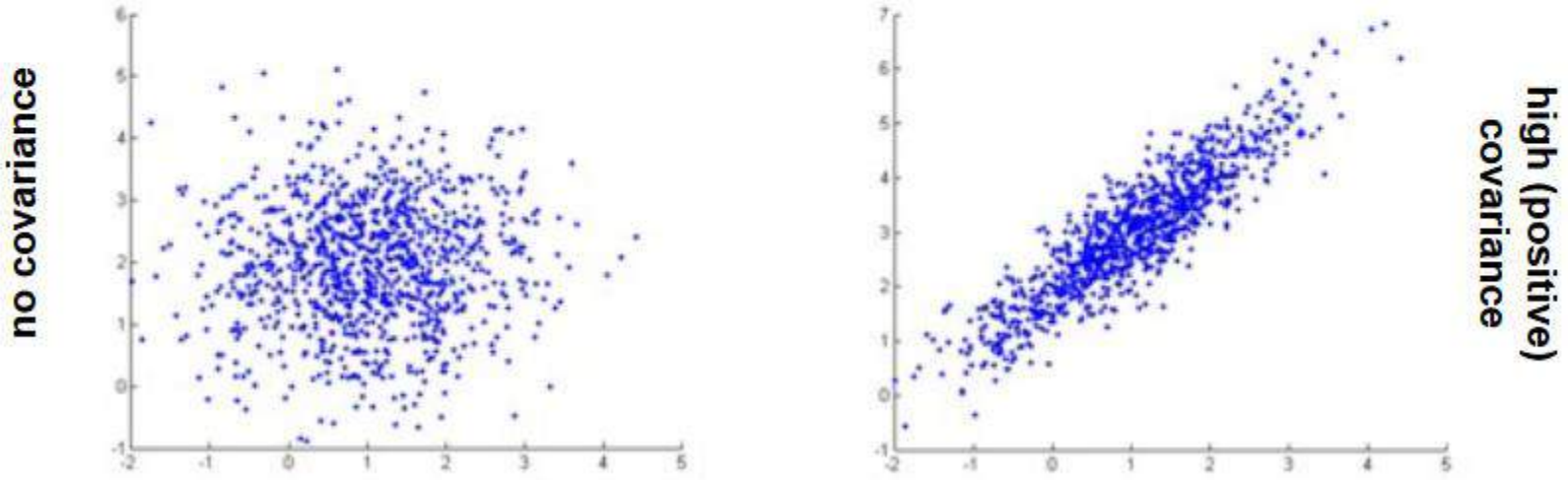
$$N(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

- **x** and μ now *d*-dimensional vectors
  - μ gives center of distribution in *d*-dimensional space
- σ^2 replaced by Σ, the *d* x *d* covariance matrix
  - Σ contains pairwise covariances of every pair of features
  - Diagonal elements of Σ are variances σ^2 of individual features
  - Σ describes distribution's shape and spread

Multivariate Gaussian Distribution

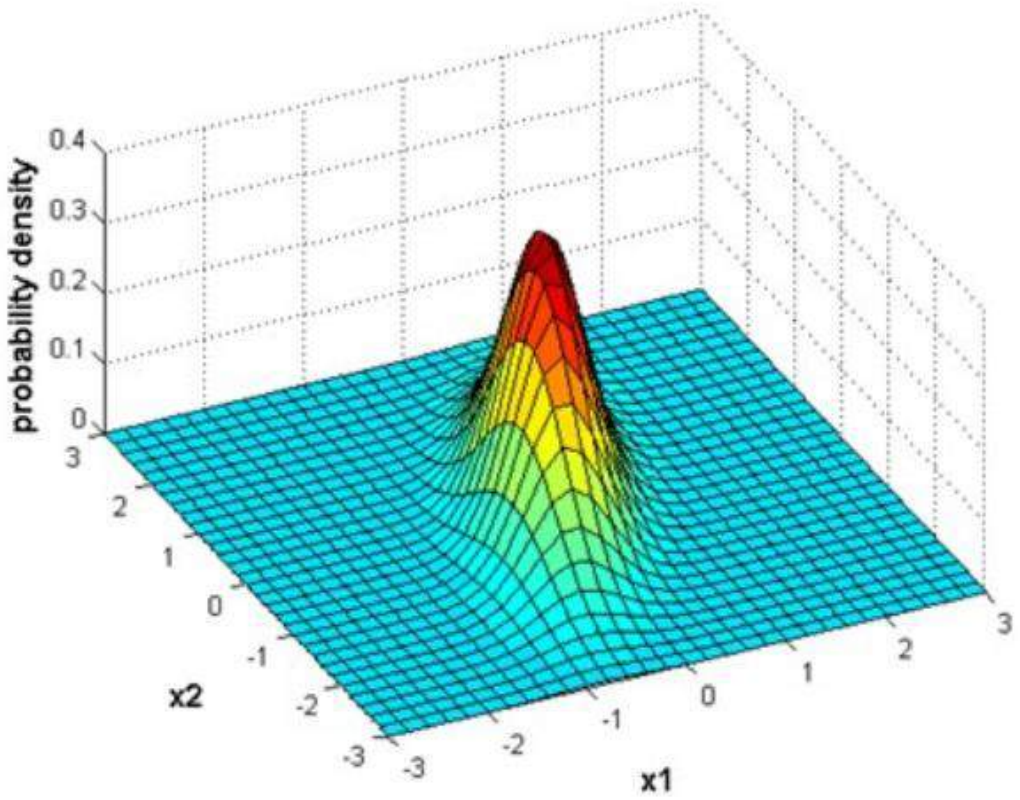
Covariance

- Measures tendency for two variables to deviate from their means in same (or opposite) directions at same time



Multivariate Gaussian Distribution

In two dimensions

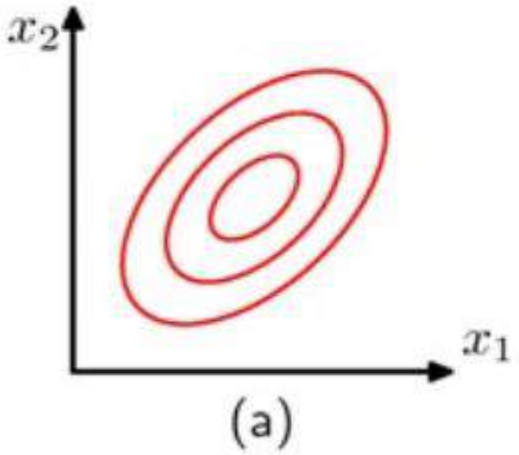


$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

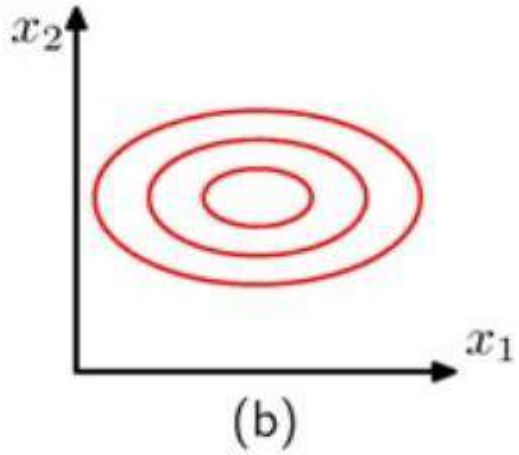
$$\boldsymbol{\Sigma} = \begin{bmatrix} 0.25 & 0.3 \\ 0.3 & 1 \end{bmatrix}$$

Multivariate Gaussian Distribution

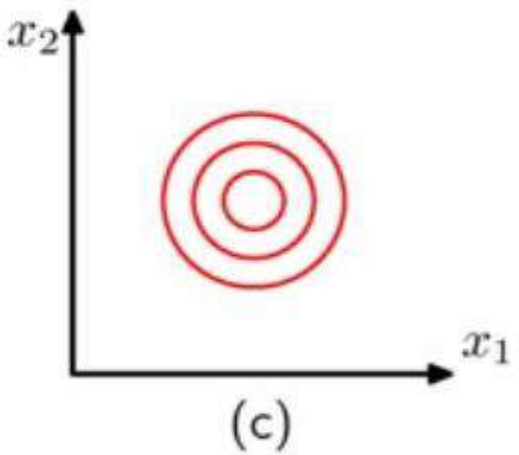
In two dimensions



(a)



(b)



(c)

$$\Sigma = \begin{bmatrix} 2 & 0.6 \\ 0.6 & 2 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Decision Theory

Decision Theory

- Suppose **x** is an input vector together with a corresponding vector **t** of target variables
- Goal: predict **t** given a new value for **x.**
- The joint probability distribution ***p(x, t)*** provides a complete summary of the uncertainty associated with these variables.
- Determination of ***p(x, t)*** from a set of training data is called ***inference*** and is a difficult problem.

Decision Theory

Inference step

Determine either $p(t|\mathbf{x})$ or $p(\mathbf{x}, t)$.

Decision step

For given $\mathbf{x}$, determine optimal t.

Example : Medical diagnosis problem

Input: X-ray image of a patient
Input vector x is the set of pixel intensities in the image

Output: Presence of cancer = Class C1,
Absence of cancer, = Class C2.
Choose t to be a binary variable such that
t =0 corresponds to C1 and t =1 corresponds to C2.

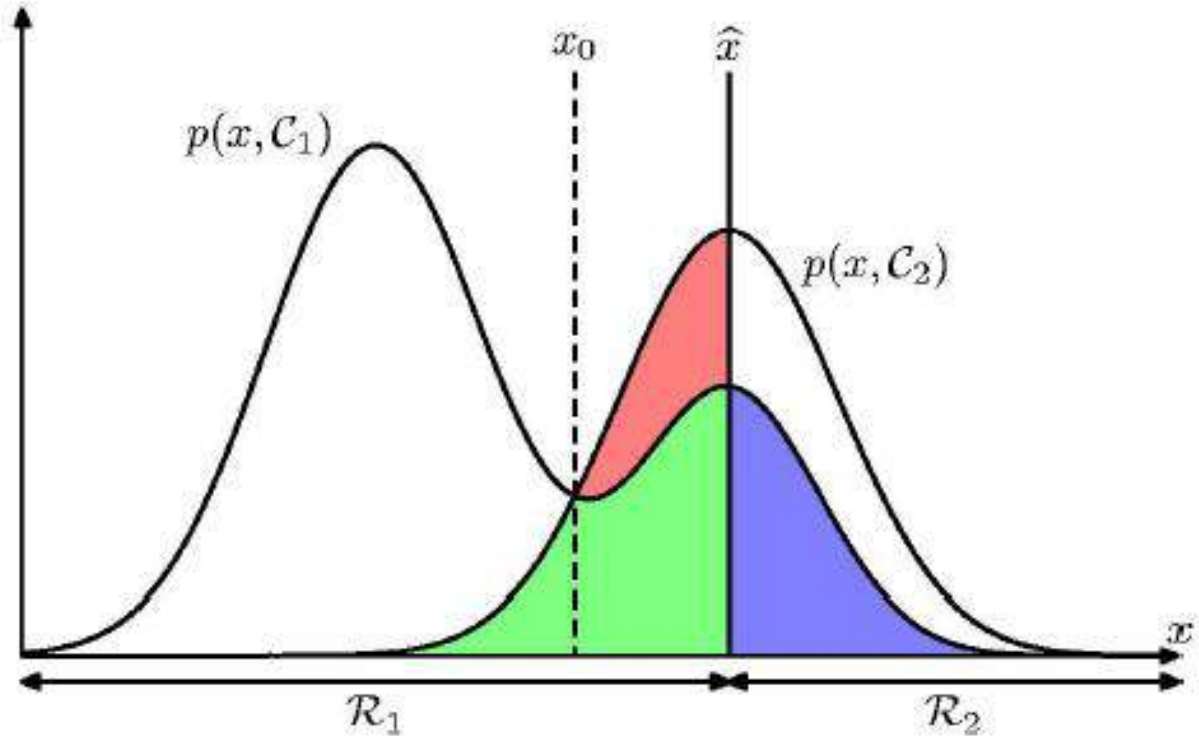
We are interested in the probabilities of the two classes given the image, which are given by $p(C_k|x)$.
Using Bayes' theorem,

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})}$$

Minimum Misclassification Rate

- Divide the input space into regions Rk called decision regions, one for each class, such that all points in Rk are assigned to class Ck
- Boundaries between decision regions are called decision boundaries or decision surfaces
- A mistake occurs when an input vector belonging to class C1 is assigned to class C2 or vice versa.

