Principles of Machine Learning

Vasant Honavar
Artificial Intelligence Research Laboratory
College of Information Sciences and Technology
Center for Big Data Analytics and Discovery Informatics
Graduate Programs in IST, Computer Science, Bioinformatics & Genomics, Neuroscience
The Pennsylvania State University

vhonavar@ist.psu.edu
http://ailab.ist.psu.edu
http://faculty.ist.psu.edu/vhonavar

Research Interests

- Machine learning: Statistical, information theoretic, linguistic and structural approaches to machine learning; learning predictive relationships from sequential, graph-structured, multi-relational, multimodal, partially specified, partially labeled, distributed data, linked data
- Causal Inference: Causal inference from disparate experimental and observational studies, causal inference from relational data, causal inference from temporal data
- Knowledge Representation and Inference: Logical, probabilistic, and decision-theoretic knowledge representation and inference; federated knowledge bases; selective information sharing; federated services; representing and reasoning about qualitative preferences
- Applied Informatics
  - Bioinformatics: Macromolecular structure and function, analysis, inference, modeling, and prediction of macromolecular (protein-protein , protein-RNA, and protein-DNA) interaction networks and interfaces, immune networks, etc.
  - Health Informatics: Predictive and causal modeling of health outcomes from patient (health records, genomics, socio-economic, environmental) data
  - Brain Informatics: Modeling and analysis of structure and dynamics of brain networks from fMRI data
- Algorithmic Discovery:
  - Algorithmic abstractions of scientific domains
  - Representations of scientific artifacts (experiments, data, models, assumptions, hypotheses, theories ⋯)

What is this course about?

- Why should machines learn?
- When can Machines Learn?
- Why can Machines Learn?
- How can Machines Learn?
- How can Machines Learn better?

Course Overview

- Background and Motivation
- Statistical Machine Learning Theory and Applications
- Algorithmic Learning Theory and Applications
- Advanced Machine Learning Algorithms Design and Applications
- Machine Learning for Predictive Modeling from Big Data

Course Overview

- Background and Motivation
- Statistical Machine Learning Theory and Applications
  - Decision Theoretic Foundations
  - Probabilistic Generative Models
  - Discriminative Models
  - Representative Algorithms
- Algorithmic Learning Theory and Applications
  - Mistake Bound Models
  - PAC Model – sample complexity, easy and hard learning problems, how to turn hard learning problems into easy ones
  - Learning under helpful distributions
  - Representative Algorithms

Course Overview

- Advanced Machine Learning Algorithms Design and Applications
  - Probabilistic graphical models
  - Kernel machines
  - Deep learning
  - Multi-instance, multi-label, and structured label learning
  - Grammar learning
  - Causal models
- Machine Learning for Predictive Modeling from Big Data
  - Learning from large, distributed data
  - Learning from ultra high dimensional data
  - Learning from multi-modal data
  - Learning from multi-granularity data
  - Platforms and tools

Course Staff

Instructor

Vasant Honavar

Professor and Edward Frymoyer Chair of Information Sciences and Technology

Graduate Faculty:

- Computer Science and Engineering
- Bioinformatics and Genomics
- Neuroscience
- Operations Research
- Information Sciences and Technology

http://faculty.ist.psu.edu/vhonavar

vhonavar@ist.psu.edu

Office hours: 1pm to 2pm, Mon, Wed 301-A IST

TA: Sam Gur, szg180@psu.edu

Prerequisites

- Conceptual foundations of Computing
- Programming
- Mathematics
  - Set theory, logic, probability, calculus
- Data structures
  - Lists, trees, graphs
- Basics of Design and Analysis of Algorithms
- Technical writing and presentation

- Course materials
  http://faculty.ist.psu.edu/vhonavar/Courses/ml/homepage.html
  - Lecture notes, Recommended Readings, Programming resources
- Useful (*not* required) Reference Texts:
  - Machine Learning, Theodoridis
  - Machine Learning, Murphy
  - Bayesian Reasoning and Machine Learning, Barber
  - Pattern Recognition and Machine Learning, Bishop
  - A Probabilistic Theory of Pattern Recognition, Devroye, Gyorfi, and Lugosi
  - Elements of Statistical Learning, Hastie and Tibshirani
  - Machine Learning, Natarajan
  - An introduction to Computational Learning Theory, Kearns and Vazirani
  - Foundations of Machine Learning, Mohri, Rostamizadeh, and Talwalkar
  - Learning and Generalization, Vidyasagar
  - Statistical Learning Theory, Vapnik
  - Learning with Kernels, Skolkopf and Smola
  - Mining of Massive Data Sets, Rajaraman and Ullman
  - Learning Bayesian Networks, Neapolitan
  - Probabilistic Graphical Models, Koller and Friedman
  - Deep Learning, Bengio and Goodfellow

Course Mechanics

- Grading
  - Problem Sets
  - **Projects**
  - Exams
  - **Class participation**
- Academic Honesty
  - University policy on academic dishonesty
  - Problem sets, labs, term project, collaboration
- Disability

Transformative role of computation

- Computation offers the best formalism we have for understanding how information is acquired, processed, and used by
  - Computers
  - Brains
  - Genomes
  - Organizations
  - Societies
- Computation : cognitive science :: calculus : physics
- Computation: biology :: calculus : physics
- Computation: social science :: calculus : physics
- Algorithms as theories
  - We will have a theory of intelligence when we have computer programs (information processing models) that display intelligence

Machine learning is a subfield of artificial intelligence

AI is about

- Study of computational models of intelligence
- Falsifiable hypotheses about intelligent behavior
- Construction of intelligent artifacts
- Mechanization of tasks requiring intelligence
- Exploring the design space of intelligent systems

Why should machines learn?

Practical

- Intelligent behavior requires knowledge
- Explicitly specifying the knowledge needed for specific tasks is hard, and often infeasible
- If we can get machines to acquire the knowledge needed for particular tasks from observations (data), interactions (experiments), we can
  - Dramatically reduce the cost of developing intelligent systems
  - Automate aspects of scientific discovery
  - …

Machine Learning is most useful when

- the structure of the task is not well understood but representative data or interactions with the environment are available
- task (or parameters) change dynamically

Why should machines learn? – Applications

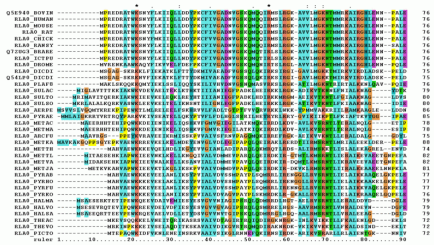
- Scientific
  - Identifying sequence correlates of protein function, predicting potential adverse drug interactions…
  - Understanding the relationship between genetic, environmental, and behavioral characteristics that contribute to health or disease
- Medicine
  - Diagnosing diseases from symptoms, test results (e.g. pneumonia, pap smears)
- Education
  - Customizing educational content and delivery to optimize learning outcomes

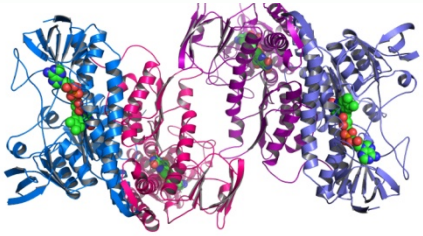
Why should machines learn? – Applications

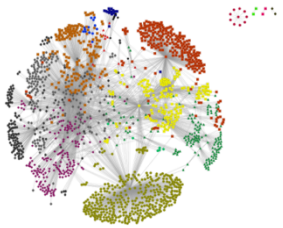
- Agriculture
  - Precision farming
- Business
  - Fraud detection (e.g. credit cards, phone calls)
  - Product recommendation (e.g. Google, Amazon, Netflix)
  - Stock trading
- Technology
  - Self-driving vehicles
  - Natural language conversation
  - Computer vision
  - Video understanding

Machine learning is essential for extracting knowledge from big data

Omics







Digital Media

VIDEO

BLOGS

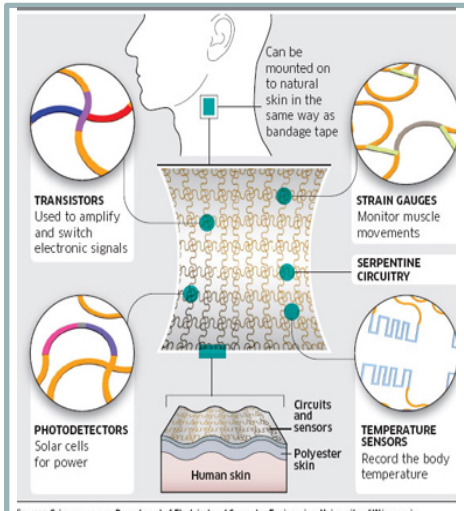
MOBILE

VOIP

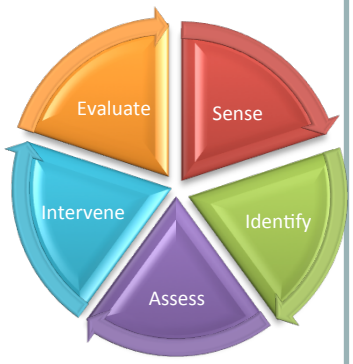
EMAIL

IM

Human Sensors



Health Care



Why should machines learn? – Science of learning

Information processing models can provide useful insights into

- How humans and animals learn
- Information requirements of learning tasks
- The precise conditions under which learning is possible
- Inherent difficulty of learning tasks
- How to improve learning – e.g. value of active versus passive learning
- Computational architectures for learning

Machine Learning – related disciplines

- Applied Statistics
  - Emphasizes statistical models of data
  - Methods typically applied to small data sets
  - Often done by a statistician increasingly assisted by a computer
- Machine learning
  - Relies on (often, but not always statistical) inference from data and knowledge (when available)
  - Emphasizes efficient data structures and algorithms for learning from data
  - Characterizing what can be learned and under what conditions
  - Obtaining guarantees regarding the quality of learned models
  - Scalability to large, complex data sets (big data)

What is Machine Learning?

- A program *M* is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance as measured by *P* on tasks in *T* in an environment *Z* improves with experience *E*.

Example 1

T – cancer diagnosis

E – a set of diagnosed cases

P – accuracy of diagnosis on new cases

Z – noisy measurements, occasionally misdiagnosed training cases

M – a program that runs on a general purpose computer

What is Machine Learning?

Example 2

T – recommending movies e.g., on Netflix

E – movie ratings data from individuals

P – accuracy of predicted movie ratings

10% improvement in prediction accuracy − $1 million prize

What is Machine Learning?

Example 3

T – Predicting protein-RNA interactions

E – A data set of known interactions

P – accuracy of predicted interactions

What is Machine Learning?

Example 4

T – Reconstructing functional connectivity of brains from brain activity (e.g., fMRI) data

E – fMRI data

P – accuracy of the reconstructed network

What is Machine Learning?

Example 5

T – solving integral calculus problems, given rules of integral calculus

E – a set of solved problems

P – score on test consisting of problems not in *E*

What is Machine Learning?

Example 6

T – predicting the risk of a disease before the onset of clinical symptoms

E – longitudinal gut microbiome data coupled with diagnostic tests

P – accuracy of predictions

What is Machine Learning?

Example 7

T – predicting sleep quality from actigraphy data

E – actigraphy data with sleep stage labels

P – accuracy of predictions

What is Machine Learning?

Example 8

T – Uncovering the causal relationship between exercise, diet and diabetes

E – Data from observations and interventions (changes in diet, exercise)

P – accuracy of causal predictions

Key requirements

- There is a pattern to be learned
- There are data to learn from

Applicant information:

age	23 years
gender	male
annual salary	$30,000
years in residence	1 year
years in job	1 year
current debt	$15,000
...	...

Approve credit?