Minimum Misclassification Rate



$$
\begin{aligned}
p(\text{mistake}) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \\
&= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2)\, \mathrm{d}\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1)\, \mathrm{d}\mathbf{x}.
\end{aligned}
$$

Minimum Misclassification Rate

$$p(\text{correct}) = \sum_{k=1}^{K} p(\mathbf{x} \in \mathcal{R}_k, \mathcal{C}_k)$$

$$= \sum_{k=1}^{K} \int_{\mathcal{R}_k} p(\mathbf{x}, \mathcal{C}_k)\, \mathrm{d}\mathbf{x}$$

Inference and Decision

- We have broken the **classification problem** down into two separate stages, the ***inference stage*** in which we use training data to learn a model for $p(C_k|x)$, and the subsequent ***decision stage*** in which we use these posterior probabilities to make optimal class assignments.
- An alternative possibility would be to solve both problems together and simply learn a function that maps inputs x directly into decisions. Such a function is called a ***discriminant function***.
- **Three distinct approaches to solving decision problems**

Inference and Decision

1st approach

- Determine the class-conditional densities $p(x \mid C_k)$ for each class C_k individually.
- Separately infer the prior class probabilities $p(C_k)$. Then use Bayes' theorem

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})}$$

$$p(\mathbf{x}) = \sum_k p(\mathbf{x}|C_k)p(C_k)$$

Inference and Decision

2nd approach

- Solve the inference problem of determining the posterior class probabilities $p(C_k|x)$, and then subsequently use decision theory to assign each new x to one of the classes.
- Approaches that model the posterior probabilities directly are called ***discriminative models***.

Inference and Decision

3rd approach

- Find a function ***f(x)***, called a ***discriminant function***, which maps each input ***x*** directly onto a class label.
- Example - For two-class problems, ***f(·)*** might be binary valued and such that ***f = 0*** represents class C_1 and ***f = 1*** represents class C_2. In this case, probabilities play no role.

Inference and Decision

- 1st approach is the most demanding because it involves finding $p(x,C_k)$.
- For many applications, x will have high dimensionality, and consequently we may need a large training set in order to be able to determine the class-conditional densities to reasonable accuracy.
- The class priors $p(C_k)$ can often be estimated simply from the fractions of the training set data points in each of the classes.

Inference and Decision

- One advantage of 1st approach, however, is that it also allows **p(x)** to be determined. This can be useful for detecting new data points that have low probability under the model and for which the predictions may be of low accuracy, which is known as ***outlier detection*** or ***novelty detection.***

$$p(\mathbf{x}) = \sum_k p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)$$

Inference and Decision

- If we only wish to make classification decisions, then it can be wasteful of computational resources, and excessively demanding of data, to find $p(x, C_k)$ when in fact we only really need the posterior probabilities $p(C_k|x)$, which can be obtained directly through 2nd approach.

Inference and Decision

- The 3rd approach is simple, where the data is used to find a ***discriminant function f(x)*** that maps each ***x*** directly onto a class label, thereby combining the inference and decision stages into a single learning problem.

Information Theory

Measure of Information

- It rained heavily in Shillong yesterday
- There was a heavy rainfall in Rajasthan last night.
- The amount of information conveyed by a message is inversely proportional to its probability of occurrence. That is

$$I_k \alpha \frac{1}{p_k}$$

Measure of Information

- The information conveyed by a message cannot be negative. It has to be at least 0, i.e.,

$$I_k \geq 0$$

- If $p_k = 1$, then $I_k = 0$.
- The information conveyed composite statement which is independent is simply given by the sum of the individual self-information contents, i.e.,

$$I(m_1, m_2) = I(m_1) + I(m_2)$$

Measure of Information

- The only mathematical operator satisfies above properties is the logarithmic operator. Therefore,

$$I_k = \log_r \frac{1}{p_k} \; units$$

- If r = 2, unit is bits

Average Information Content (Entropy)

- Consider a source x = {x_1, x_2, ..., x_N}, with probabilities P = {p_1, p_2, ..., p_N}
- A message of length L emitted by the source, then it contains
- p_1 x L number of symbol x_1
- p_2 x L number of symbol x_2, ...,
- p_N x L number of symbol x_N

Average Information Content (Entropy)

- The self-information of x_1 is

$$I_{x_1} = \log_2 \frac{1}{p_1} \; bits$$

- Each symbol x_1 conveys information of $\log_2(1/p_1)$ bits and such (p_1 x L) number of x_1 symbols are present on an average in a length of L symbols.
- The total information conveyed by symbols of type x_i is

$$p_i \times L \times \log_2 \frac{1}{p_i} \; bits$$

Average Information Content (Entropy)

- The total information conveyed by the source is

$$I_T = p_1 \times L \times \log_2 \frac{1}{p_1} + p_2 \times L \times \log_2 \frac{1}{p_2} + \cdots p_N \times L \times \log_2 \frac{1}{p_N}$$

- The average information conveyed by the source by emitting L symbols is denoted by its entropy H[x]

$$H[x] = \frac{I_T}{L} = p_1 \times \log_2 \frac{1}{p_1} + p_2 \times \log_2 \frac{1}{p_2} + \cdots p_N \times \log_2 \frac{1}{p_N}$$

Entropy

$$\mathrm{H}[x] = -\sum_{x} p(x) \log_2 p(x)$$

Important quantity in

- coding theory
- statistical physics
- machine learning

Entropy

Consider a discrete random variable **x** with 8 possible states, each of which is equally likely,

How many bits to transmit the state of **x**?

$$\mathrm{H}[x] = -8 \times \frac{1}{8} \log_2 \frac{1}{8} = 3 \text{ bits.}$$

Conditional Entropy

- Suppose we have a joint distribution ***p(x, y)*** from which we draw ***x*** and ***y***.
- If ***x*** is already known, then the additional information needed to specify ***y*** is given by ***–ln p(y|x)***.
- Thus the average additional information needed to specify ***y*** can be written as

$$\mathrm{H}[\mathbf{y}|\mathbf{x}] = -\iint p(\mathbf{y},\mathbf{x}) \ln p(\mathbf{y}|\mathbf{x})\, \mathrm{d}\mathbf{y}\, \mathrm{d}\mathbf{x}$$

$$\mathrm{H}[\mathbf{x},\mathbf{y}] = \mathrm{H}[\mathbf{y}|\mathbf{x}] + \mathrm{H}[\mathbf{x}]$$

Thank You

Machine Learning
DSECL ZG565

Dr. Chetana Gavankar, Ph.D,
IIT Bombay-Monash University Australia
Chetana.gavankar@pilani.bits-pilani.ac.in

BITS Pilani
Pilani Campus

innovate achieve lead

BITS Pilani

Pilani Campus

Lecture No. – 3 | Bayesian Learning

Date – 09/11/2019

Time – 9:00 AM – 11:00 AM

Session Content

Bayesian learning

- Basics (T1 book by Tom Mitchell - 6.1)
- Bayes Theorem (T1 book by Tom Mitchell - 6.2)
- MAP Hypothesis (T1 book by Tom Mitchell - 6.3)
- MLE Hypothesis (T1 book by Tom Mitchell - 6.4)
- Minimum Description Length (MDL) principle

(T1 book by Tom Mitchell - 6.6)

Bayesian Learning

- Bayesian learning algorithms that calculate explicit probabilities for hypotheses, such as the naive Bayes classifier, are among the most practical approaches to certain types of learning problems

- For example: Problem of learning to classify text documents such as electronic news articles.
- For such learning tasks, the naive Bayes classifier is among the most effective algorithms known

Features of Bayesian learning

- Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct.
- Flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example.
- Bayesian methods can accommodate hypotheses that make probabilistic predictions (e.g., hypotheses such as "this pneumonia patient has a 93% chance of complete recovery").

Features of Bayesian learning

- Prior knowledge can be combined with observed data to determine the final probability of a hypothesis.
- Prior knowledge is provided by asserting
  - prior probability for each candidate hypothesis, and
  - probability distribution over observed data for each possible hypothesis.
- New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.

Practical Issues

- Require initial knowledge of many probabilities
  - Often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions.
- Significant computational cost required to determine the Bayes optimal hypothesis in the general case (linear in the number of candidate hypotheses)

Probability Theory

- Classical Definition
  - Consider an experiment which has N equally likely outcomes, and let n of these outcomes correspond to a specific event A, then
  - P(A) = $\frac{\#\ successful\ outcomes}{\#\ possible\ outcomes}$ = n / N

Ω = {  ,  } Coin toss

Ω = {  } Die toss

Images taken from "Probabilistic Graphical Models" by David Sontag

AXIOMS of Probability Theory

- A probability p(ω) for each outcome ω must satisfy the following axioms

$$p(\omega) \geq 0, \quad \sum_{\omega \in \Omega} p(\omega) = 1$$

E.g., p() = .6

p() = .4

Images taken from "Probabilistic Graphical Models" by David Sontag

Random Variable

Notation

- A **random variable X** represents outcomes or states of the world.
- We will write p(x) to mean Probability(X = x)
- **Sample space**: the space of all possible outcomes (may be discrete, continuous, or mixed)
- p(x) is the **probability mass (density) function**
  - Assigns a number to each point in sample space
  - Non-negative, sums (integrates) to 1
  - Intuitively: how often does x occur, how much do we believe in x.

BITS Pilani

Probability Distributions

- The outcomes for random variables and their associated probabilities can be organized in to distributions
- Two types of distributions based on types of Random variables: Discrete and Continuous
- Discrete:
  - Binomial, Poisson, Geometric distributions
- Continuous
  - Gaussian, exponential, t, F, chi-squared distibutions

Describing distributions

- One way is to construct a graph and analyze the graph to make inferences
  - Discrete: Prob Mass Function (pmf), Cumulative density function
  - Continuous: prob density function (pdf)
- Mean, variance and standard deviations to represent the entire distribution

Bernoulli Distribution

- A r.v. X is said to follow Bernoulli's distribution when there are only two possible outcomes
  - By convention either Success (1) or Failure (0)
  - And there is only one trial
  - Ex: tossing a coin at the start f the match
- Let p represents the probability of success and (1-p) represent the probability of failure, then the probability mass function is defined as

$$f(x) = \begin{cases} p & \text{if } x=1 \\ 1-p & x=0 \end{cases}$$

$$p^x (1-p)^{(1-x)}$$

Binomial Distribution

- Again binary outcomes, but for n independent trials
  - Probability of success (p) remains the same for all the trials
  - Probability of r success is given by

$P(X=r) = {}^{n}C_{r}\, p^{r}\, (1-p)^{n-r}$

Mean $E(X) = np$

Variance $Var(X) = npq$ (where $q=1-p$)

Estimate Probabilities from Data

- For continuous attributes:
  - Probability density estimation:
    - Assume attribute follows a normal distribution
    - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
    - Once probability distribution is known, use it to estimate the conditional probability $P(X_i|Y)$

Probability Densities

Continuous Probability Distribution

Let X be a continuous rv. Then a *probability distribution or probability density function (pdf)* of X is a function $f(x)$ such that for any two numbers a and b,

$$P(a \le X \le b) = \int_a^b f(x)dx$$

The graph of f is the *density curve*.

BITS Pilani

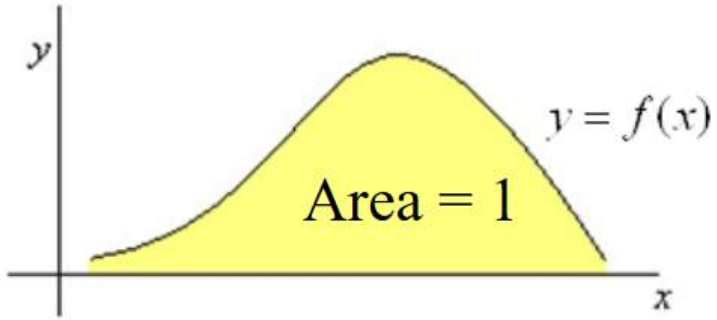
Probability Densities

Probability Density Function

For $f(x)$ to be a pdf

1. $f(x) > 0$ for all values of x.

2. The area of the region between the graph of f and the $x-$ axis is equal to 1.

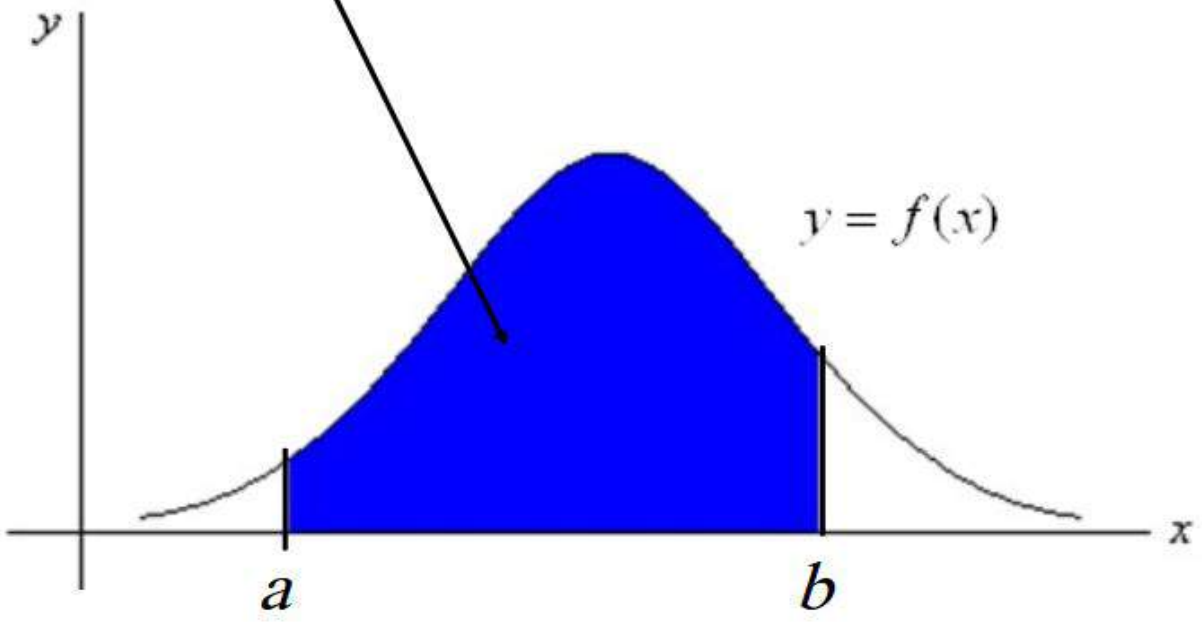


BITS Pilani

Probability Densities

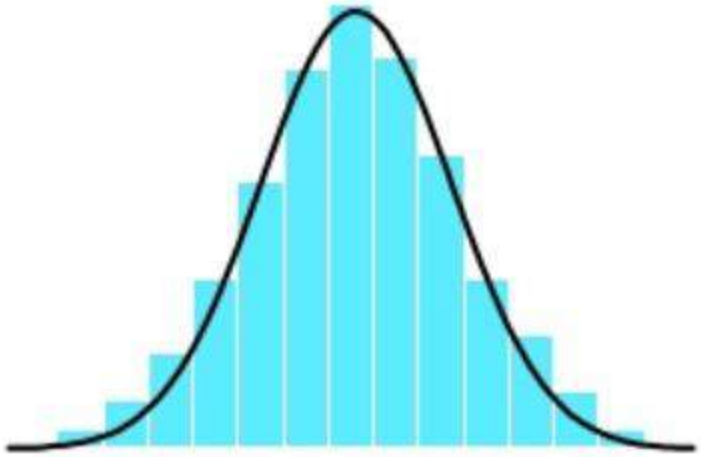
Probability Density Function

$P(a \leq X \leq b)$ is given by the area of the shaded region.



Gaussian Distribution

- The commonest and the most useful continuous distribution.
- A symmetrical probability distribution where most results are located in the middle and few are spread on both sides.
- It has the shape of a bell.
- Can entirely be described by its mean and standard deviation.

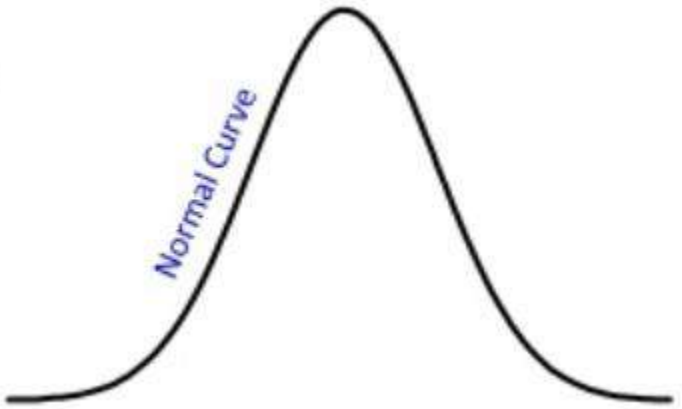


BITS Pilani

Gaussian Distribution

Normal Curve:

- A graphical representation of the normal distribution.
- It is determined by the mean and the standard deviation.
- It a symmetric unimodal bell-shaped curve.
- Its tails extending infinitely in both directions.
- The wider the curve, the larger the standard deviation and the more variation exists in the process.
- The spread of the curve is equivalent to six times the standard deviation of the process.

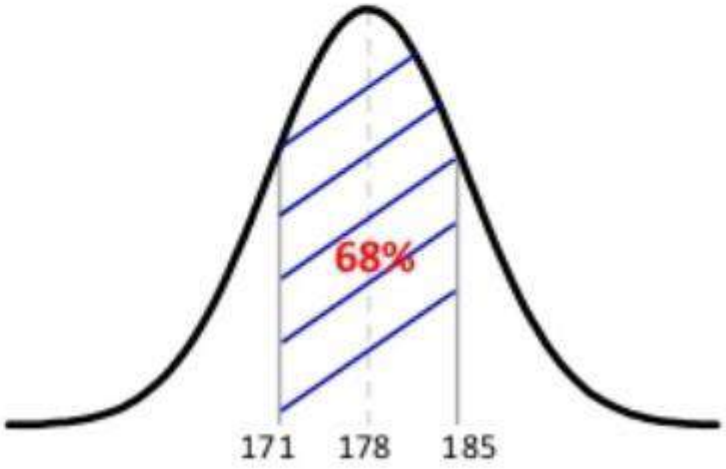


BITS Pilani

Gaussian Distribution

- Suppose that the heights of a sample men are normally distributed.
- The mean height is **178** cm and a standard deviation is **7** cm.

- **We can generalize that:**
  - **68%** of population are between **171** cm and **185** cm.
  - This might be a generalization, but it's true if the data is normally distributed.



BITS Pilani

Mean, Variance & Standard Deviation

- The mean of a discrete random variable is the ***weighted average*** of all of its values. The weights are the probabilities.

- This parameter is also called the expected value of X and is represented by E(X).

$$E(X) = \mu = \sum_{all\ x} xP(x)$$

- The variance is

$$V(X) = \sigma^2 = \sum_{all\ x} (x-\mu)^2 P(x)$$

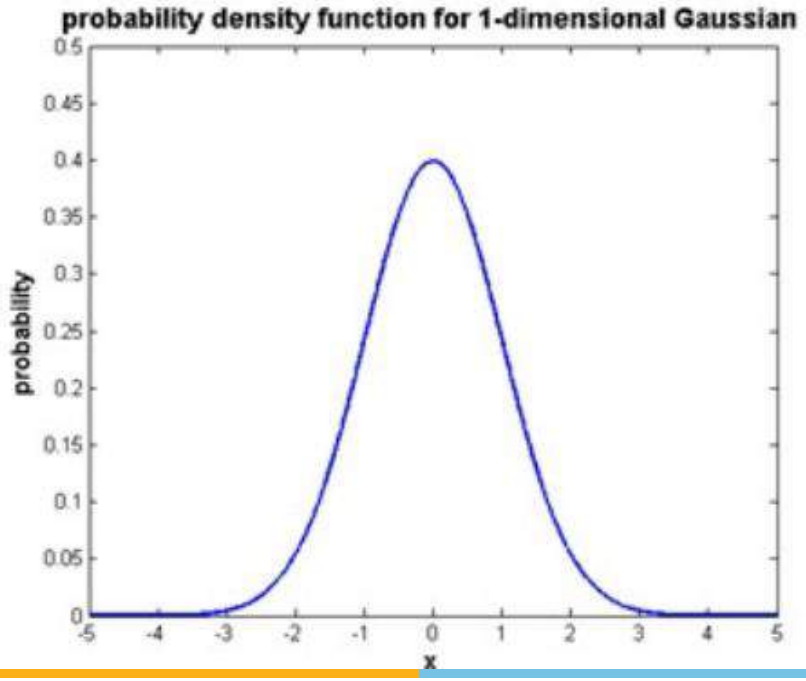
- The standard deviation is

$$\sigma = \sqrt{\sigma^2}$$

Gaussian Distribution

In one dimension

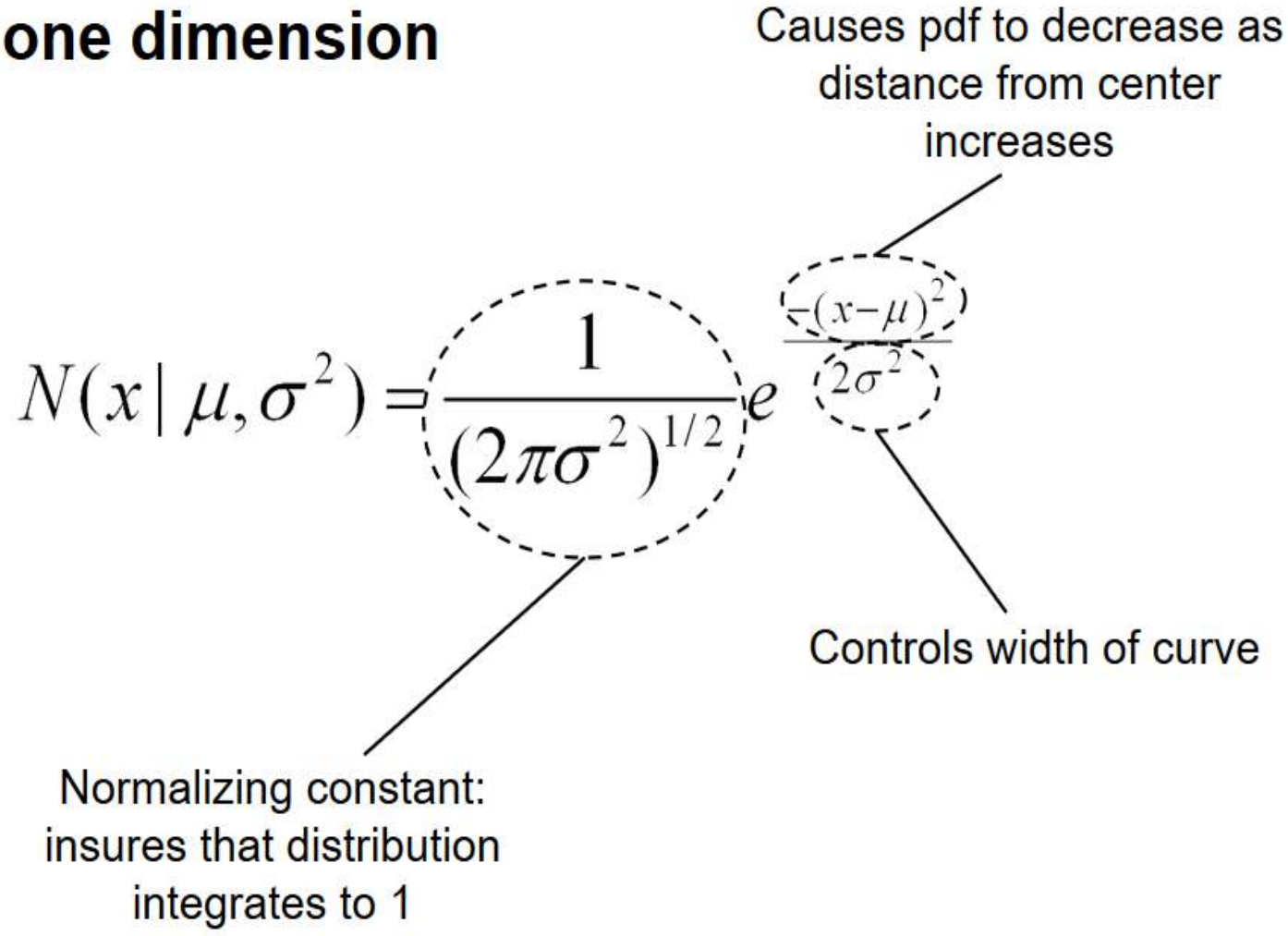
$$N(x \mid \mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$