Learning to approve credit

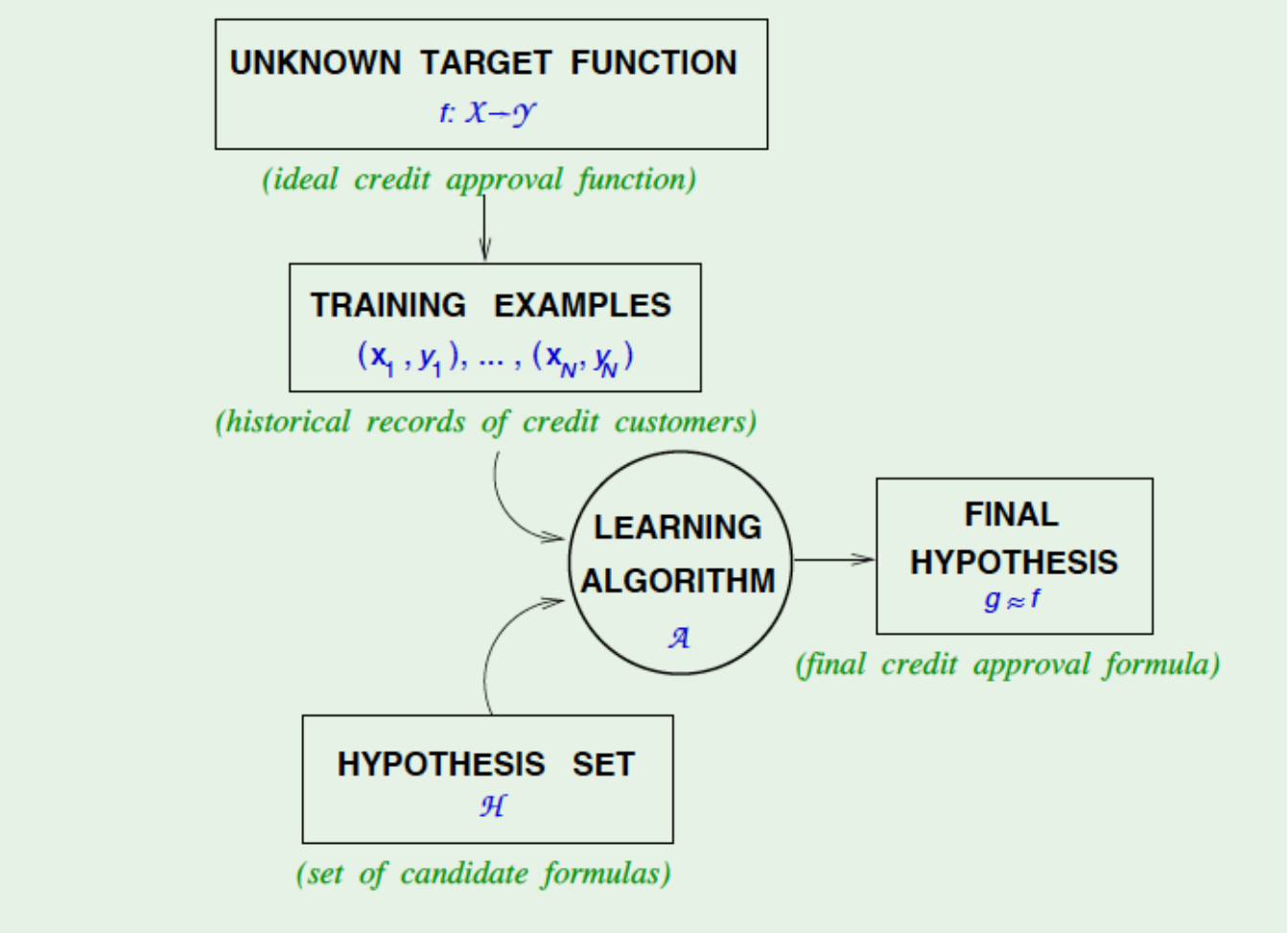
Formalization:

- Input: $\mathbf{x}$ (*customer application*)
- Output: y (*good/bad customer?*)
- Target function: $f : \mathcal{X} \rightarrow \mathcal{Y}$ (*ideal credit approval formula*)
- Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$ (*historical records*)

$\downarrow \quad \downarrow \quad \downarrow$

- Hypothesis: $g : \mathcal{X} \rightarrow \mathcal{Y}$ (*formula to be used*)

Learning to approve to credit



Canonical Learning Problems

Supervised Learning:

- Given labeled samples, predict labels on future samples
  - Classification
  - Regression
  - Time series prediction
- Many variants based on what constitutes a predictive model
- Many variants based on what constitutes a sample and a label
  - Multi instance learning
  - Multi-label learning
  - Multi-instance, multi-label learning
  - Distributional learning
- Many variants based on data type
  - Feature vectors
  - Sequences
  - Networks
  - Relations

Canonical Learning Problems

Unsupervised Learning: given unlabeled samples, discover representations, features, structure, etc.

- Clustering
- Compression
- Representation

Many variants based on what constitutes samples, data types

Semi-supervised Learning: given some labeled samples, and large amounts of unlabeled samples, predict labels of unlabeled samples

- Transductive (unlabeled samples given at learning time)
- Inductive (new unlabeled samples given at prediction time)

Multi-view learning:

- Given data from multiple sources about some underlying system, discover how they relate to each other;
- integrate the data to make predictions that are more reliable than those obtainable using any single data source

Canonical Learning Problems

Reinforcement Learning: Given the means of observing and interacting with an environment, learn how to act rationally

- Many variants based on what constitutes observation, interaction, and action

Causal inference: given observational and experimental data, causal assumptions, identify causal relations

- Identification
- Transport
- Meta analysis

Learning input – output functions: Classification, regression

Target function *f* – unknown to the learner – $f \in F$

Learner's hypothesis about what *f* might be – $h \in H$

H – hypothesis space

Instance space – *X* – domain of *f, h*

Output space – *Y* – range of *f*, *h*

Example – an ordered pair (x,y) where

$x \in X$ and $f(x) = y \in Y$

F and *H* may or may not be the same!

Training set *E* – a multi set of examples

Learning algorithm *L* – a procedure which given some *E,* outputs an $h \in H$

Learning input – output functions

- Must choose
  - Hypothesis language
  - Instance language
  - Semantics associated with both
- Machines can learn only functions that have *finite descriptions or representations* if we require learning programs to be halting programs

Examples:

- "Tom likes science fiction horror films"
- "F = ma"

Learning from Data

- Premise – A hypothesis (e.g., a classifier) that is consistent with a sufficiently large number of representative training examples is likely to accurately classify novel instances drawn from the same universe
- We can prove that this is an optimal approach (under reasonable assumptions) – more on this later
- When the number of examples is limited, the learner needs to be smarter (e.g., find a concise hypothesis that is consistent with the data)

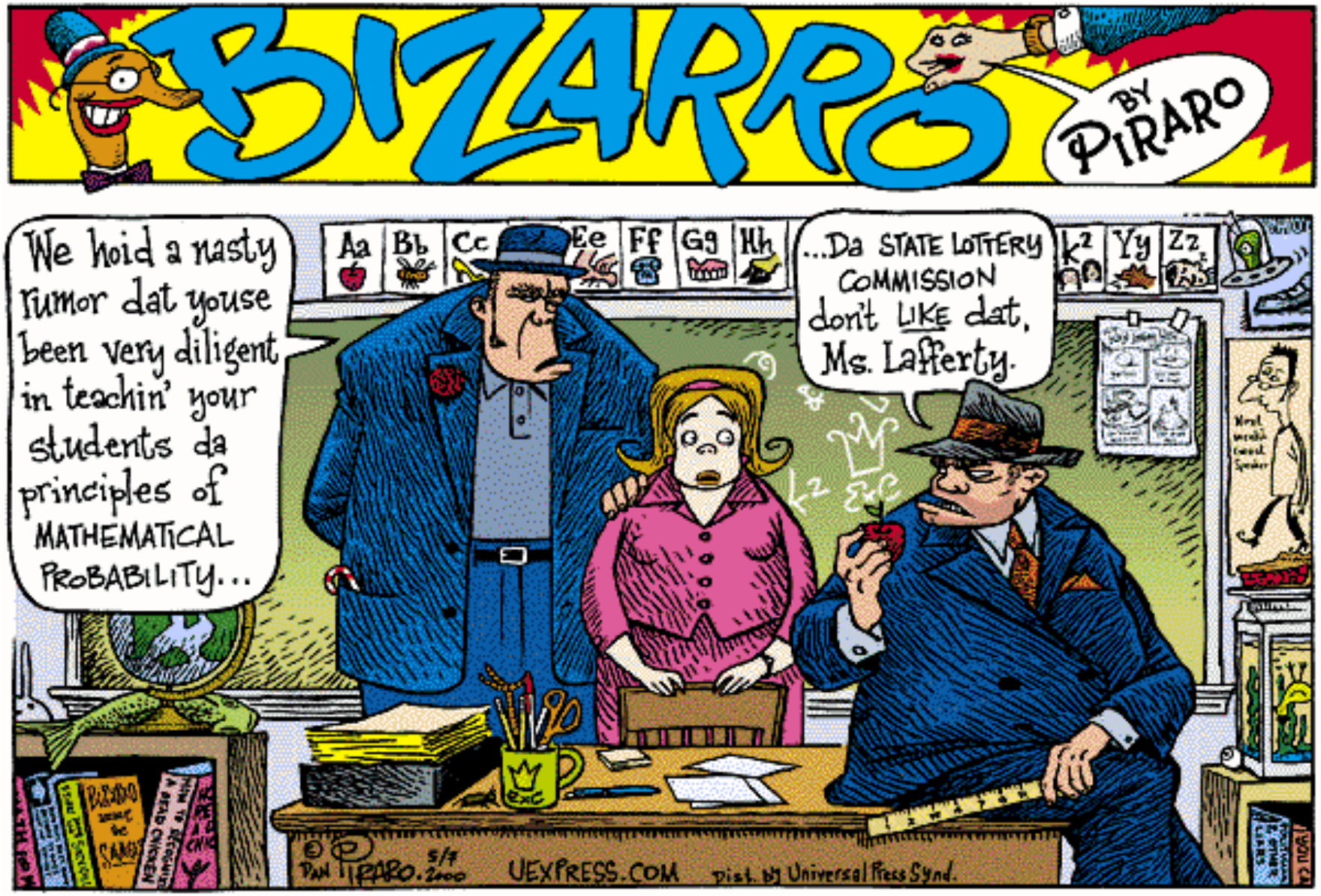
Learning as Probabilistic Inference

- Probabilistic inference provides a basis for updating beliefs based on evidence
- Learning is tantamount to updating beliefs about the world based on data.
  - Sound probabilistic basis for understanding many learning algorithms and designing new algorithms
  - Several practical reasoning and learning algorithms

Review

You should review material on

- Probability
- Random variables
- Distributions over random variables
- Independence and conditional independence


BIZARRO
BY PIRARO
We hoid a nasty rumor dat youse been very diligent in teachin' your students da principles of MATHEMATICAL PROBABILITY...
...Da STATE LOTTERY COMMISSION don't LIKE dat, Ms. Lafferty.
Aa Bb Cc Ee Ff Gg Hh
Yy Zz
PIRARO 5/7 2000
UEXPRESS.COM
Dist. by Universal Press Synd.