BITS Pilani

Gaussian Distribution

In one dimension



BITS Pilani

Estimate Probabilities from Data

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(X_i \mid Y_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(X_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (X_i,Y_i) pair

- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Slide adopted from "Introduction to Data mining" Vipin Kumar

JOINT Distributions

- Probability distribution of two random variables X $\{x_1, x_2, \ldots x_n\}$ and Y$\{y_1, y_2 .. y_k\}$
  - Occurrence of X=xi and Y=yi together
- Example:
  - P(X=0, Y<=1)
  - P(X=1)

$$= \sum_{y=0}^{2} P(X = 1, Y)$$

$$= 1/6 + 1/6 + 1/8$$

		Y = 0	Y = 1	Y = 2
X	0	1/4	1/6	1/8
	1	1/6	1/6	1/8

JOINT Distribution

- Marginal Distribution
  - Sum over any one variable is called Marginal Distribution

$$P(X=x) = \sum_{y \in Y} P(X, Y)$$

$$P(Y=y) = \sum_{x \in X} P(X, Y)$$

Conditional Probability

- Estimating probability for two or more related events
  - Measure influence of one variable over another
- Let A and B be two events, $p(B) > 0$

  $$p(A|B) = p(A \cap B) / p(B)$$
- Using random variable notations,
  - $p(a|b)$ denotes the probability of A=a and B=b
    - i.e. $p(A=a \mid B=b)$

Basic Formulas for Probabilities

- *Product Rule*: probability P(A ∧ B) of a conjunction of two events A and B:

$$P(A \wedge B) = P(A \mid B)\, P(B) = P(B \mid A)\, P(A)$$

- *Sum Rule*: probability of a disjunction of two events A and B:

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

- *Theorem of total probability*: if events $A_1, \ldots, A_n$ are mutually exclusive with $\sum_{i=1}^{n} P(A_i) = 1$, then

$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

Example 2

- Does a patient have cancer or not?
  - The patient takes a lab test and the test returns a correct positive result in only 98% of the cases in which the disease is actually present
  - And a correct negative result in only 97% of the cases in which the disease is not present
  - Of the total population, only 0.008% has cancer

Example 2

Given:

$P(cancer) =$ $P(\neg cancer) =$

$P(+|cancer) =$ $P(-|cancer) =$

$P(+|\neg cancer) =$ $P(-|\neg cancer) =$

Using Bayes Theorem:

$P(\text{cancer}|+) = P(+|cancer)\ P(cancer) / P(+)$

$P(\neg cancer\ |+) = P(+|\ \neg cancer)\ P(\neg cancer) / P(+)$

Since denominator is common

$P(\text{cancer}|+)$ *proportional to* $P(+|cancer)\ P(cancer)$

$P(\neg cancer\ |+)$ *proportional to* $P(+|\ \neg cancer)\ P(\neg cancer)$

Example 2

$P(cancer) = 0.008$

$P(+|cancer) = 0.98$

$P(+|\neg cancer) = 1\text{-}0.97=0.3$

$P(\neg cancer) = 1\text{-}0.008=0.992$

$P(-|cancer) = 0.02$

$P(-|\neg cancer) = 0.97$

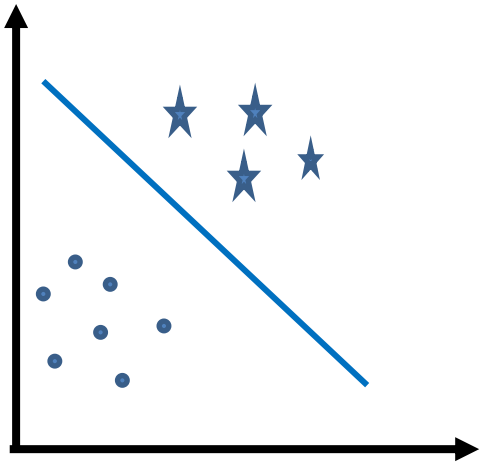
Remember: Some terminology

- Likelihood function: $P(\text{data} \mid \theta)$
- Prior: $P(\theta)$
- Posterior: $P(\theta \mid \text{data})$

- **Conjugate prior**: $P(\theta)$ is the conjugate prior for likelihood function $P(\text{data} \mid \theta)$ if the forms of $P(\theta)$ and $P(\theta \mid \text{data})$ are the same.

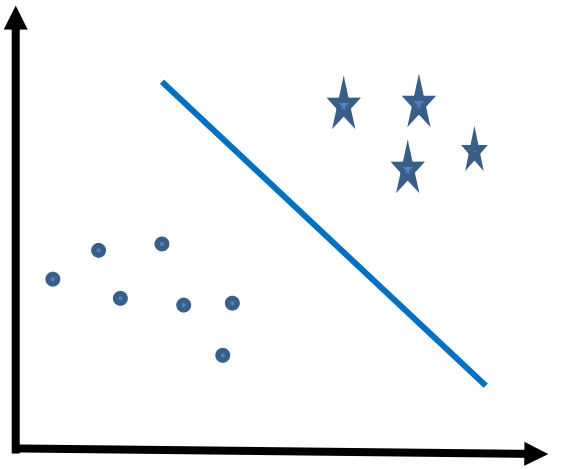
Machine Learning

- Objective of ML: Given data X and target variable Y, determine the joint distribution P(X, Y)
- Classification Problem
  - Decision boundary that separates one class from another class
  - Determine P(Y|X)
  - These models are called Deterministic



Machine Learning

- Alternate approach is to understand the process that generated the data
  - Generative Models P(X,Y)
  - Build a model for all the positive cases or category 1
  - Build another model for all the negative cases or category 2
  - For predicting a new test case
    - Run the test case with both the models and choose the model with maximum probability



Machine Learning

- Generative models
  - Build model to estimate the posterior probability P(Y|X) by estimating
  - likelihood of data given target (hypothesis) P(X|Y)
  - Prior probabilities over target P(Y)
  - In general, for a specific class Y=ck,

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

Hypothesis

- Relationship between the input and output values.
- Lets say that **target function** $y=f(\mathbf{x})$ However, f(.) is unknown function to us.
- Machine learning algorithms try to guess a hypothesis function $h(\mathbf{x})$ that approximates the unknown $f(.)$
- Set of all possible hypotheses is known as the Hypothesis set or space H(.)
- Goal is the learning process is to find the final hypothesis that best approximates the unknown target function.

Bayes Theorem

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

- $P(h)$ = prior probability of hypothesis h
- $P(D)$ = prior probability of training data D
- $P(h \mid D)$ = probability of h given D
- $P(D \mid h)$ = probability of D given h

Choosing Hypotheses

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

- Generally want the most probable hypothesis given the training data

  Maximum a posteriori hypothesis h_{MAP}:

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

- If assume $P(h_i) = P(h_j)$ then can further simplify, and choose the *Maximum likelihood* (ML) hypothesis

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

Brute Force MAP Hypothesis

1. For each hypothesis *h* in *H*, calculate the posterior probability

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

1. Output the hypothesis h_{MAP} with the highest posterior probability

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(h|D)$$

Practical Issues

- Require initial knowledge of many probabilities
  - Often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions.
- Significant computational cost required to determine the Bayes optimal hypothesis in the general case (linear in the number of candidate hypotheses)

MAP Hypothesis

- Using Bayes theorem, we compute the MAP hypothesis for all probable hypothesis (or all unique class labels)
- Identify the best hypothesis describing the data as

$$h_{MAP} = argmax_{h \in H} P(h|D)$$

$$= argmax_{h \in H} \frac{P(D|h)P(h)}{P(D)}$$

$$= argmax_{h \in H} P(D|h)P(h)$$

H: set of all hypothesis

P(D) is independent of h and is same for all hypothesis, therefore dropped

Maximum Likelihood estimation

- When no prior information is available, all hypothesis are equally likely i.e. p(hi) = p(hj)
  - This is also true for a balanced class problem where all the classes are equally likely
  - This is known as Uniform prior
  - MAP hypothesis further simplifies to:

$$H_{ML} = argmax_{h \in H} \mathrm{P}(\mathrm{D} \mid \mathrm{h})$$

This is called Maximum Likelihood Hypothesis

ML setting

- Bayesian Analysis
  - start with some belief about the system, called a prior.
  - Then we obtain some data and use it to update our belief.
  - The outcome is called a posterior.
  - Should we obtain even more data, the old posterior becomes a new prior and the cycle repeats.
  - People often use likelihood for evaluation of models: a model that gives higher likelihood to real data is better

ML Setting

- P(h | D) a posterior determines the class label
- It's a probability distribution over model parameters obtained from prior beliefs and data.
- When one uses likelihood to get point estimates of model parameters, it's called Maximum Likelihood estimation or MLE.
- If one also takes the prior into account, then it's maximum a posteriori estimation (MAP).
- MLE and MAP are the same if the prior is uniform
- This forms the basis for Naïve Bayes classifier

Relation to Concept Learning

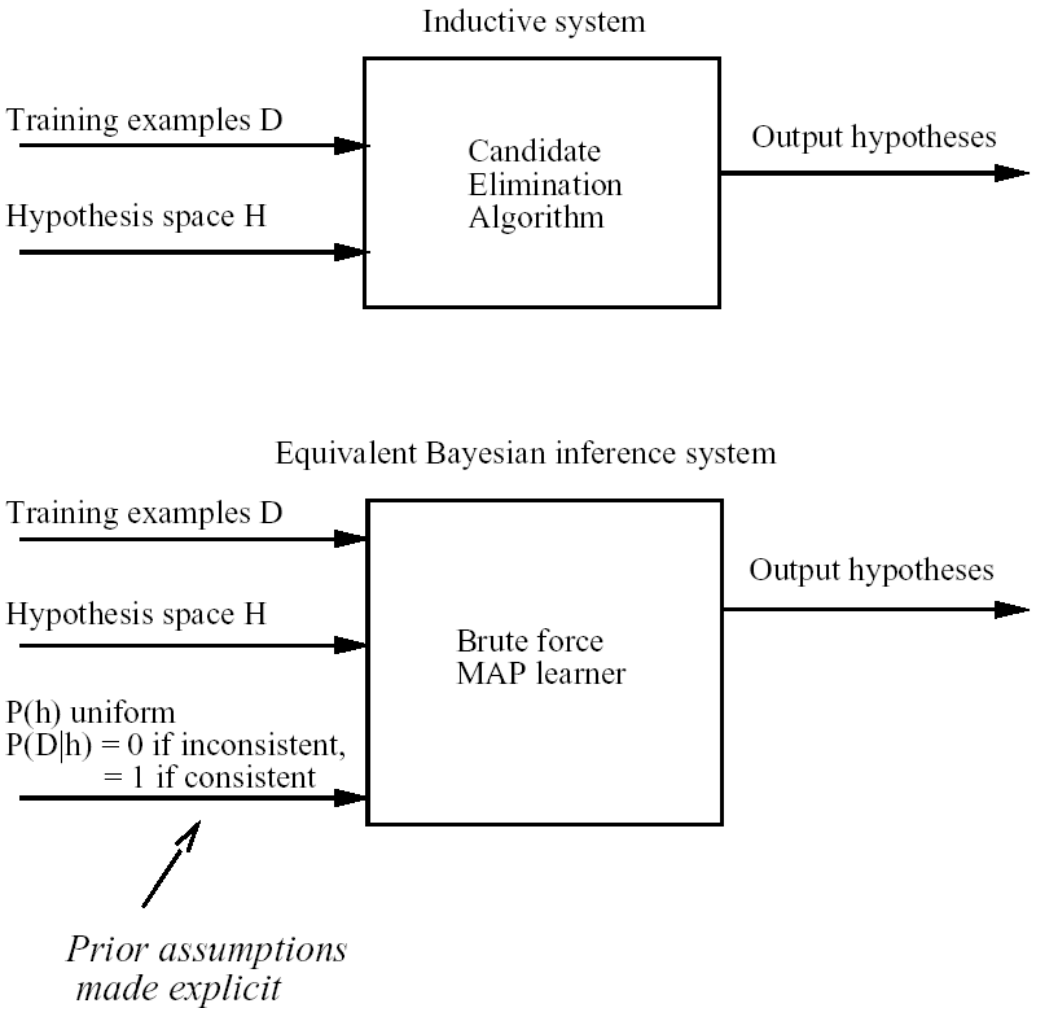
- Consider our usual concept learning task
  - instance space *X*, hypothesis space *H*, training examples *D*
  - consider the FindS learning algorithm (outputs most specific hypothesis from the version space $VS_{H,D}$)
- What would Bayes rule produce as the MAP hypothesis?
- Does *FindS* output a MAP hypothesis??