Representing and Reasoning under Uncertainty

- Probability Theory provides a framework for representing and reasoning under uncertainty
  - Represent beliefs about the world as sentences (much like in propositional logic)
  - Associate probabilities with sentences
  - Reason by manipulating sentences according to sound rules of probabilistic inference
  - Results of inference are probabilities associated with conclusions that are justified by beliefs and data (observations)

Probabilistic inference

- Beliefs:
  - If Oksana studies, there is an 60% chance that she will pass the test; and a 40 percent chance that she will not.
  - If she does not study, there is 20% percent chance that she will pass the test and 80% chance that she will not.
- Observation: Oksana did not study.
- Example Inference task:
  - What is the chance that Oksana will pass the test?
  - What is the chance that she will fail?
- Probability theory generalizes propositional logic
  - Probability theory associates probabilities that lie in the interval [0,1] as opposed to 0 or 1 (exclusively)

Sources of uncertainty

Uncertainty modeled by Probabilistic assertions may

- In a deterministic world be due to
  - Laziness: failure to enumerate exceptions, qualifications, etc. that may be too numerous to state explicitly
  - Sensory limitations
  - Ignorance: lack of relevant facts etc.
- In a stochastic world be due to
  - Inherent uncertainty (as in quantum physics)

The framework is agnostic about the source of uncertainty

The world according to Agent Bob

- An atomic event or world state is a complete specification of the state of the agent's world.
- Event set is a set of mutually exclusive and exhaustive possible world states (relative to an agent's representational commitments and sensing abilities)
- From the point of view of an agent Bob who can sense only 3 colors and 2 shapes, the world can be in only one of 6 states
- Atomic events (world states) are
  - mutually exclusive
  - exhaustive

Semantics: Probability as a subjective measure of belief

- Suppose there are 3 agents – Sanghack, Sam, Aria, in a world where a fair dice has been tossed.
- Sanghack observes that the outcome is a "6" and whispers to Sam that the outcome is "even" but
- Aria knows nothing about the outcome.

Set of possible mutually exclusive and exhaustive world states
= {1, 2, 3, 4, 5, 6}

Set of possible states of the world based on what Sam knows
= {2, 4, 6}

Probability as a subjective measure of belief

Probability is a measure over all of the world states that are possible, or simply, possible worlds, given what an agent knows

$$Possibleworlds_{Sanghack} = \{6\},\ Possibleworlds_{Sam} = \{2,4,6\}$$
$$Possibleworlds_{Aria} = \{1,2,3,4,5,6\}$$

$$\Pr_{Sanghack}(worldstate = 6) = 1$$
$$\Pr_{Sam}(worldstate = 6) = \frac{1}{3}$$
$$\Pr_{Aria}(worldstate = 6) = \frac{1}{6}$$

Sanghack, Sam, and Aria assign different beliefs to the same world state because of differences in what they have observed or have been told!

Random variables

- The "domain" of a random variable is the set of values it can take. The values are mutually exclusive and exhaustive.
- The domain of a Boolean random variable X is {true, false} or {1, 0}
- Discrete random variables take values from a countable domain.
  - The domain of the random variable Color may be {Red, Green}.
  - If E = {(Red, Square), (Green, Circle), (Red, Circle), (Green, Square)}, the proposition (Color = Red) is True in the world states {(Red, Square), (Red, Circle)}.
  - Each state of a discrete random variable corresponds to a proposition e.g., (Color = Red)

Syntax

- Basic element: random variable
  - Similar to propositional (Boolean) logic: possible worlds defined by assignment of values to random variables.
  - *Cavity* (do I have a cavity?)
  - *Weather* is one of <*sunny, rainy, cloudy, snow*>
  - Domain values must be exhaustive and mutually exclusive
- Elementary proposition constructed by assignment of a value to a random variable
  - *Weather = sunny*, *Cavity = false*
  - (abbreviated as ¬*cavity*)
- Complex propositions formed from elementary propositions and standard logical connectives
  - *Weather = sunny* ∨ *Cavity = false*

Syntax and Semantics

- Atomic event: A complete specification of the state of the world about which the agent is uncertain
- Atomic events correspond to a possible worlds (much like in the case of propositional logic)

  E.g., if the world consists of only two Boolean variables *Cavity* and *Toothache*, then there are 4 distinct atomic events or 4 possible worlds:

  $Cavity = false \wedge Toothache = false$

  $Cavity = false \wedge Toothache = true$

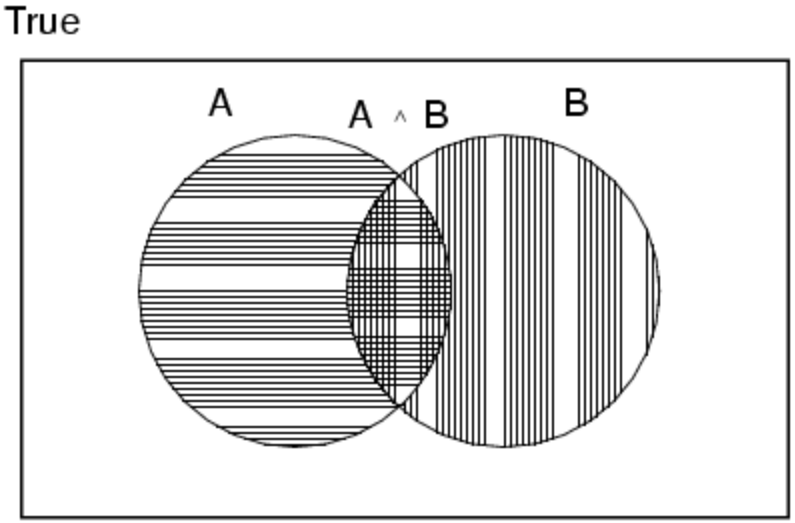
  $Cavity = true \wedge Toothache = false$

  $Cavity = true \wedge Toothache = true$

- Atomic events are mutually exclusive and exhaustive

Axioms of probability

- For any propositions *A, B*
  - $0 \leq P(A) \leq 1$
  - $P(true) = 1$ and $P(false) = 0$
  - $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$



Prior probability

- Prior or unconditional probabilities of propositions
  - P(*Cavity* = true) = 0.1 and P(*Weather* =Rainy) = 0.72 correspond to belief prior to arrival of any (new) evidence
- Probability distribution gives values for all possible assignments:
  - **P**(*Rainy*) = <0.72, 0.1, 0.08, 0.1>
  - Note that the probabilities sum to 1
- Joint probability distribution for a set of random variables gives the probability of every atomic event on those random variables
  - **P**(*Cavity, Weather*) = a 4 × 2 matrix of values

Joint probability distribution

- Joint probability distribution for a set of random variables gives the probability of every atomic event on those random variables

  - **P**(*Weather, Cavity*) = a 4 × 2 matrix of values:

Weather=	sunny	rainy	cloudy	snow
Cavity = true	0.144	0.02	0.016	0.02
Cavity = false	0.576	0.08	0.064	0.08

- Every question about a domain can be answered by the joint distribution

Inference using the joint distribution

	Toothache	¬ Toothache
Cavity	0.4	0.1
¬ Cavity	0.1	0.4

$$P(cavity) = P(cavity, Toothache) + P(cavity, \neg Toothache)$$

Conditional probability

- Conditional or posterior probabilities
  - P(*Cavity* | *Toothache*) = 0.8

    (note *Cavity* is shorthand for *Cavity = True*)

  Probability of *Cavity* given *Toothache*

- Notation for conditional distributions:

  P(*Cavity* | *Toothache*) = 2-element vector of 2-element vectors)

  P(*Cavity* | *Toothache, Cavity*) = 1

- New evidence may be irrelevant (Probability of Cavity given Toothache is independent of Weather)

  P(*Cavity | Toothache, Sunny*) = P(*Cavity* | *Toothache*) = 0.8

Conditional probability

- Definition of conditional probability:

$$P(a \mid b) = P(a \wedge b) / P(b) \text{ if } P(b) > 0$$

- Product rule gives an alternative formulation:

$$P(a \wedge b) = P(a \mid b)\, P(b) = P(b \mid a)\, P(a)$$

Example:

- Suppose I have two coins – one a normal fair coin, and the other a rigged coin (with heads on both sides). I pick a coin at *random, toss it,* and tell you that the outcome of the toss is a Head.
- What is the probability that I am looking at a fair coin?

Conditional probability

- A general version holds for whole distributions, e.g.,

  $\mathbf{P}(\textit{Weather,Cavity}) = \mathbf{P}(\textit{Weather} \mid \textit{Cavity})\ \mathbf{P}(\textit{Cavity})$

- View as a compact notation for a set of 4 × 2 equations, not matrix multiplication

- Chain rule is derived by successive application of product rule:

$$\mathbf{P}(X_1, \ldots, X_n) = \mathbf{P}(X_1,\ldots,X_{n-1})\ \mathbf{P}(X_n \mid X_1,\ldots,X_{n-1})$$
$$= \mathbf{P}(X_1,\ldots,X_{n-2})\ \mathbf{P}(X_{n-1} \mid X_1,\ldots,X_{n-2})\ \mathbf{P}(X_n \mid X_1,\ldots,X_{n-1})$$
$$= \ldots$$
$$= \pi_i\ \mathbf{P}(X_i \mid X_1, \ldots, X_{i-1}) \text{ (i ranges from 1 to n)}$$

Probability as a measure over possible worlds

- Suppose I have two coins – one a normal fair coin, and the other with 2 heads. I pick a coin at *random* and toss it. What is the probability that the outcome is a head?

$$\Omega = \{(Fair, H), (Fair, T), (Rigged, H), (Rigged, T)\}$$

$$\mu = \left\{\frac{1}{4}, \frac{1}{4}, \frac{1}{2}, 0\right\}$$

$$\Pr(H) = \sum_{\omega| = H} \mu(\omega) = \frac{1}{4} + \frac{1}{2} = \frac{3}{4}$$

Conditional probability as a Measure over Possible worlds not ruled out by evidence

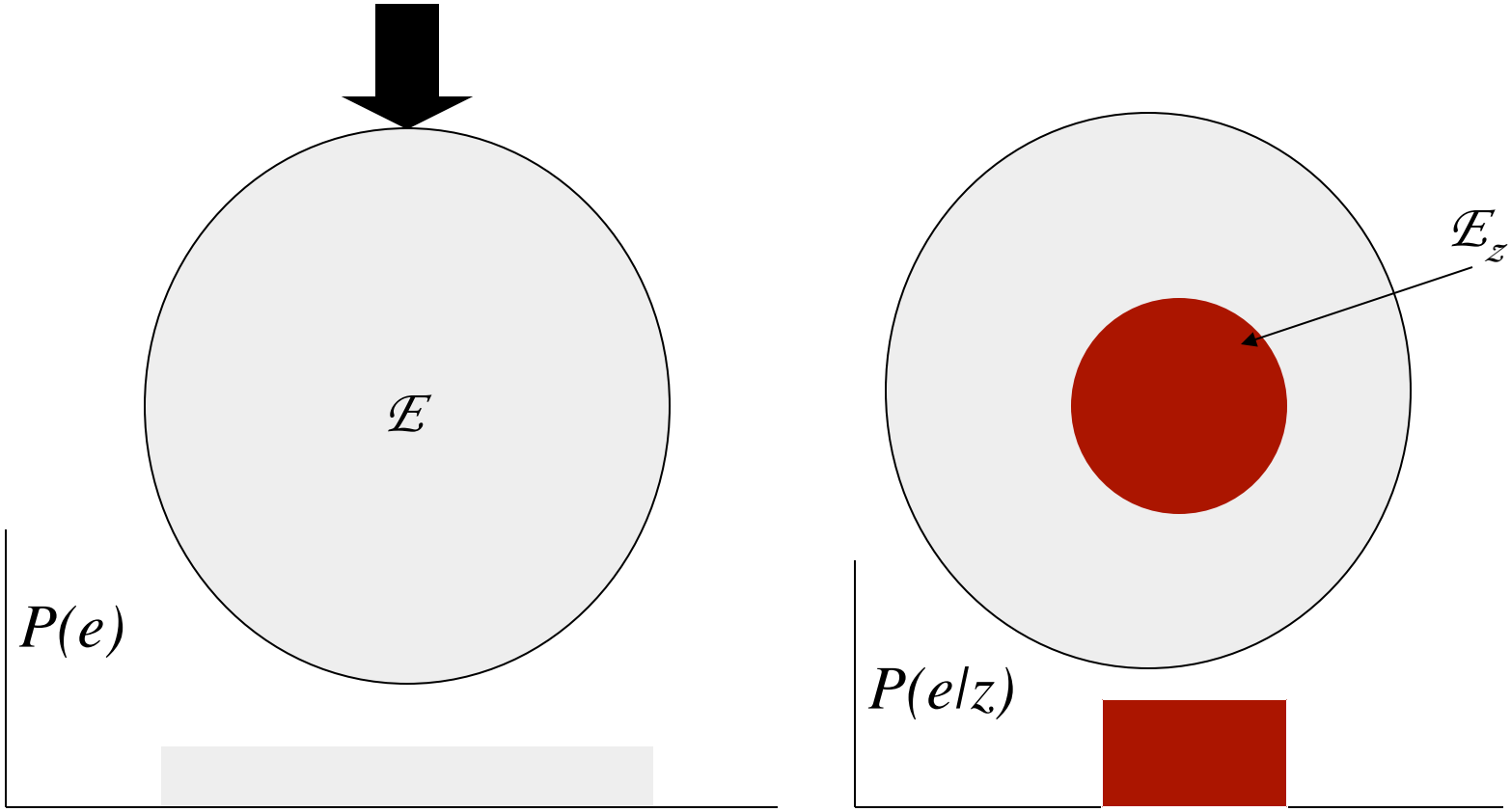
- A given piece of evidence e rules out all possible worlds that are incompatible with e or selects the possible worlds in which e is $True$. Evidence e *induces* a new measure μ_e.

$$\mu_e(\omega) = \begin{cases} \frac{1}{P(e)}\mu(\omega) \text{ if } \omega \models e \\ 0 \text{ if } \omega \not\models e \end{cases}$$

$$P(h|e) = \sum_{\omega \models h} \mu_e(\omega) = \frac{1}{P(e)} \sum_{\omega \models h \wedge e} \mu(\omega) = \frac{P(h \wedge e)}{P(e)}$$

Effect of Evidence on Possible worlds

Evidence z e.g., (color = red) rules out some assignments of values to some of the random variables



Inference by enumeration

- Start with the joint probability distribution:

	toothache		¬ *toothache*	
	catch	¬ *catch*	*catch*	¬ *catch*
cavity	.108	.012	.072	.008
¬ *cavity*	.016	.064	.144	.576

- For any proposition ϕ, sum the atomic events where it is true: $P(\phi) = \Sigma_{\omega:\omega \models \phi} P(\omega)$

- P(*toothache*) = 0.108 + 0.012 + 0.016 + 0.064 = 0.2

Inference by enumeration

- Start with the joint probability distribution:

	toothache		¬ *toothache*	
	catch	¬ *catch*	*catch*	¬ *catch*
cavity	.108	.012	.072	.008
¬ *cavity*	.016	.064	.144	.576

- Can also compute conditional probabilities:

$$P(\neg cavity \mid toothache) = \frac{P(\neg cavity \wedge toothache)}{P(toothache)}$$

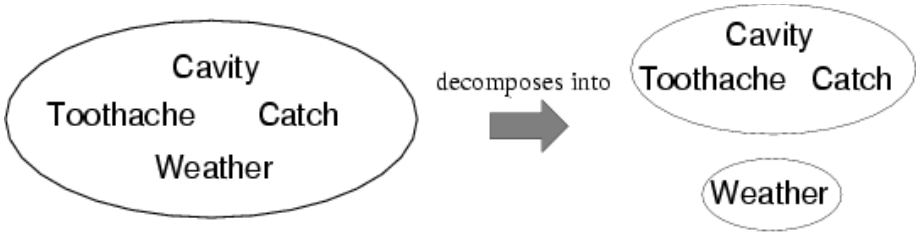
$$= \frac{0.016+0.064}{0.108 + 0.012 + 0.016 + 0.064}$$

$$= 0.4$$

Independence

- *A* and *B* are independent iff

  P(*A*|*B*) = **P**(*A*) or **P**(*B*|*A*) = **P**(*B*) or **P**(A, B) = **P**(*A*) **P**(*B*)



P(*Toothache, Catch, Cavity, Weather*)

= **P**(*Toothache, Catch, Cavity*) **P**(*Weather*)

- 32 entries reduced to 12;
- *n* independent variables, $O(2^n)$ reduced to $O(n)$
- Absolute independence powerful but rare
- How can we manage a large numbers of variables?

Conditional Independence

- *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*:

- $P(X \mid Y, Z) = P(X \mid Z)$ that is,

$$(\forall x_i, y_j, z_k) P(X = x_i \mid Y = y_j, Z = z_k) = P(X = x_i \mid Z = z_k)$$

Independence and Conditional Independence

Let $\mathbf{Z}_1,...\mathbf{Z}_n$ and $\mathbf{W}$ be pairwise disjoint sets of random variables on a given event space.

$\mathbf{Z}_1,...,\mathbf{Z}_n$ are mutually independent given $\mathbf{W}$ if

$$P(\mathbf{Z}_1 \cup .... \cup \mathbf{Z}_n | \mathbf{W}) = \prod_{i=1}^{n} P(\mathbf{Z}_i | \mathbf{W})$$

$P(\mathbf{Z}_1 | \mathbf{Z}_2 \cup \mathbf{W}) = P(\mathbf{Z}_1 | \mathbf{W})$ if $\mathbf{Z}_1$ and $\mathbf{Z}_2$ are independent.

Note that these represent sets of equations, for all possible value assignments to random variables

Independence Properties of Random Variables

Let $\mathbf{W}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be pairwise disjoint sets of random variables on a given event space.

Let $I(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ denote that $\mathbf{X}$ and $\mathbf{Z}$ are *independent* given $\mathbf{Y}$.

That is, $P(\mathbf{X} \cup \mathbf{Z} | \mathbf{Y}) = P(\mathbf{X}|\mathbf{Y})P(\mathbf{Z}|\mathbf{Y})$, or $P(\mathbf{X}|\mathbf{Y} \cup \mathbf{Z}) = P(\mathbf{X}|\mathbf{Y})$. Then :

a. $I(\mathbf{X}, \mathbf{Z}, \mathbf{Y}) \Rightarrow I(\mathbf{Y}, \mathbf{Z}, \mathbf{X})$

b. $I(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W}) \Rightarrow I(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$

c. $I(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W}) \Rightarrow I(\mathbf{X}, \mathbf{Z} \cup \mathbf{W}, \mathbf{Y})$

d. $I(\mathbf{X}, \mathbf{Z}, \mathbf{Y}) \wedge I(\mathbf{X}, \mathbf{Z} \cup \mathbf{Y}, \mathbf{W}) \Rightarrow I(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W})$

Proof : Follows from definition of *independence*.

Quick proof that independence is symmetric

- Assume: $P(X|Y, Z) = P(X|Y)$
- X and Z are independent given Y

$$P(Z \mid X,Y) = \frac{P(X,Y \mid Z)P(Z)}{P(X,Y)} \quad \textit{(Bayes's Rule)}$$

$$= \frac{P(Y \mid Z)P(X \mid Y,Z)P(Z)}{P(X \mid Y)P(Y)} \quad \textit{(Chain Rule)}$$

$$= \frac{P(Y \mid Z)P(X \mid Y)P(Z)}{P(X \mid Y)P(Y)} \quad \textit{(By Assumption)}$$

$$= \frac{P(Y \mid Z)P(Z)}{P(Y)} = P(Z \mid Y) \quad \textit{(Bayes's Rule)}$$

Bayes Rule

Does patient have cancer or not?

A patient takes a lab test and the result comes back positive. 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. Furthermore, .008 of the entire population have this cancer.

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

$$P(+ \mid cancer) = \qquad P(- \mid cancer) =$$

$$P(+ \mid \neg cancer) = \qquad P(- \mid \neg cancer) =$$

Bayes Rule

Does patient have cancer or not?

$$P(cancer) = 0.008 \qquad P(\neg cancer) = 0.992$$

$$P(+ \mid cancer) = 0.98 \qquad P(- \mid cancer) = 0.02$$

$$P(+ \mid \neg cancer) = 0.03 \qquad P(- \mid \neg cancer) = 0.97$$

$$P(cancer|+) = \frac{P(+|cancer)P(cancer)}{P(+)};$$

$$P(\neg cancer|+) = \frac{P(+|\neg cancer)P(\neg cancer)}{P(+)}$$

$$P(cancer|+)P(+) = 0.98 \times 0.008 = 0.0078;$$

$$P(\neg cancer|+)P(+) = 0.03 \times 0.992 = 0.0298$$

$$P(+) = 0.0078 + 0.0298$$

$$P(cancer \mid +) = 0.21; \qquad P(\neg cancer \mid +) = 0.79$$

The patient, more likely than not, does not have cancer

Bayes Rule

- Product rule
  - $P(a \wedge b) = P(a \mid b)\, P(b) = P(b \mid a)\, P(a)$
  - Bayes' rule: $P(a \mid b) = P(b \mid a)\, P(a) / P(b)$

- In distribution form

$$\mathbf{P}(Y|X) = \mathbf{P}(X|Y)\, \mathbf{P}(Y) / \mathbf{P}(X) = \alpha \mathbf{P}(X|Y)\, \mathbf{P}(Y)$$

Decision Theoretic Foundations

- What is an “optimal” classifier?
- How can a classifier assign labels optimally?
- Can we build an optimal classifier?
- Example

Decision theoretic foundations of classification

Consider the problem of classifying an instance X into one of two mutually exclusive classes ω_1 or ω_2

$P(\omega_1 | X)$ = probability of class ω_1 given the evidence X

$P(\omega_2 | X)$ = probability of class ω_2 given the evidence X

What is the probability of error?

$P(error | X) = P(\omega_1 | X)$ if we choose ω_2

$= P(\omega_2 | X)$ if we choose ω_1

Minimum Error Classification

To minimize classification error

Choose ω_1 if $P(\omega_1|X) > P(\omega_2|X)$

Choose ω_2 if $P(\omega_2|X) > P(\omega_1|X)$

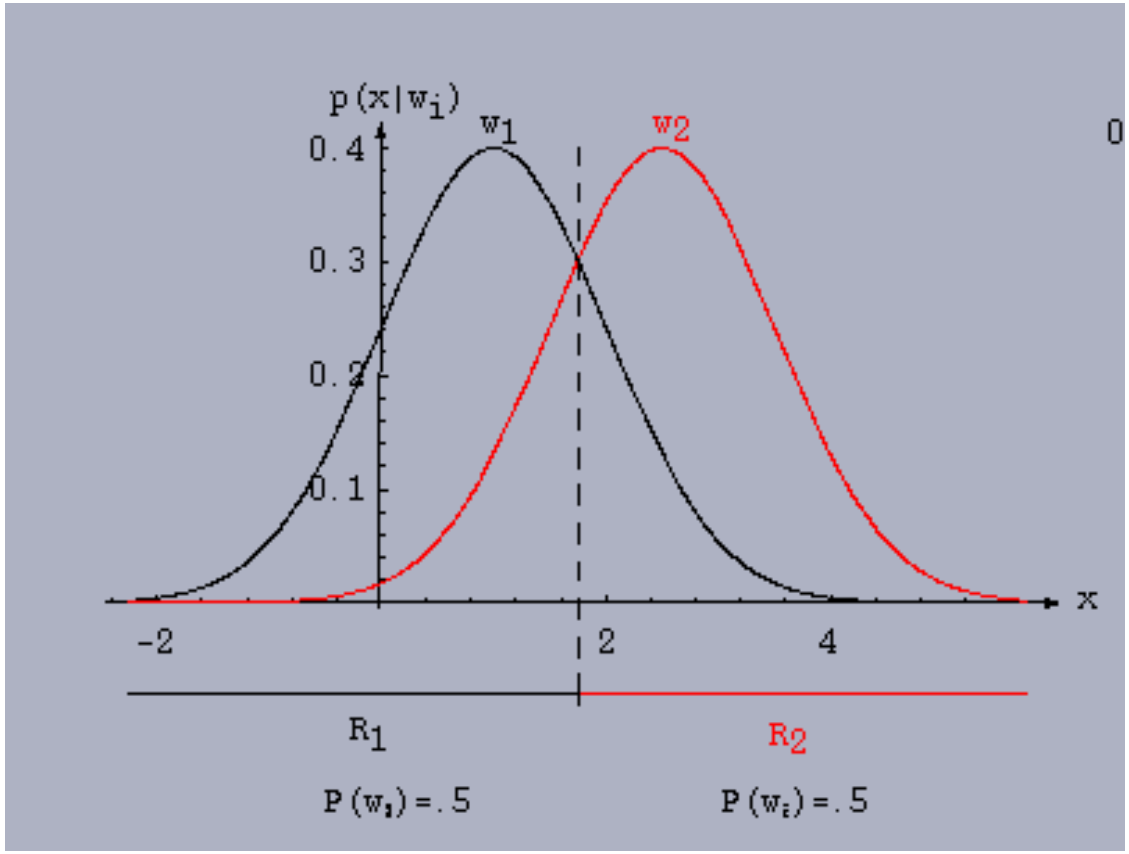
which yields

$$P(error \mid X) = \min\left[P(\omega_1|X), P(\omega_2|X)\right]$$

We have :