Relation to Concept Learning

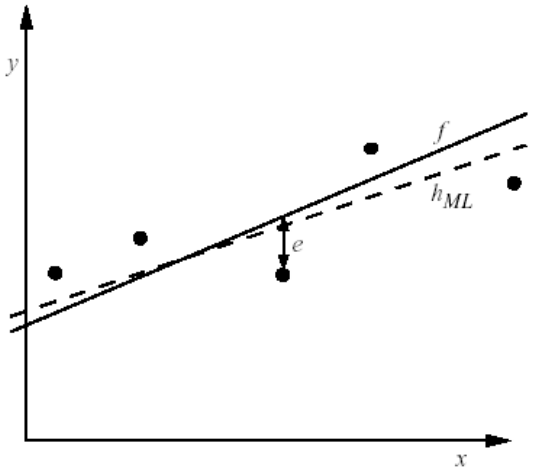
- Assume fixed set of instances $\langle x_1, \ldots, x_m \rangle$
- Assume D is the set of classifications: $D = \langle c(x_1), \ldots, c(x_m) \rangle$
- Choose $P(D|h)$:
  - $P(D|h) = 1$ if h consistent with D
  - $P(D|h) = 0$ otherwise
- Choose $P(h)$ to be *uniform* distribution
  - $P(h) = 1/|H|$ for all h in H
- Then,

$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$$

Characterizing Learning Algorithms by Equivalent MAP Learners



Learning A Real Valued Function



Consider any real-valued target function *f*

Training examples <x_i, d_i>, where d_i is noisy training value

- $d_i = f(x_i) + e_i$
- e_i is random variable (noise) drawn independently for each x_i according to some Gaussian distribution with mean=0

Then the maximum likelihood hypothesis h_{ML} is the one that minimizes the sum of squared errors: $h_{ML} = \arg\min_{h \in H} \sum_{i=1}^{m} (d_i - h(x_i))^2$

Learning A Real Valued Function

$$
\begin{aligned}
h_{ML} &= \underset{h \in H}{\operatorname{argmax}} p(D|h) \\
&= \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} p(d_i|h) \\
&= \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2}
\end{aligned}
$$

- Maximize natural log of this instead...

$$
\begin{aligned}
h_{ML} &= \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2 \\
&= \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} -\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2 \\
&= \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} -\left(d_i - h(x_i)\right)^2 \\
&= \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} \left(d_i - h(x_i)\right)^2
\end{aligned}
$$

Learning to Predict Probabilities

- Consider predicting survival probability from patient data
- Training examples $\langle x_i, d_i \rangle$, where d_i is 1 or 0
- Want to train neural network to output a *probability* given x_i (not a 0 or 1)
- In this case can show

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i))$$

- Weight update rule for a sigmoid unit:

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

where

$$\Delta w_{jk} = \eta \sum_{i=1}^{m} (d_i - h(x_i))\ x_{ijk}$$

Minimum Description Length Principle

Occam's razor: prefer the shortest hypothesis

MDL: prefer the hypothesis h that minimizes

$$h_{MDL} = \underset{h \in H}{\operatorname{argmin}} L_{C_1}(h) + L_{C_2}(D|h)$$

where $L_C(x)$ is the description length of x under encoding C

Example: H = decision trees, D = training data labels

- $L_{C1}(h)$ is # bits to describe tree h
- $L_{C2}(D|h)$ is # bits to describe D given h
  - Note $L_{C2}(D|h) = 0$ if examples classified perfectly by h. Need only describe exceptions
- Hence h_{MDL} trades off tree size for training errors

Minimum Description Length Principle

$$
\begin{aligned}
h_{MAP} &= \arg\max_{h \in H} P(D|h)P(h) \\
&= \arg\max_{h \in H} \log_2 P(D|h) + \log_2 P(h) \\
&= \arg\min_{h \in H} -\log_2 P(D|h) - \log_2 P(h) \quad (1)
\end{aligned}
$$

Interesting fact from information theory:

The optimal (shortest expected coding length) code for an event with

probability p is $-\log_2 p$ bits.

So interpret (1):

- $-\log_2 P(h)$ is length of h under optimal code
- $-\log_2 P(D|h)$ is length of D given h under optimal code

→ prefer the hypothesis that minimizes

length(*h*) + *length*(*misclassifications*)

Most Probable Classification of New Instances

- So far we've sought the most probable *hypothesis* given the data D (i.e., h_{MAP})
- Given new instance x, what is its most probable *classification*?
  - $h_{MAP}(x)$ is not the most probable classification!
- Consider:
  - Three possible hypotheses:

$$P(h_1|D) = .4,\ P(h_2|D) = .3,\ \ P(h_3|D) = .3$$

  - Given new instance x,

$$h_1(x) = +,\ \ h_2(x) = -,\ \ h_3(x) = -$$

  - What's most probable classification of x?

Bayes Optimal Classifier

- **Bayes optimal classification:**

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i) P(h_i|D)$$

- Example:

$P(h_1|D) = .4,\ P(-|h_1) = 0,\ P(+|h_1) = 1$
$P(h_2|D) = .3,\ P(-|h_2) = 1,\ P(+|h_2) = 0$
$P(h_3|D) = .3,\ P(-|h_3) = 1,\ P(+|h_3) = 0$

therefore

$$\sum_{h_i \in H} P(+|h_i) P(h_i|D) = .4$$
$$\sum_{h_i \in H} P(-|h_i) P(h_i|D) = .6$$

and

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i) P(h_i|D) = -$$

Gibbs Classifier

- Bayes optimal classifier provides best result, but can be expensive if many hypotheses.
- Gibbs algorithm:
  1. Choose one hypothesis at random, according to $P(h|D)$
  2. Use this to classify new instance
- Surprising fact: Assume target concepts are drawn at random from H according to priors on H. Then:

$$E[error_{Gibbs}] \leq 2E[error_{BayesOptional}]$$

- Suppose correct, uniform prior distribution over H, then
  - Pick any hypothesis from VS, with uniform probability
  - Its expected error no worse than twice Bayes optimal

Thank You

BITS Pilani

innovate achieve lead

Machine Learning
DSECL ZG565

Dr. Chetana Gavankar, Ph.D,
IIT Bombay-Monash University Australia
Chetana.gavankar@pilani.bits-pilani.ac.in

BITS Pilani
Pilani Campus

Lecture No. – 3 | Bayesian Learning

Date – 09/11/2019

Time – 9:00 AM – 11:00 AM

Session Content

- Probabilistic Generative Classifiers
- Bayes optimal classifier (T1 book by Tom Mitchell - 6.7)
- Gibbs Algorithm (T1 book by Tom Mitchell - 6.8)
- Naïve Bayes Classifier (T1 book by Tom Mitchell - 6.9)
- Text classification model (T1 book by Tom Mitchell - 6.9)
- Image classification

Bayesian Learning

- Bayesian learning algorithms that calculate explicit probabilities for hypotheses, such as the naive Bayes classifier, are among the most practical approaches to certain types of learning problems

- For example: Problem of learning to classify text documents such as electronic news articles.
- For such learning tasks, the naive Bayes classifier is among the most effective algorithms known

Features of Bayesian learning

- Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct.
- Flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example.
- Bayesian methods can accommodate hypotheses that make probabilistic predictions (e.g., hypotheses such as "this pneumonia patient has a 93% chance of complete recovery").

Features of Bayesian learning

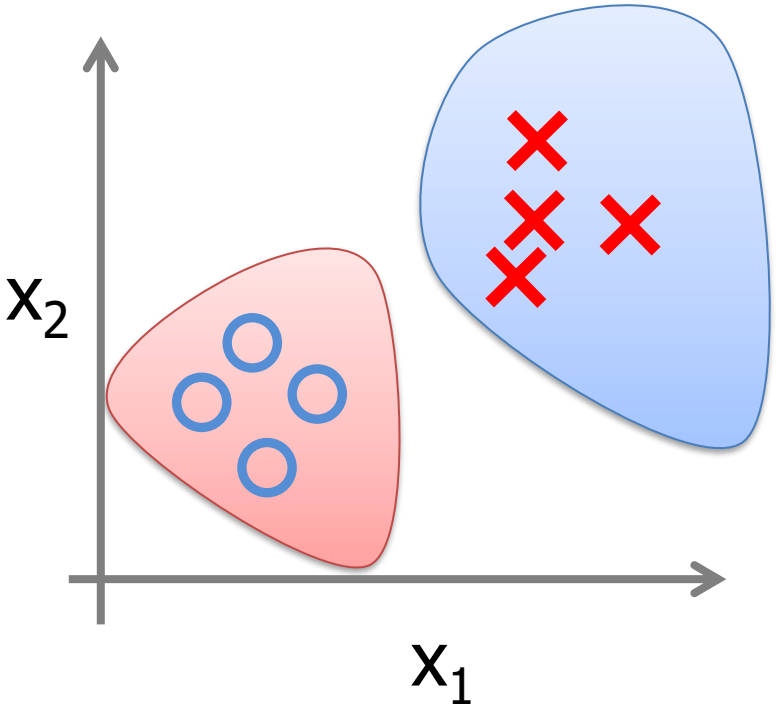
- Prior knowledge can be combined with observed data to determine the final probability of a hypothesis.
- Prior knowledge is provided by asserting
  - prior probability for each candidate hypothesis, and
  - probability distribution over observed data for each possible hypothesis.
- New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.

Probabilistic Generative Classifiers

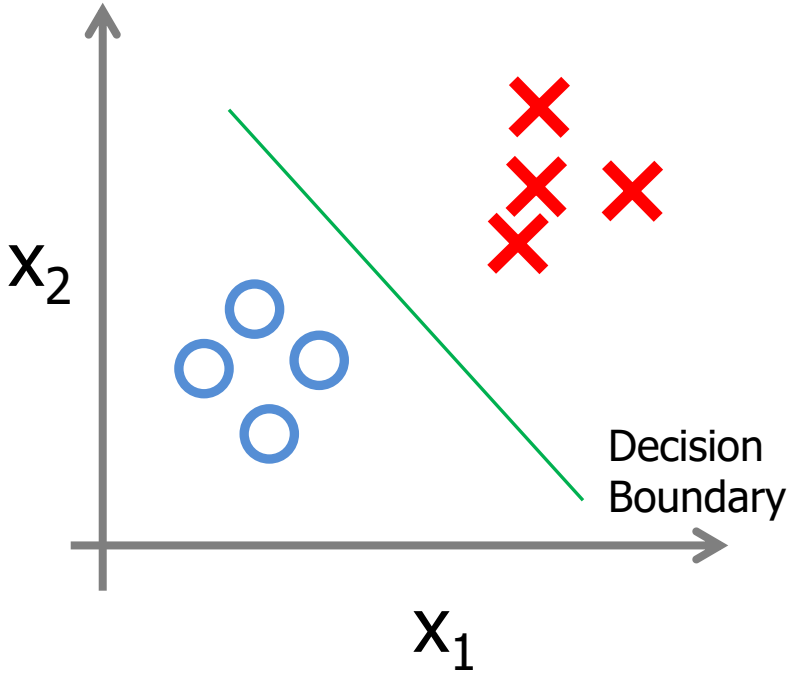
- Approach is to understand the process that generated the data
  - Generative Models P(X,Y)
  - Build a model for all the positive cases or category 1
  - Build another model for all the negative cases or category 2
  - For predicting a new test case
    - Run the test case with both the models and choose the model with maximum probability

Probabilistic Generative Model versus Probabilistic Discriminative Model

Generative:



Discriminative:



Probabilistic Models: Generative/Discriminative

- Model $p(C_k|\boldsymbol{x})$ in an *inference* stage and use it to make optimal decisions
- Approaches to computing the $p(C_k|\boldsymbol{x})$
  1. Generative
     - Model class conditional densities by $p(\boldsymbol{x}|C_k)$ together with prior probabilities $p(C_k)$
     - Then use Bayes rule to compute posterior

$$p(C_k \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C_k)p(C_k)}{p(\boldsymbol{x})}$$

  2. Discriminative
     - Directly model conditional probabilities $p(C_k|\boldsymbol{x})$

13

Probabilistic Generative Model versus Probabilistic Discriminative Model

Generative	Discriminative
Ex: Naïve Bayes	Ex: Logistic Regression
Estimate $P(Y)$ and $P(X\|Y)$	Finds class label directly $P(Y\|X)$
Prediction $\hat{y} = \text{argmax}_y \, P(Y = y)P(X = x\|Y = y)$	Prediction $\hat{y} = P(Y = y\|X = x)$

Generative Models

- Generative models
  - Build model to estimate the posterior probability P(Y|X) by estimating
  - likelihood of data given target (hypothesis) P(X|Y)
  - Prior probabilities over target P(Y)
  - In general, for a specific class Y=ck,

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

Most Probable Classification of New Instances

- So far we've sought the most probable *hypothesis* given the data D (i.e., h_{MAP})
- Given new instance x, what is its most probable *classification*?
  - $h_{MAP}(x)$ is not the most probable classification!
- Consider:
  - Three possible hypotheses:

$$P(h_1|D) = .4,\ P(h_2|D) = .3,\ \ P(h_3|D) = .3$$

  - Given new instance x,

$$h_1(x) = +,\ \ h_2(x) = -,\ \ h_3(x) = -$$

  - What's most probable classification of x?

Bayes Optimal Classifier

- **Bayes optimal classification:**

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

- Example:

$P(h_1|D) = .4,\ P(-|h_1) = 0,\ P(+|h_1) = 1$
$P(h_2|D) = .3,\ P(-|h_2) = 1,\ P(+|h_2) = 0$
$P(h_3|D) = .3,\ P(-|h_3) = 1,\ P(+|h_3) = 0$

therefore

$$\sum_{h_i \in H} P(+|h_i)P(h_i|D) = .4$$
$$\sum_{h_i \in H} P(-|h_i)P(h_i|D) = .6$$

and

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) = -$$

Gibbs Classifier

- Bayes optimal classifier provides best result, but can be expensive if many hypotheses.
- Gibbs algorithm:
  1. Choose one hypothesis at random, according to $P(h|D)$
  2. Use this to classify new instance
- Surprising fact: Assume target concepts are drawn at random from H according to priors on H. Then:

$$E[error_{Gibbs}] \leq 2E[error_{BayesOptional}]$$

- Suppose correct, uniform prior distribution over H, then
  - Pick any hypothesis from VS, with uniform probability
  - Its expected error no worse than twice Bayes optimal

Conditional independence

- **Definition**: X is conditionally independent of Y given Z, if the probability distribution governing X is independent of the value of Y, given the value of Z

$$(\forall i, j, k)\ P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z_k)$$

$$P(X|Y, Z) = P(X|Z)$$

Example:

$$P(\text{Thunder}|\text{Rain}, \text{Lightning}) = P(\text{Thunder}|\text{Lightning})$$

Slide credit: Tom Mitchell

Applying conditional independence

- Naïve Bayes assumes X_i are conditionally independent given Y

e.g., $P(X_1|X_2, Y) = P(X_1|Y)$

$$P(X_1, X_2|Y) = P(X_1|X_2, Y)P(X_2|Y)$$
$$= P(X_1|Y)P(X_2|Y)$$

General form: $P(X_1, \cdots, X_n|Y) = \prod_{j=1}^{n} P(X_j|Y)$

How many parameters to describe $P(X_1, \cdots, X_n|Y)$? $P(\mathrm{Y})$?

- Without conditional independence assumption?
- With conditional independence assumption?

Slide credit: Tom Mitchell

Naïve Bayes Independence assumption

- Assumption:

$$P(X_1, \cdots, X_n|Y) = \prod_{j=1}^{n} P(X_j|Y)$$

- i.e., X_i and X_j are conditionally independent given Y for $i \neq j$

Slide credit: Tom Mitchell

Naïve Bayes classifier

- Bayes rule:

$$P(Y = y_k | X_1, \cdots, X_n) = \frac{P(Y = y_k) P(X_1, \cdots, X_n | Y = y_k)}{\sum_j P(Y = y_j) P(X_1, \cdots, X_n | Y = y_j)}$$

- Assume conditional independence among X_i's:

$$P(Y = y_k | X_1, \cdots, X_n) = \frac{P(Y = y_k) \Pi_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j) \Pi_i P(X_i | Y = y_j)}$$

- Pick the most probable (MAP) Y

$$\hat{Y} \leftarrow \underset{y_k}{\operatorname{argmax}} P(Y = y_k) \Pi_i P(X_i | Y = y_k)$$

Prior Probability

MLE

Slide credit: Tom Mitchell

NAÏVE BAYES CLASSIFIER

- Assume independence among attributes X_i when class is given:
  - $P(X_1, X_2, \ldots, X_d \mid Y_j) = P(X_1 \mid Y_j)\, P(X_2 \mid Y_j) \ldots P(X_d \mid Y_j)$
  - Now we can estimate $P(X_i \mid Y_j)$ for all X_i and Y_j combinations from the training data
  - New point is classified to Y_j if $P(Y_j) \prod P(X_i \mid Y_j)$ is maximal.

Example 1

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

A: attributes

M: mammals

N: non-mammals

$$P(A \mid M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A \mid N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A \mid M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A \mid N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

P(A|M)P(M) > P(A|N)P(N)

=> Mammals

Naive Bayes Classifier

- Assume target function $f : X \rightarrow V$, where each instance x described by attributes $<a_1, a_2 \ldots a_n>$.
- Most probable value of $f(x)$ is:

$$v_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j | a_1, a_2 \ldots a_n)$$

$$v_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} \frac{P(a_1, a_2 \ldots a_n | v_j) P(v_j)}{P(a_1, a_2 \ldots a_n)}$$

$$= \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2 \ldots a_n | v_j) P(v_j)$$

Naive Bayes assumption:

$$P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$$

Naive Bayes classifier:

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j)$$

Naive Bayes Algorithm

- Naive Bayes Learn(*examples*)

  For each target value v_j

  $$\hat{P}(v_j) \leftarrow \text{estimate } P(v_j)$$

  For each attribute value a_i of each attribute a

  $$\hat{P}(a_i \mid v_j) \leftarrow \text{estimate } P(a_i \mid v_j)$$

- Classify New Instance(*x*)

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i | v_j)$$

Naive Bayes: Example

- Consider *PlayTennis*, and new instance

  <*Outlk = sun, Temp = cool, Humid = high, Wind = strong*>

- Want to compute:

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j)$$

$P(y)$ P($sun|y$) $P(cool|y)$ $P(high|y)$ P($strong|y$) = .005

$P(n)$ P($sun|n$) $P(cool|n)$ $P(high|n)$ P($strong|n$) = .021

$\rightarrow v_{NB} = n$

Issues with Naïve Bayes Classifier

Naïve Bayes Classifier:

P(Refund = Yes | No) = 3/7
P(Refund = No | No) = 4/7
P(Refund = Yes | Yes) = 0
P(Refund = No | Yes) = 1
P(Marital Status = Single | No) = 2/7
P(Marital Status = Divorced | No) = 1/7
P(Marital Status = Married | No) = 4/7
P(Marital Status = Single | Yes) = 2/3
P(Marital Status = Divorced | Yes) = 1/3
P(Marital Status = Married | Yes) = 0

For Taxable Income:
If class = No: sample mean = 110
sample variance = 2975
If class = Yes: sample mean = 90
sample variance = 25

- P(Yes) = 3/10
  P(No) = 7/10

- P(Yes | Married) = 0 x 3/10 / P(Married)
  P(No | Married) = 4/7 x 7/10 / P(Married)

Slide adopted from "Introduction to Data mining" Vipin Kumar

Issues with Naïve Bayes Classifier

Consider the table with Tid = 7 deleted

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Naïve Bayes Classifier:

P(Refund = Yes | No) = 2/6
P(Refund = No | No) = 4/6
P(Refund = Yes | Yes) = 0
P(Refund = No | Yes) = 1
P(Marital Status = Single | No) = 2/6
P(Marital Status = Divorced | No) = 0
P(Marital Status = Married | No) = 4/6
P(Marital Status = Single | Yes) = 2/3
P(Marital Status = Divorced | Yes) = 1/3
P(Marital Status = Married | Yes) = 0/3
For Taxable Income:
If class = No: sample mean = 91
sample variance = 685
If class = No: sample mean = 90
sample variance = 25

Given X = (Refund = Yes, Divorced, 120K)

$P(X \mid No) = 2/6 \times 0 \times 0.0083 = 0$

$P(X \mid Yes) = 0 \times 1/3 \times 1.2 \times 10^{-9} = 0$

Naïve Bayes will not be able to classify X as Yes or No!

Issues with Naïve Bayes Classifier

- If one of the conditional probabilities is zero, then the entire expression becomes zero
- Need to use other estimates of conditional probabilities than simple fractions
- Probability estimation:

$$\text{Original} : P(A_i \mid C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace} : P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c}$$

$$\text{m-estimate} : P(A_i \mid C) = \frac{N_{ic} + mp}{N_c + m}$$

c: number of classes

p: prior probability of the class

m: parameter

N_c: number of instances in the class

N_{ic}: number of instances having attribute value A_i in class c

A Simple Example

Text	Tag
"A great game"	Sports
"The election was over"	Not sports
"Very clean match"	Sports
"A clean but forgettable game"	Sports
"It was a close election"	Not sports

Which tag does the sentence *A very close game* belong to? i.e. P(sports| *A very close game*)

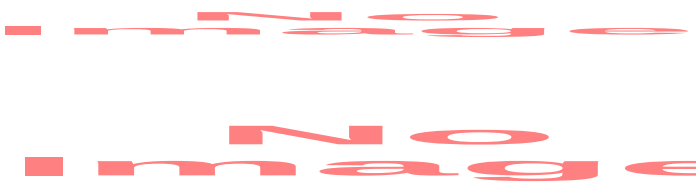
Feature Engineering: Bag of words i.e use word frequencies without considering order

Using Bayes Theorem:

P(sports| *A very close game*)

$$= \frac{P(\textit{A very close game}|\ \text{sports})\ P(\text{sports})}{P(\textit{A very close game})}$$

We assume that every word in a sentence is **independent** of the other ones



"close" doesn't appear in sentences of sports tag, So P(close | sports) = 0, which makes product 0

Laplace smoothing

- Laplace smoothing: we add 1 or in general constant k to every count so it's never zero.
- To balance this, we add the number of possible words to the divisor, so the division will never be greater than 1
- In our case, the possible words are ['a', 'great', 'very', 'over', 'it', 'but', 'game', 'election', 'clean', 'close', 'the', 'was', 'forgettable', 'match'].