$$P(\omega_1|X) = P(X \mid \omega_1)P(\omega_1);$$

$$P(\omega_2|X) = P(X \mid \omega_2)P(\omega_2)$$



Choose ω_1 if $P(\omega_1|X) > P(\omega_2|X)$ i.e. $X \in R_1$

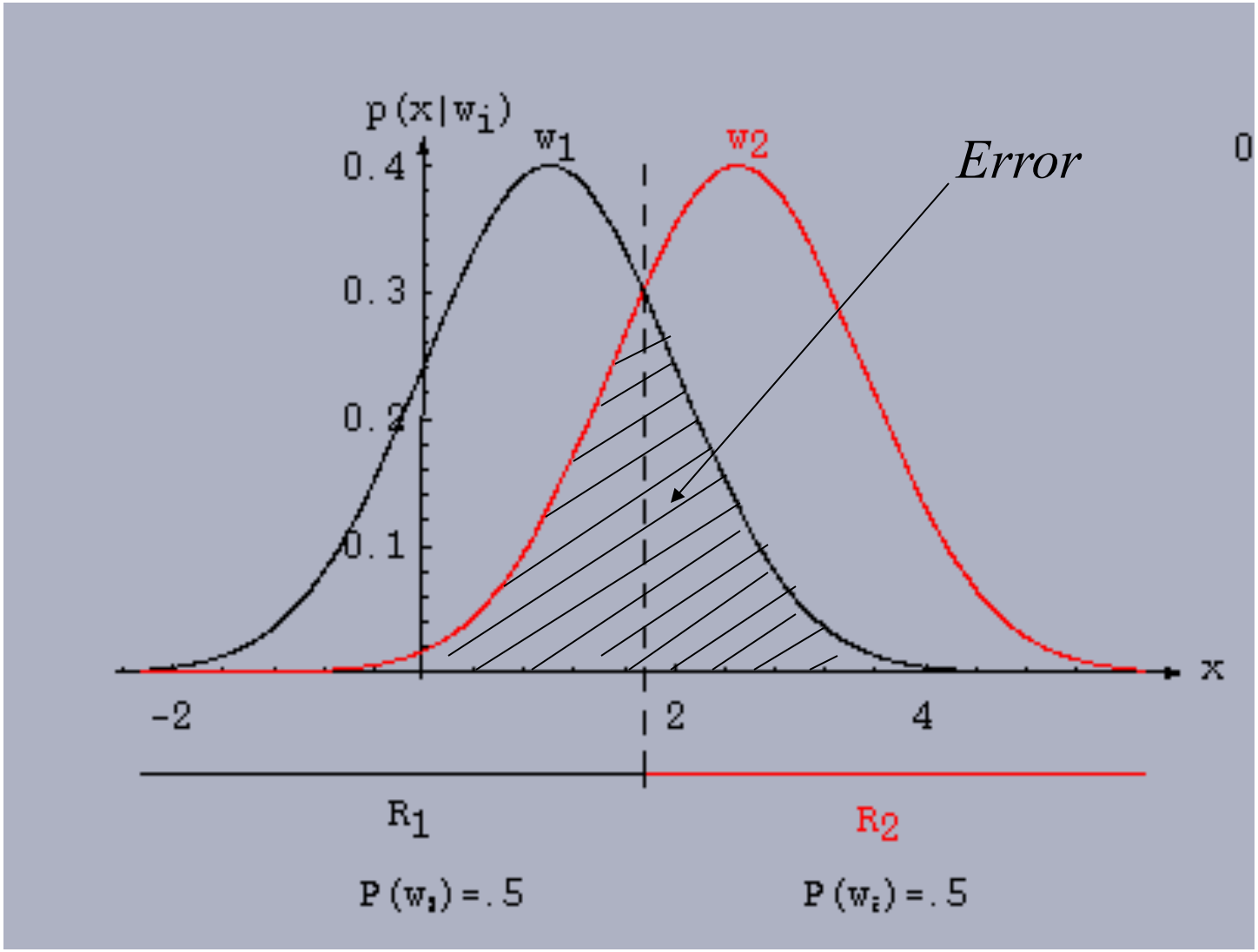
Choose ω_2 if $P(\omega_2|X) > P(\omega_1|X)$ i.e. $X \in R_2$

Optimality of Bayes Decision Rule

We can show that the Bayesian classifier

- is optimal in that it is guaranteed to minimize the probability of misclassification

Optimality of Bayes Decision Rule



Optimality of the Bayes Decision Rule

$$
\begin{aligned}
P_e &= P(x \in R_2, x \in \omega_1) + P(x \in R_1, x \in \omega_2) \\
&= P(x \in R_2 \mid \omega_1) P(\omega_1) + P(x \in R_1 \mid \omega_2) P(\omega_2) \\
&= P(\omega_1) \int_{R_2} p(x \mid \omega_1) dx + P(\omega_2) \int_{R_1} p(x \mid \omega_2) dx
\end{aligned}
$$

Applying Bayes Rule :

$$p(x \mid \omega_i) P(\omega_i) = P(\omega_i \mid x) p(x) = p(x, \omega_i)$$

$$P_e = \int_{R_2} P(\omega_1 \mid x) p(x) dx + \int_{R_1} P(\omega_2 \mid x) p(x) dx$$

Optimality of the Bayes Decision Rule

$$P_e = \int_{R_2} p(\omega_1 \mid x)p(x)dx + \int_{R_1} p(\omega_2 \mid x)p(x)dx$$

Because $R_1 \cup R_2$ covers the entire input space,

$$\int_{R_1} P(\omega_1 \mid x)p(x)dx + \int_{R_2} P(\omega_1 \mid x)p(x)dx = P(\omega_1)$$

$$P_e = P(\omega_1) - \int_{R_1} \left(P(\omega_1 \mid x) - P(\omega_2 \mid x)\right)p(x)dx$$

P_e is minimized by choosing

R_1 such that $P(\omega_1 \mid x) > P(\omega_2 \mid x)$

and

R_2 such that $P(\omega_2 \mid x) > P(\omega_1 \mid x)$

Optimality of Bayes Decision Rule

- The proof generalizes to multivariate input spaces
- Similar result can be proved in the case of discrete (as opposed to continuous) input spaces – replace integration over the input space by summation

Bayes Decision Rule yields Minimum Error Classification

To minimize classification error

Choose ω_1 if $P(\omega_1|X) > P(\omega_2|X)$

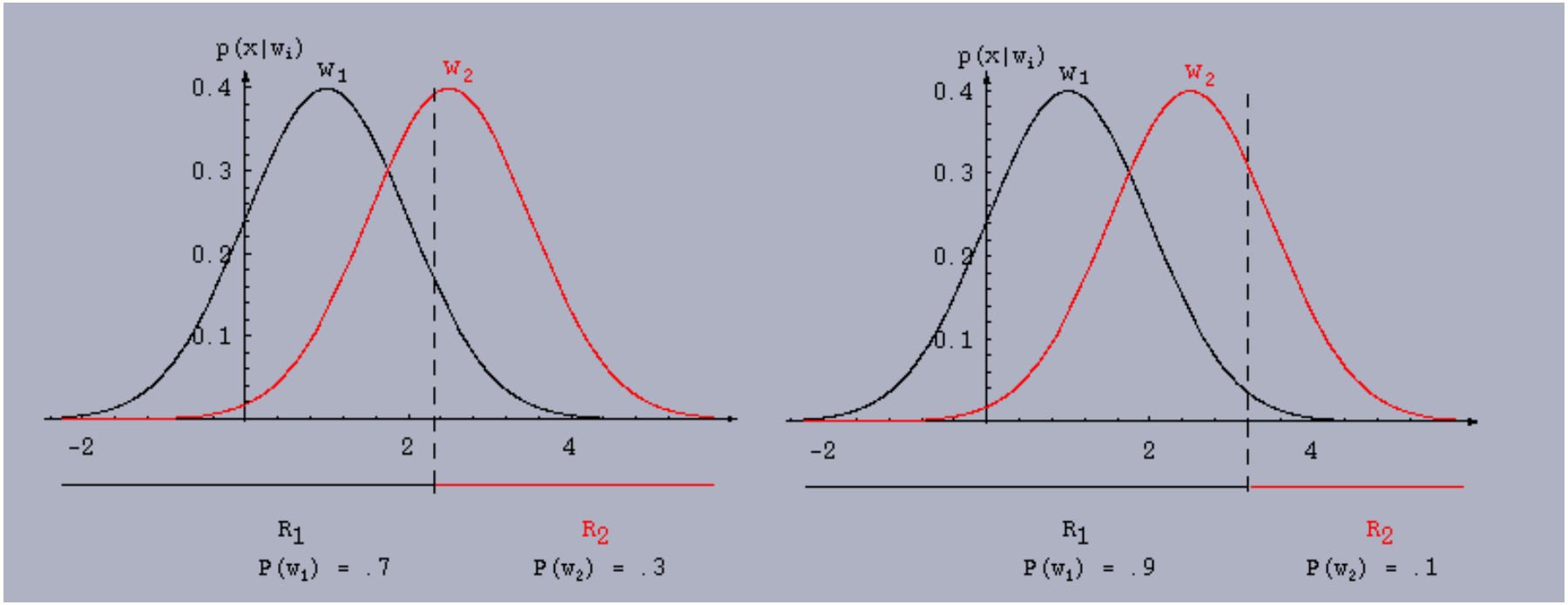
Choose ω_2 if $P(\omega_2|X) > P(\omega_1|X)$

which yields

$$P(error \mid X) = \min\left[P(\omega_1|X), P(\omega_2|X)\right]$$

Bayes Decision Rule

Behavior of Bayes decision rule as a function of prior probability of classes



Bayes Optimal Classifier

Classification rule that guarantees minimum error :

Choose ω_1 if $P(X \mid \omega_1)P(\omega_1) > P(X \mid \omega_2)P(\omega_2)$

Choose ω_2 if $P(X \mid \omega_2)P(\omega_2) > P(X \mid \omega_1)P(\omega_1)$

If $P(X \mid \omega_1) = P(X \mid \omega_2)$

classification depends entirely on $P(\omega_1)$ and $P(\omega_2)$

If $P(\omega_1) = P(\omega_2)$,

classification depends entirely on $P(X \mid \omega_1)$ and $P(X \mid \omega_2)$

Bayes classification rule combines the effect of the two terms optimally - so as to yield minimum error classification.

Generalization to multiple classes $c(X) = \arg\max_{\omega_j} P(\omega_j \mid X)$

Minimum Risk Classification

Let λ_{ij} = risk or cost associated with assigning an instance to class ω_j when the correct classification is ω_i

$R(\omega_i \mid X)$ = expected loss incurred in assigning X to class ω_i

$$R(\omega_1 \mid X) = \lambda_{11} P(\omega_1 \mid X) + \lambda_{21} P(\omega_2 \mid X)$$

$$R(\omega_2 \mid X) = \lambda_{12} P(\omega_1 \mid X) + \lambda_{22} P(\omega_2 \mid X)$$

Classification rule that guarantees minimum risk:

Choose ω_1 if $R(\omega_1 \mid X) < R(\omega_2 \mid X)$

Choose ω_2 if $R(\omega_2 \mid X) < R(\omega_1 \mid X)$

Flip a coin otherwise

Minimum Risk Classification

λ_{ij} = risk or cost associated with assigning an instance to class ω_j when the correct classification is ω_i

Ordinarily $(\lambda_{21} - \lambda_{22})$ and $(\lambda_{12} - \lambda_{11})$ are positive

(cost of being correct is less than the cost of error)

So we choose ω_1 if $\frac{P(X|\omega_1)}{P(X|\omega_2)} > \frac{(\lambda_{21} - \lambda_{22})}{(\lambda_{12} - \lambda_{11})} \frac{P(\omega_2)}{P(\omega_1)}$

Otherwise choose ω_2

Minimum error classification rule is a special case:

$$\lambda_{ij} = 0 \text{ if } i = j \text{ and } \lambda_{ij} = 1 \text{ if } i \neq j$$

This classification rule can be shown to be optimal in that it is guaranteed to minimize the risk of misclassification

Summary of Bayesian recipe for classification

λ_{ij} = risk or cost associated with assigning an instance to class ω_j when the correct classification is ω_i

Choose ω_1 if $\frac{P(X|\omega_1)}{P(X|\omega_2)} > \frac{(\lambda_{21}-\lambda_{22})}{(\lambda_{12}-\lambda_{11})}\frac{P(\omega_2)}{P(\omega_1)}$

Choose ω_2 if $\frac{P(X|\omega_1)}{P(X|\omega_2)} < \frac{(\lambda_{21}-\lambda_{22})}{(\lambda_{12}-\lambda_{11})}\frac{P(\omega_2)}{P(\omega_1)}$

Minimum error classification rule is a special case :

Choose ω_1 if $\frac{P(X|\omega_1)}{P(X|\omega_2)} > \frac{P(\omega_2)}{P(\omega_1)}$ Otherwise choose ω_2

Bayesian recipe for classification

Note that $P(\omega_i \mid \mathbf{x}) = \dfrac{P(\mathbf{x}|\omega_i)P(\omega_i)}{P(\mathbf{x})}$

Model $P(\mathbf{x} \mid \omega_1)$, $P(\mathbf{x}|\omega_2)$, $P(\omega_1)$, and $P(\omega_2)$

Using Bayes rule, choose ω_1 if $P(\mathbf{x} \mid \omega_1)P(\omega_1) > P(\mathbf{x}|\omega_2)P(\omega_2)$

Otherwise choose ω_2

Summary of Bayesian recipe for classification

- The Bayesian recipe is simple, optimal, and in principle, straightforward to apply
- To use this recipe in practice, we need to know $P(X|\omega_i)$ – the generative model for data for each class and $P(\omega_i)$ – the prior probabilities of classes
- Because these probabilities are unknown, we need to estimate them from data – or learn them!
- X is typically high-dimensional or may have complex structure
- Need to estimate $P(X|\omega_i)$ from data

Naïve Bayes Classifier

- How to learn $P(X \mid \omega_i)$?

One solution: Assume that the random variables in X are conditionally independent given the class.

- Result: Naïve Bayes classifier which performs optimally under certain assumptions
- A simple, practical learning algorithm grounded in Probability Theory

When to use

- Attributes that describe instances are likely to be conditionally independent given classification
- The data is insufficient to estimate all the probabilities reliably if we do not assume independence

Conditional Independence

Let $Z_1,\ldots Z_n$ and W be random variables on a given event space.

$Z_1,\ldots,Z_n$ are mutually independent given W if

$$P(Z_1, Z_2,\ldots Z_n|W) = \prod_{i=1}^{n} P(Z_i|W)$$

Note that these represent sets of equations, for all possible value assignments to random variables

Implications of Independence

- Suppose we have 5 Binary attributes and a binary class label
- Without independence, in order to specify the joint distribution, we need to specify a probability for each possible assignment of values to each variable resulting in a table of size 2^6=64
- Suppose the features are independent given the class label – we only need $5(2x2)$=20 entries
- The reduction in the number of probabilities to be estimated is even more striking when N, the number of attributes is large – from O(2^N) to O(N)

Naive Bayes Classifier

Consider a discrete valued target function $f : \chi \rightarrow \Omega$

where an instance $X = (X_1, X_2 \ldots X_n) \in \chi$ is described

in termsof attribute values $X_1 = x_1, \ X_2 = x_2, \ \ldots X_n = x_n$

where $x_i \in Domain(X_i)$

$$\omega_{MAP} = \arg\max_{\omega_j \in \Omega} P(\omega_j \mid X_1 = x_1, X_2 = x_2 \ldots X_n = x_n)$$

$$= \arg\max_{\omega_j \in \Omega} \frac{P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n \mid \omega_j) P(\omega_j)}{P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)}$$

$$= \arg\max_{\omega_j \in \Omega} P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n \mid \omega_j) P(\omega_j)$$

ω_{MAP} *is called the maximum a posteriori classification*

Naive Bayes Classifier

$$\omega_{MAP} = \arg\max_{\omega_j \in \Omega} P(\omega_j \mid X_1 = x_1, X_2 = x_2 \ldots X_n = x_n)$$

$$= \arg\max_{\omega_j \in \Omega} P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n \mid \omega_j) P(\omega_j)$$

If the attributes are *independent* given the class, we have

$$\omega_{MAP} = \arg\max_{\omega_j \in \Omega} \prod_{i=1}^{n} P\left(X_i = x_i \mid \omega_j\right) P(\omega_j)$$

$$= \omega_{NB}$$

$$= \arg\max_{\omega_j \in \Omega} P(\omega_j) \prod_{i=1}^{n} P\left(X_i = x_i \mid \omega_j\right)$$

Naive Bayes Learner

For each possible value ω_j of Ω,

$$\hat{P}(\Omega = \omega_j) \leftarrow Estimate(P(\Omega = \omega_j), D)$$

For each possible value a_{i_k} of X_i

$$\hat{P}(X_i = a_{i_k} \mid \omega_j) \leftarrow Estimate\left(P(X_i = a_{i_k} \mid \Omega = \omega_j), D\right)$$

Classify a new instance $X = (x_1, x_2, \ldots x_N)$

$$c(X) = \arg\max_{\omega_j \in \Omega} P(\omega_j) \prod_{i=1}^{n} P(X_i = x_i \mid \omega_j)$$

Estimate is a procedure for estimating the relevant probabilities from set of training examples

Learning Dating Preferences

Instances –

ordered 3-tuples of attribute values corresponding to

Height (tall, short)
Hair (dark, blonde, red)
Eye (blue, brown)

Classes –

+, –

Training Data

Instance	Class label
I_1 (t, d, l)	+
I_2 (s, d, l)	+
I_3 (t, b, l)	-
I_4 (t, r, l)	-
I_5 (s, b, l)	-
I_6 (t, b, w)	+
I_7 (t, d, w)	+
I_8 (s, b, w)	+

Probabilities to estimate

$P(+) = 5/8$

$P(-) = 3/8$

$P(Height \mid c)$	t	s
+	3/5	2/5
–	2/3	1/3

$P(Hair \mid c)$	d	b	r
+	3/5	2/5	0
–	0	2/3	1/3

$P(Eye \mid c)$	l	w
+	2/5	3/5
–	1	0

Classify (Height=t, Hair=b, eye=l)

$P(X \mid +) = (3/5)(2/5)(2/5) = (12/125)$

$P(X \mid -) = (2/3)(2/3)(1) = (4/9)$

$P(+|X) \alpha P(+)P(X|+)=(5/8)(12/125)=0.06$

$P(-|X) \alpha P(-)P(X|-)=(3/8)(4/9)=0.1667$

Classify (Height=t, Hair=r, eye=w)

Note the problem with zero probabilities

Solution – Use Laplacian correction

Estimation of Probabilities from Small Samples

$$\hat{P}(X_i = a_{i_k} \mid \omega_j) \leftarrow \frac{n_{ji_k} + mp_{ji}}{n_j + m}$$

n_j is the number of training examples of class ω_j

n_{ji_k} = number of training examples of class ω_j which have attribute value a_{i_k} for attribute X_i

p_{ji} is the prior estimate for $\hat{P}(X_i = a_{i_k} \mid \omega_j)$

m is the weight given to the prior

$$\text{As } n \rightarrow \infty, \hat{P}(X_i = a_{i_k} \mid \omega_j) \rightarrow \frac{n_{ji_k}}{n_j}$$

This is effectively the same as using Dirichlet priors

Sample Applications of Naïve Bayes Classifier

Naive Bayes is among the most useful algorithms

- Learning dating preferences
- Learn which news articles are of interest
- Learn to classify web pages by topic
- Learn to classify SPAM
- Learn to assign proteins to functional families

What attributes shall we use to represent text?

Learning to Classify Text

- Target function *Interesting*: Documents → {+,-}
- Learning: Use training examples to estimate

  $P(+)$, $P(-)$, $P(d \mid +)$, $P(d \mid -)$

Alternative generative models for documents:

- Represent each document as a sequence of words
  - In the most general case, we need a probability for each word occurrence in each position in the document, for each possible document length

$$P(d \mid \omega_i) = P(length(d)) \prod_{i=1}^{length(d)} P(X_i \mid \omega_i, length(d))$$

This would require estimating for each document,

$$|Vocabulary|^{length(d)} \times |\Omega|$$

probabilities for each possible document length!

To simplify matters, assume that probability of encountering a specific word in a particular position is independent of the position, and of document length

Treat each document as a bag of words!

Bag of Words Representation

So we estimate one position-independent class-conditional probability $P(w_k \mid \omega_j)$ for each word instead of the set of position-specific word occurrence probabilities $P(X_1 = w_k \mid \omega_j) \ldots P(X_{length(d)} = w_k \mid \omega_j)$

The number of probabilities to be estimated drops to

$$|\text{Vocabulary}| \times |\Omega|$$

The result is a generative model for documents that treats each document as an ordered tuple of word frequencies

More sophisticated models can consider dependencies between adjacent word positions

Learning to Classify Text

With the bag of words representation, we have

$P(d \mid \omega_j)$ is proportional to $\left\{ \frac{\left(\sum_k n_{kd} \right)!}{\prod_k n_{kd}!} \right\} \prod_k \left(P(w_k \mid \omega_j) \right)^{n_{kd}}$

where n_{kd} is the number of occurences of w_k in document d

(ignoring dependence on length of the document)

We can estimate $P(w_k \mid \omega_j)$ from the labeled bags of words we have.