Apply Laplace Smoothing

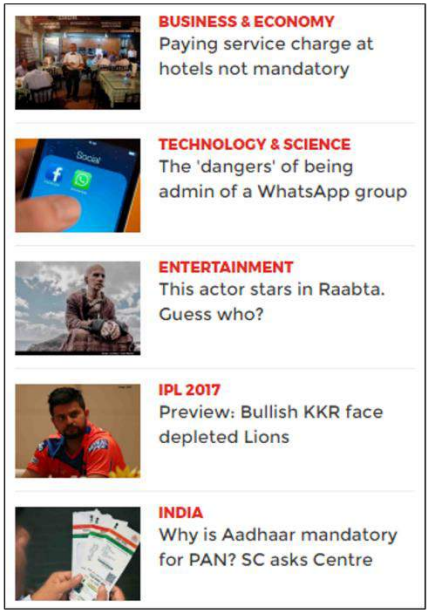
Word	P(word | Sports)	P(word | Not Sports)
a	2+1 / 11+14	1+1 / 9+14
very	1+1 / 11+14	0+1 / 9+14
close	0+1 / 11+14	1+1 / 9+14
game	2+1 / 11+14	0+1 / 9+14

$$P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports) \times P(Sports)$$
$$= 2.76 \times 10^{-5}$$
$$= 0.0000276$$

$$P(a|Not\,Sports) \times P(very|Not\,Sports) \times P(close|Not\,Sports) \times P(game|Not\,Sports) \times P(Not\,Sports)$$
$$= 0.572 \times 10^{-5}$$
$$= 0.00000572$$

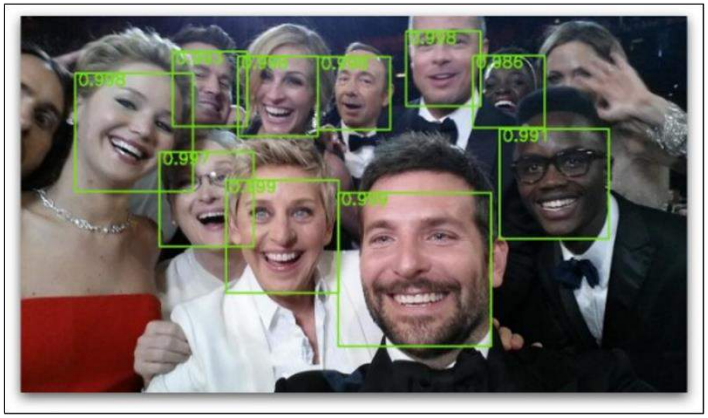
Naïve Bayes Classifier Applications

Categorizing News

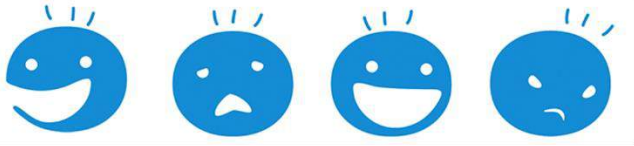


Email Spam Detection

Face Recognition



Sentiment Analysis



Naive Bayes Classifier

- Along with decision trees, neural networks, one of the most practical learning methods.
- When to use
  - Moderate or large training set available
  - Attributes that describe instances are conditionally independent given classification
- Successful applications:
  - Diagnosis
  - Classifying text documents

Learning to Classify Text

- Why?
  - Learn which news articles are of interest
  - Learn to classify web pages by topic

- Naive Bayes is among most effective algorithms
- What attributes shall we use to represent text documents??

Learning to Classify Text (2/4)

Target concept Interesting? : *Document* $\rightarrow\{+, -\}$

1. Represent each document by vector of words
   - one attribute per word position in document
2. Learning: Use training examples to estimate
   - $P(+)$
   - $P(-)$
   - $P(doc|+)$
   - $P(doc|-)$

Naive Bayes conditional independence assumption

$$P(doc|v_j) = \prod_{i=1}^{length(doc)} P(a_i = w_k|v_j)$$

where $P(a_i = w_k \mid v_j)$ is probability that word in position i is w_k, given v_j

one more assumption: $P(a_i = w_k|v_j) = P(a_m = w_k|v_j), \forall i, m$

Learning to Classify Text (3/4)

LEARN_NAIVE_BAYES_TEXT (*Examples*, V)

1. *collect all words and other tokens that occur in Examples*

- *Vocabulary* ← all distinct words and other tokens in *Examples*

2. *calculate the required* $P(v_j)$ *and* $P(w_k \mid v_j)$ *probability terms*

- For each target value v_j in *V* do
  - $docs_j$ ← subset of *Examples* for which the target value is v_j
  - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
  - $Text_j$ ← a single document created by concatenating all members of $docs_j$

Learning to Classify Text (4/4)

- $n \leftarrow$ total number of words in $Text_j$ (counting duplicate words multiple times)
- for each word w_k in *Vocabulary*
  - $n_k \leftarrow$ number of times word w_k occurs in $Text_j$
  - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

CLASSIFY_NAIVE_BAYES_TEXT (*Doc*)

- *positions* $\leftarrow$ all word positions in *Doc* that contain tokens found in *Vocabulary*
- Return v_{NB} where $v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_{i \in positions} P(a_i|v_j)$

Baseline: Bag of Words Approach

aardvark	0
about	2
all	2
Africa	1
apple	0
anxious	0
...	
gas	1
...	
oil	1
…	
Zaire	0

Case Study: Text Classification

- Classify e-mails
  - Y = {Spam, NotSpam}
- Classify news articles
  - Y = what is the topic of the article?
- Classify webpages
  - Y = {student, professor, project, …}
- What about the features X?
  - The text!

Features X are entire document - X_i for ith word in article

Article from rec.sport.hockey

```
Path: cantaloupe.srv.cs.cmu.edu!das-news.harvard.e
From: xxx@yyy.zzz.edu (John Doe)
Subject: Re: This year's biggest and worst (opinic
Date: 5 Apr 93 09:53:39 GMT

I can only comment on the Kings, but the most
obvious candidate for pleasant surprise is Alex
Zhitnik. He came highly touted as a defensive
defenseman, but he's clearly much more than that.
Great skater and hard shot (though wish he were
more accurate). In fact, he pretty much allowed
the Kings to trade away that huge defensive
liability Paul Coffey. Kelly Hrudey is only the
biggest disappointment if you thought he was any
good to begin with. But, at best, he's only a
mediocre goaltender. A better choice would be
Tomas Sandstrom, though not through any fault of
his own, but because some thugs in Toronto decided
```

Naïve Bayes for Text Classification

- **Naïve Bayes assumption helps a lot!**
  - $P(X_i = x_i|Y = y)$ is just the probability of observing word x_i at the ith position in a document on topic y.
  - Assume X_i is independent of all other words in document given the label y: $P(X_i = x_i|Y = y, X_{-i}) = P(X_i = x_i|Y = y)$.

$$h_{NB}(x) = \arg\max_y P(y) \prod_{i=1}^{lengthDoc} P(X_i = x_i|y)$$

  - For each label y, have 1000 distributions of size 10000 to estimate.
  - This is 10000×1000 items, which is big but much less than 10000^{1000}...

Bag of Words Model

- Typical additional assumption – **Position in document doesn't matter**:

$$P(X_i = x_i \mid Y = y) = P(X_k = x_i \mid Y = y)$$

the probability distributions of words are the same at each position: $P_i = P_j$ for all i, j.

- **"Bag of Words"** model – order of words in the document is ignored
- Now, only 10000 quantities $P(x_i|y)$ to estimate for each label y (the 10000 possible values that x_i can be) plus the prior.

$$h_{NB}(x) = \arg\max_y P(y) \prod_{i=1}^{1000} P(x_i|y)$$

Bag of Words model

- Typical additional assumption – **Position in document doesn't matter**:

$$P(X_i = x_i \mid Y = y) = P(X_k = x_i \mid Y = y)$$

- **"Bag of Words"** model – order of words on the page ignored

Can simplify further:

$$\prod_{i=1}^{\text{lengthDoc}} P(x_i|y) = \prod_{w=1}^{W} P(w|y)^{\text{count}(w)}$$

Bag of Words representation

- Since we are assuming the order of words doesn't matter, an alternative representation of document is as vector of counts:
  - $x^{(j)}$ = number of occurrences of word j in document x.
  - Typical document: [0 0 1 0 0 3 0 0 0 1 0 0 0 1 0 0 2 0 0 ...]
  - Called "bag of words" or "term vector" or "vector space model" representation

Naïve Bayes with Bag of Words for text classification

- Learning phase
  - Class Prior $P(Y)$
  - $P(X_i|Y)$
- Test phase:
  - For each document
    - Use naïve Bayes decision rule

$$h_{NB}(x) = \arg\max_{y} P(y) \prod_{i=1}^{1000} P(x_i|y)$$

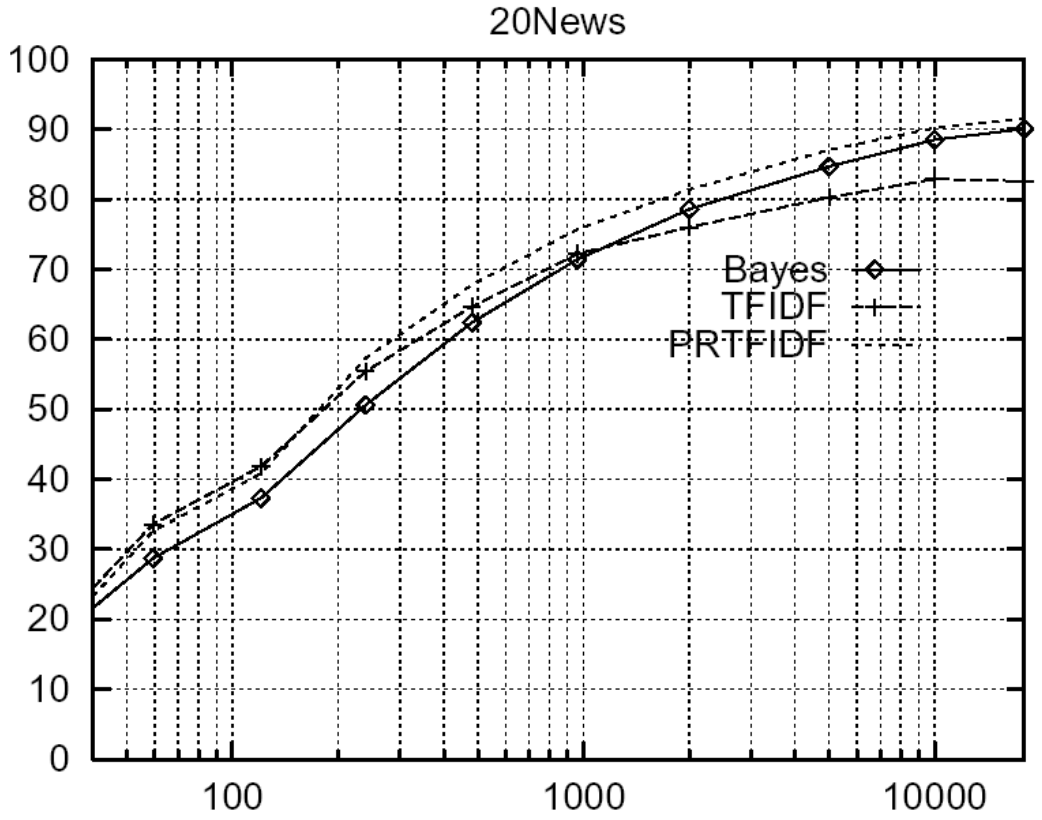
Twenty NewsGroups

- Given 1000 training documents from each group Learn to classify new documents according to which newsgroup it came from

comp.graphics	misc.forsale	alt.atheism	sci.space
comp.os.ms-windows.misc	rec.autos	soc.religion.christian	sci.crypt
comp.sys.ibm.pc.hardware	rec.motorcycles	talk.religion.misc	sci.electronics
comp.sys.mac.hardware	rec.sport.baseball	talk.politics.mideast	sci.med
comp.windows.x	rec.sport.hockey	talk.politics.misc	
		talk.politics.guns	

- Naive Bayes: 89% classification accuracy

Learning Curve for 20 Newsgroups



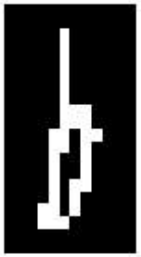
- Accuracy vs. Training set size (1/3 withheld for test)

Estimating Parameters: X_i Continuous

innovate achieve lead

What if features are continuous?

- E.g., character recognition: X_i is intensity at ith pixel



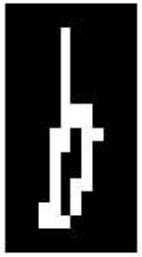
- Gaussian Naïve Bayes (GNB):

$$P(X_i = x|Y = y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} e^{-\frac{(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

distribution of feature X_i is Gaussian with a mean and variance that can depend on the label y_k and which feature X_i it is

What if features are continuous?