Naïve Bayes Text Classifier

- Given 1000 training documents from each group, learn to classify new documents according to the newsgroup where it belongs
- Naive Bayes achieves 89% classification accuracy

comp.graphics
comp.os.ms-windows.misc
comp.sys.ibm.pc.hardware
comp.sys.mac.hardware comp.windows.x

alt.atheism
soc.religion.christian
talk.religion.misc
talk.politics.mideast
talk.politics.misc
talk.politics.guns

misc.forsale
rec.autos
rec.motorcycles
rec.sport.baseball
rec.sport.hockey

sci.space
sci.crypt
sci.electronics
sci.med

Naïve Bayes Text Classifier

Representative article from rec.sport.hockey

Path: cantaloupe.srv.cs.cmu.edu!das-news.harvard.edu!ogicse!uwm.edu
From: xxx@yyy.zzz.edu (John Doe)
Subject: Re: This year's biggest and worst (opinion)...
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 Learner – Summary

- Produces minimum error classifier if attributes are conditionally independent given the class

When to use

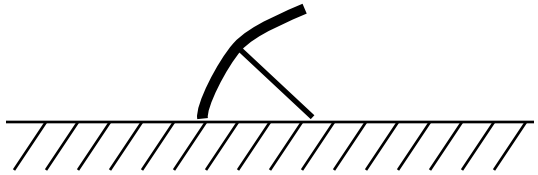
- Attributes that describe instances are likely to be conditionally independent given classification
- There is not enough data to estimate all the probabilities reliably if we do not assume independence
- Often works well even if when independence assumption is violated (Domigos and Pazzani, 1996)
- Can be used iteratively – Kang et al., 2006

On Estimation

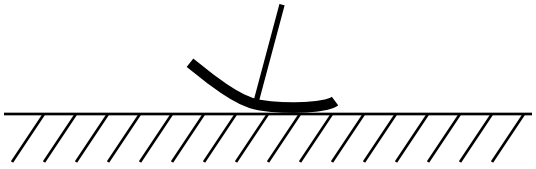
Estimating probabilities from data (discrete case)

- Maximum likelihood estimation
- Bayesian estimation
- Maximum a posteriori estimation

Example: Binomial Experiment



Head



Tail

- When tossed, the thumbtack can land in one of two positions: *Head* or *Tail*
- We denote by θ the (unknown) probability $P(H)$.
- Estimation task—
- Given a sequence of toss samples $x[1], x[2], \ldots, x[M]$ we want to estimate the probabilities $P(H) = \theta$ and $P(T) = 1 - \theta$

Statistical parameter fitting

Consider samples $x[1], x[2], \ldots, x[M]$ such that

- The set of values that X can take is known
- Each is sampled from the same distribution
- Each is sampled independently of the rest

} i.i.d. samples

The task is to find a parameter Θ so that the data can be summarized by a probability $P(x[j] | \Theta)$.

- The parameters depend on the given family of probability distributions: multinomial, Gaussian, Poisson, etc.
- We will focus first on binomial and then on multinomial distributions
- The main ideas generalize to other distribution families

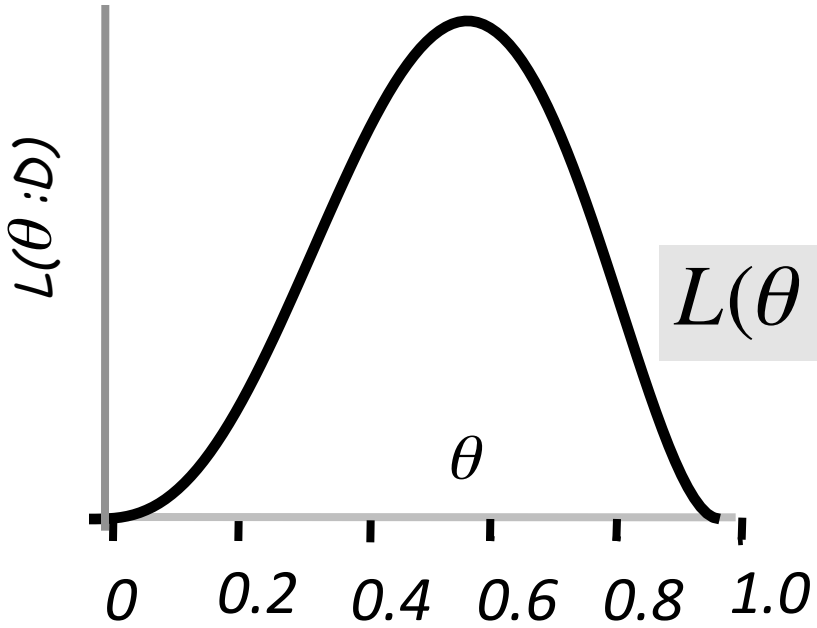
The Likelihood Function

How good is a particular θ?

It depends on how likely it is to generate the observed data

$$L(\theta : D) = P(D \mid \theta) \propto \prod_m P(x[m] \mid \theta)$$

The likelihood for the sequence H, T, T, H, H is



$$L(\theta : D) \propto \theta \cdot (1-\theta) \cdot (1-\theta) \cdot \theta \cdot \theta$$

Likelihood function

- The likelihood function $L(\theta : D)$ provides a measure of relative preferences for various values of the parameter θ given a collection of observations D drawn from a distribution that is parameterized by fixed but unknown θ.
- $L(\theta : D)$ is the probability of the observed data D viewed as a function of θ.
- Suppose data D is 5 heads out of 8 tosses. What is the likelihood function assuming that the observations were generated by a binomial distribution with an unknown but fixed parameter θ?

$$\binom{8}{5}\theta^5(1-\theta)^3$$

Sufficient Statistics

- To compute the likelihood in the thumbtack example we only require N_H and N_T (the number of heads and the number of tails)

$$L(\theta : D) \propto \theta^{N_H} \cdot (1-\theta)^{N_T}$$

- N_H and N_T are sufficient statistics for the parameter θ that specifies the binomial distribution
- A statistic is simply a function of the data
- A sufficient statistic *s* for a parameter θ is a function that summarizes from the data D, the relevant information $s(D)$ needed to compute the likelihood $L(\theta : D)$.
- If s is a sufficient statistic for θ, and $s(D) = s(D')$,

  then $L(\theta : D) = L(\theta : D')$

Maximum Likelihood Estimation

- Main Idea: Learn parameters that maximize the likelihood function
- Maximum likelihood estimation is
  - Intuitively appealing
  - One of the most commonly used estimators in statistics
  - Assumes that the parameters to be estimated are fixed, but unknown

Example: MLE for Binomial Data

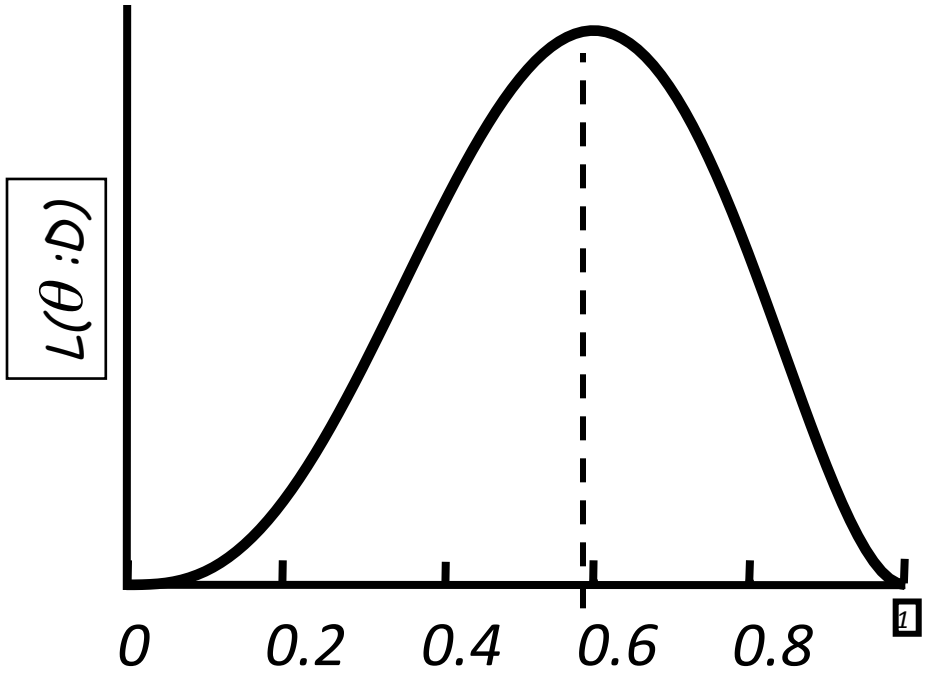
- Applying the MLE principle we get
- (Why?)

$$\hat{\theta} = \frac{N_H}{N_H + N_T}$$

Example:

$(N_H, N_T) = (3,2)$

ML estimate is 3/5 = 0.6



MLE for Binomial data

$$L(\theta : D) = \binom{N}{N_H} \theta^{N_H} \cdot (1-\theta)^{N_T}$$

$$\log L(\theta : D) = N_H \log\theta + N_T \log(1-\theta)$$

The likelihood is positive for all legitimate values of θ

So maximizing the likelihood is equivalent to maximizing its logarithm i.e. log likelihood

$$\frac{\partial}{\partial\theta} \log L(\theta : D) = 0 \text{ at extrema of } L(\theta : D)$$

$$\frac{\partial}{\partial\theta} \log L(\theta : D) = \frac{N_H}{\theta} + \frac{N_T(-1)}{(1-\theta)} = 0$$

$$(N_H + N_T)\theta = N_H$$

$$\theta_{ML} = \frac{N_H}{(N_H + N_T)}$$

Note that the likelihood is indeed maximized at $\theta = \theta_{ML}$ because in the neighborhood of θ_{ML}, the value of the likelihood is smaller than it is at $\theta = \theta_{ML}$

Maximum and curvature of likelihood around the maximum

- At the maximum, the derivative of the log likelihood is zero
- At the maximum, the second derivative is negative
- The curvature of the log likelihood is defined as

$$I(\theta) = -\frac{\partial}{\partial \theta^2} \log L(\theta : D)$$

- Large observed curvature $I(\theta_{ML})$ at $\theta = \theta_{ML}$ is associated with a sharp peak, intuitively indicating less uncertainty about the maximum likelihood estimate
- $I(\theta_{ML})$ is called the Fisher information

Maximum Likelihood Estimate

ML estimate can be shown to be

- Asymptotically unbiased

$$\lim_{N\to\infty} E\left(\theta_{ML}\right) = \theta_{True}$$

- Asymptotically consistent - converges to the true value as the number of examples approaches infinity

$$\lim_{N\to\infty} \Pr\left\{\left\|\theta_{ML} - \theta_{True}\right\| \le \varepsilon\right\} = 1$$

$$\lim_{N\to\infty} E\left(\left\|\theta_{ML} - \theta_{True}\right\|^2\right) = 0$$

- Asymptotically efficient – achieves the lowest variance that any estimate can achieve for a training set of a certain size (satisfies the Cramer-Rao bound)

Maximum Likelihood Estimate

- ML estimate can be shown to be representationally invariant – If θ_{ML} is an ML estimate of θ, and $g(\theta)$ is a function of θ, then $g(\theta_{ML})$ is an ML estimate of $g(\theta)$
- When the number of samples is large, the probability distribution of θ_{ML} *has Gaussian distribution with mean* θ_{True} (the actual value of the parameter) – a consequence of the central limit theorem
  - A random variable which is a sum of a large number of random variables has a Gaussian distribution – ML estimate is related to the sum of random variables
- We can use the likelihood ratio to reject the null hypothesis corresponding to $\theta = \theta_0$ as unsupported by data if the ratio of the likelihoods evaluated at θ_0 and at θ_{ML} is *small.* (The ratio can be calibrated when the likelihood function is approximately quadratic)

From Binomial to Multinomial

- Suppose a random variable X can take the values $1,2,\ldots,K$
- We want to learn the parameters $\theta_1, \theta_2, \ldots, \theta_K$
- Sufficient statistics: $N_1, N_2, \ldots, N_K$ - the number of times each outcome is observed
- Likelihood function

$$L(\theta : D) \propto \prod_{k=1}^{K} \theta_k^{N_k}$$

- ML estimate

$$\hat{\theta}_k = \frac{N_k}{\sum_{\ell} N_\ell}$$

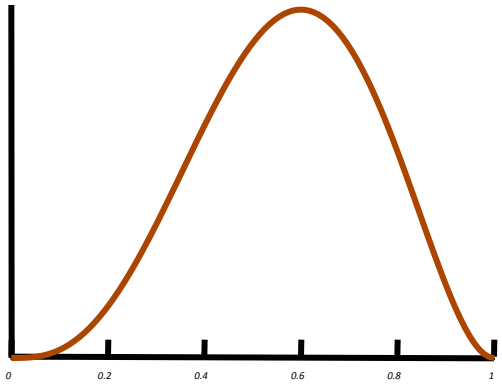
Summary of Maximum Likelihood estimation

- Define a likelihood function which is a measure of how likely it is that the observed data were generated from a probability distribution with a particular choice of parameters
- Select the parameters that maximize the likelihood
- In simple cases, ML estimate has a closed form solution
- In other cases, ML estimation may require numerical optimization

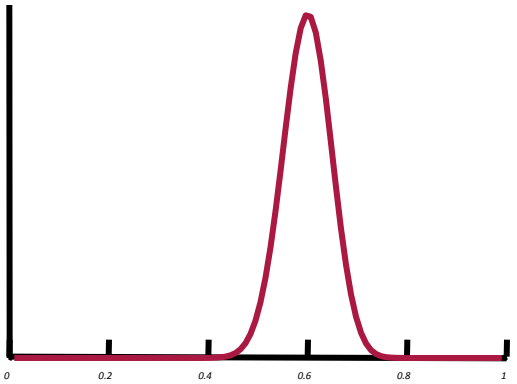
- Problem with ML estimate – assigns zero probability to unobserved values – can lead to difficulties when estimating from small samples
- Question – How would Naïve Bayes classifier behave if some of the class conditional probability estimates are zero?