- E.g., character recognition: X_i is intensity at ith pixel



- Gaussian Naïve Bayes (GNB):

$$P(X_i = x|Y = y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} e^{-\frac{(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

- Different mean and variance for each class k and each pixel i.
- Sometimes assume variance:
  - Is independent of Y (i.e., just have σ_i)
  - Or independent of X (i.e., just have σ_k)
  - Or both (i.e., just have σ)

Estimating parameters: Y discrete, X_i continuous

- Maximum likelihood estimates:

$$\hat{\mu}_{MLE} = \frac{1}{N}\sum_{j=1}^{N} x_j$$

$$\hat{\mu}_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j X_i^j \delta(Y^j = y_k)$$

kth class

jth training image

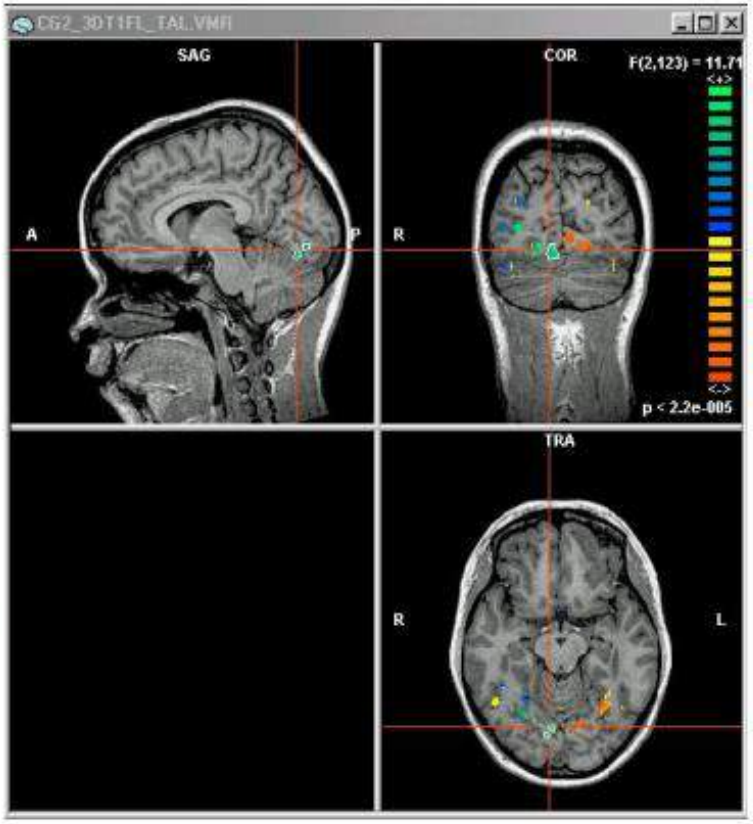
ith pixel in jth training image

$$\hat{\sigma}^2_{unbiased} = \frac{1}{N-1}\sum_{j=1}^{N} (x_j - \hat{\mu})^2$$

$$\hat{\sigma}^2_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k) - 1} \sum_j (X_i^j - \hat{\mu}_{ik})^2 \delta(Y^j = y_k)$$

Example: GNB for classifying mental states

[Mitchell et al.]



- Classify a person's cognitive state, based on brain image
  - reading a sentence or viewing a picture?
  - reading the word describing a "Tool" or "Building"?
  - reading the word describing a "Person" or an "Animal"?
- Training: Patients were shown words of different categories and then a measurement was done to see what parts of the brain responded.

Example: GNB for classifying mental states

[Mitchell et al.]

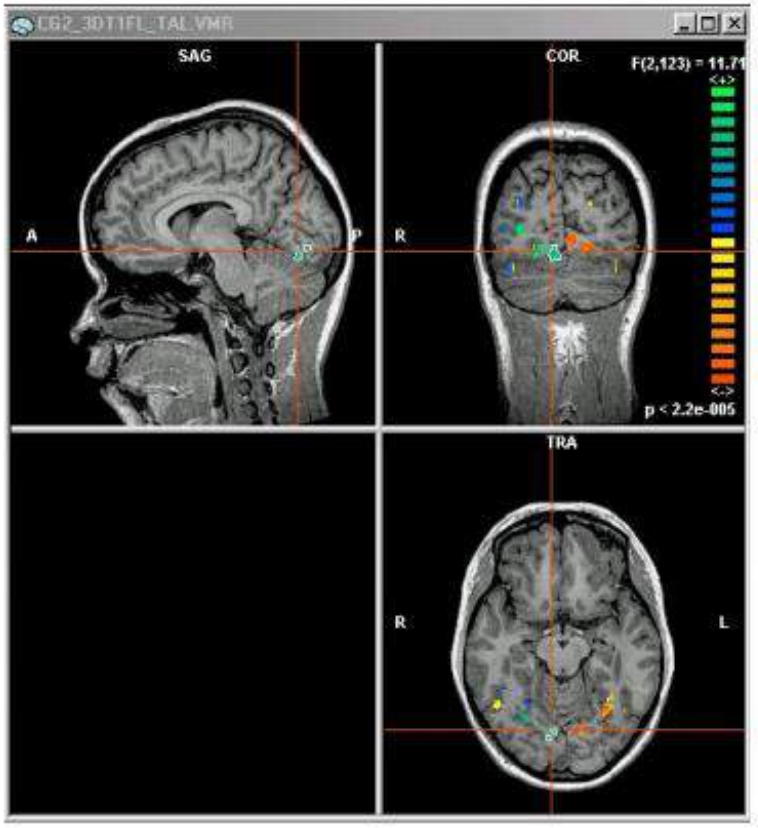
~1mm resolution

~2 images per sec.

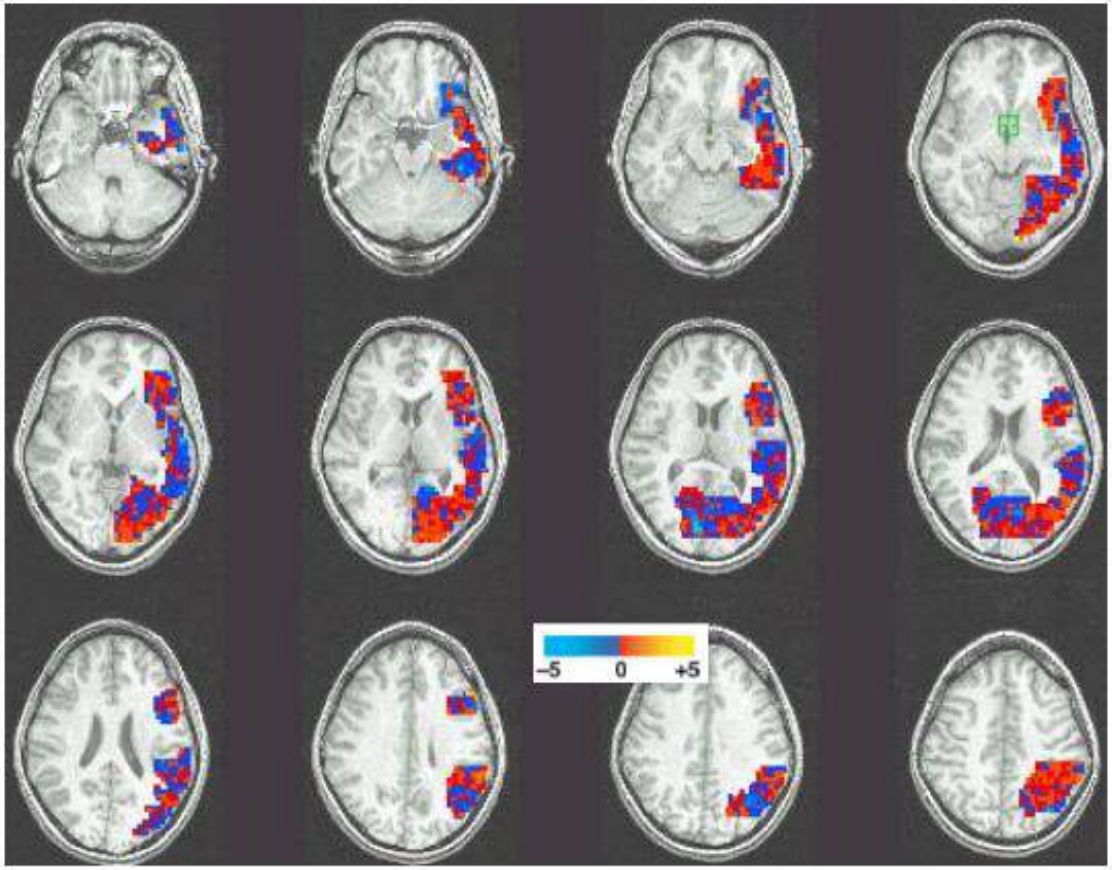
15,000 voxels/image

Non-invasive, save

Measures Blood Oxygen Level Dependent response (BOLD)



Gaussian Naïve Bayes: Learned $\mu_{voxel,word}$



[Mitchell et al.]

15,000 voxels or features

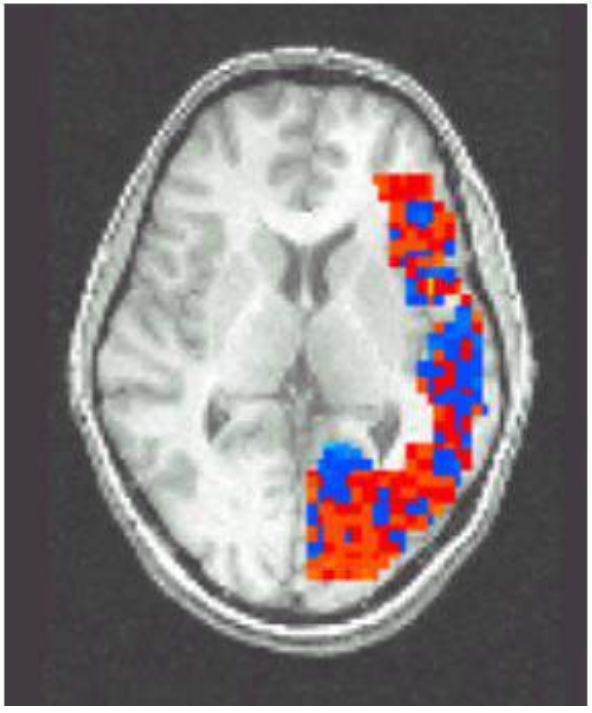
10 training examples or subjects per class

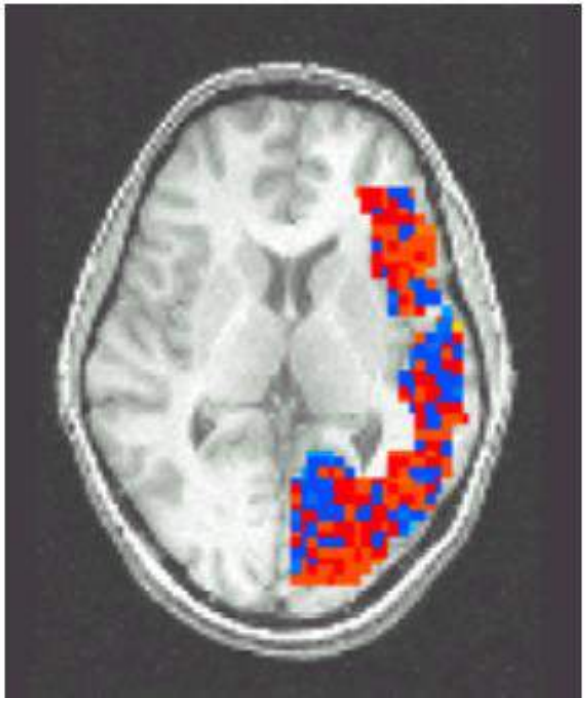
Learned Naïve Bayes Models –
Means for P(BrainActivity | WordCategory)

Pairwise classification accuracy: 85% [Mitchell et al.]

People words

Animal words





Practical Issues of Bayesian learning

- Require initial knowledge of many probabilities
  - Often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions.
- Significant computational cost required to determine the Bayes optimal hypothesis in the general case (linear in the number of candidate hypotheses)

Some references for more examples

- Movie Review:
  https://www.youtube.com/watch?time_continue=16&v=EGKeC2S44Rs

- Spam mails by Prof. Andrew Ng:

https://www.youtube.com/watch?v=z5UQyCESW64

https://www.youtube.com/watch?v=NFd0ZQk5bR4

- NLP by Prof. Dan Jurafsky, Stanford:
  https://www.youtube.com/watch?v=Fmu65a0v6Sw

In our next session

We will cover:

Bishop

- 4.1 Discriminant Functions,(4.1.1, 4.1.2)
- 4.3 Probabilistic Discriminative Classifiers,
- 4.3.1, 4.3.2 logistic regression
- Difference between Generative and Discriminative classifier