Bayesian Estimation

- MLE commits to a specific value of the unknown parameter (s)
- MLE is the same in both cases shown



vs.



Of course, in general, one cannot summarize a function by a single number!

Intuitively, the confidence in the estimates should be different

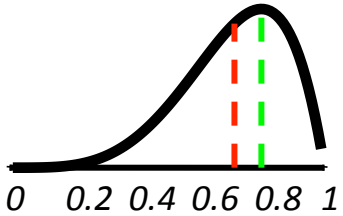
Bayesian Estimation

Maximum Likelihood approach is Frequentist at its core

- Assumes there is an unknown but fixed parameter θ
- Estimates θ with some confidence
- Prediction of probabilities using the estimated parameter value

Bayesian Approach

- Represents uncertainty about the unknown parameter
- Uses probability to quantify this uncertainty:
  - Unknown parameters as random variables
- Prediction follows from the rules of probability:
  - Expectation over the unknown parameters



Example: Binomial Data Revisited

- Suppose that we choose a uniform prior $p(\theta) = 1$ for θ in [0,1]
- $P(\theta \mid D)$ is proportional to the likelihood $L(\theta : D)$

$$p(\theta \mid D) = \frac{p(D \mid \theta) p(\theta)}{\int_0^1 p(D \mid \theta) p(\theta) d\theta}$$

$$\text{In this case, } p(D \mid \theta) = \binom{5}{1} \theta^4 (1-\theta)^1 \text{ and } \forall \theta \in [0,1], p(\theta) = \frac{1}{1-0} = 1$$

$$\int_0^1 p(D \mid \theta) p(\theta) = \binom{5}{1} \int_0^1 \left(\theta^4 - \theta^5\right) d\theta = \binom{5}{1} \left[\frac{\theta^5}{5} - \frac{\theta^6}{6}\right]_0^1 = \binom{5}{1} \frac{1}{30}$$

$$p(\theta \mid D) = 30\theta^4 (1-\theta)$$

$$P(X[m+1] = H \mid D) = \int_0^1 p(\theta \mid D) \theta d\theta = 30 \int_0^1 \theta^4 (1-\theta) \theta d\theta = 30 \left[\frac{\theta^6}{6} - \frac{\theta^7}{7}\right]_0^1 = \frac{5}{7} = 0.7142$$

Example: Binomial Data Revisited

(NH,NT) = (4,1)

MLE for P(X = H) is 4/5 = 0.8

Bayesian estimate is

$$P(x[M+1] = H \mid D) = \int \theta \cdot P(\theta \mid D) d\theta = \frac{5}{7} = 0.7142\ldots$$

In this example, MLE and Bayesian prediction differ

It can be proved that

- If the prior is well-behaved – i.e. does not assign 0 density to any feasible parameter value
  - Then both MLE and Bayesian estimate converge to the same value in the limit
- Both almost surely converge to the underlying distribution P(X)
- The ML and Bayesian approaches behave differently when the number of samples is small

All relative frequencies are not equi-probable

- In practice we might want to assert priors that allow us to express our beliefs regarding the parameter to be estimated
- For example, we might want a prior that assigns a higher probability to parameter values that describe a fair coin than it does to an unfair coin
- The beta distribution allows us to capture such prior beliefs

Beta distribution

Gamma Function:

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, dt$$

The integral converges if and only if $x > 0$.

If x is an integer that is greater than 0, it can be shown that

$$\Gamma(x) = (x-1)!$$

$$\frac{\Gamma(x+1)}{\Gamma(x)} = x$$

The beta density function with parameters $a,\ b, N = a + b,$ where a, b are real numbers $> 0,$ $beta(\theta; a, b)$ is:

$$p(\theta) = \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} \theta^{a-1}(1-\theta)^{b-1} \quad \text{where } 0 \le \theta \le 1$$

Beta distribution

If a, b are real numbers > 0, then

$$\int_0^1 \theta^a (1-\theta)^b \, d\theta = \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)}$$

If θ has distribution given by $beta(\theta; a, b)$, then $E(\theta) = \frac{a}{N}$.

Let $D = \{X[1], \ldots, X[M]\}$ be
a sequence of iid samples from a binomial distribution;
Let $N_H = s$; $N_T = t$; and $p(\theta) = beta(\theta; a, b)$
Then we can show that $p(\theta|D) = beta(\theta; a+s, b+t)$

Update of the parameter with a beta prior based on data yields a beta posterior

Conjugate Families

- The property that the posterior distribution follows the same parametric form as the prior distribution is called conjugacy
- Conjugate families are useful because:
  - For many distributions we can represent them with hyper parameters
  - They permit sequential update of the posterior based on data
  - In many cases we have closed-form solution for prediction
- Beta prior is a conjugate family for the binomial likelihood

Bayesian prediction

$$\text{prior}: beta(\theta; a, b)$$

$$\text{Data}: D = \{X[1].....X[M]\}$$

$$\text{posterior}: p(\theta \mid D) = beta(\theta; a + N_H, b + N_T)$$

$$\text{prediction}: \mathrm{P}(X[M+1] = H \mid D) = \frac{a + N_H}{N + M} = \frac{(a + N_H)}{(a + b) + (N_H + N_T)}$$

Dirichlet Priors

- Recall that the likelihood function is $L(\Theta : D) = \prod_{k=1}^{K} \theta_k^{N_k}$
- A Dirichlet prior with hyperparameters $\alpha_1,\ldots,\alpha_K$ is defined as

$$P(\Theta) = \frac{\Gamma(N)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1}; \qquad 0 \le \theta_k \le 1; \qquad \sum_{k=1}^{K} \theta_k = 1$$

where $\Theta = (\theta_1 \ldots \theta_K)$

- Then the posterior has the same form, with hyperparameters $\alpha_1+N_1,\ldots,\alpha_K+N_K$

$$P(\Theta \mid D) \propto P(\Theta) P(D \mid \Theta)$$

$$\propto \prod_{k=1}^{K} \theta_k^{\alpha_k - 1} \prod_{k=1}^{K} \theta_k^{N_k} = \prod_{k=1}^{K} \theta_k^{\alpha_k + N_k - 1}$$

Dirichlet Priors

- Dirichlet priors enable closed form prediction based on multinomial samples:
  - If $P(\Theta)$ is Dirichlet with hyperparameters $\alpha_1,\ldots,\alpha_K$ then

$$P(X[1]=k)=\int \theta_k \cdot P(\Theta)d\Theta=\frac{\alpha_k}{\sum_\ell \alpha_\ell}$$

- Since the posterior is also Dirichlet, we get

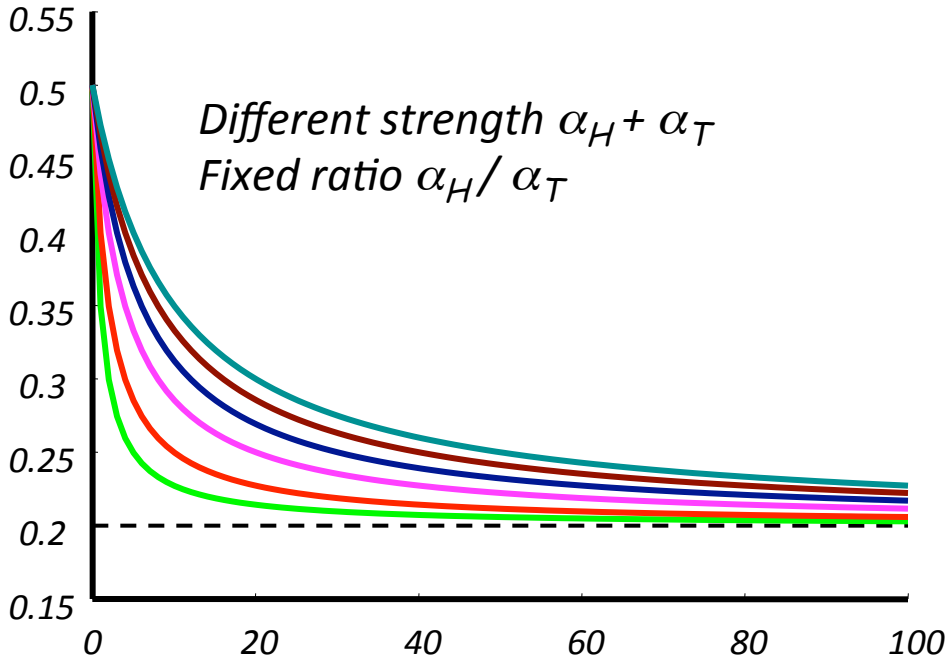
$$P(X[M+1]=k \mid D)=\int \theta_k \cdot P(\Theta \mid D)d\Theta=\frac{\alpha_k+N_k}{\sum_\ell (\alpha_\ell+N_\ell)}$$

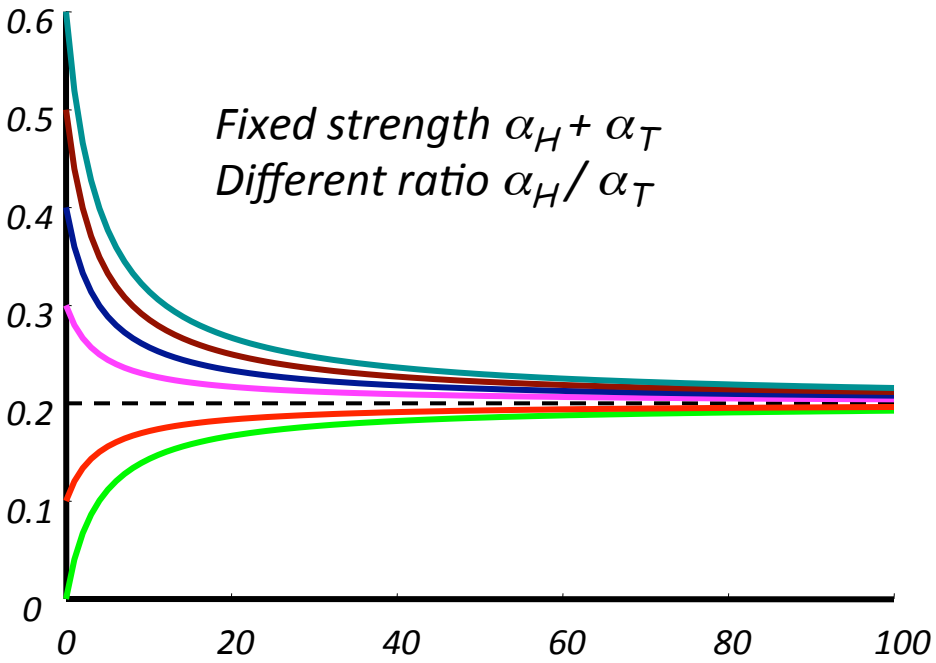
Intuition behind priors

- The hyperparameters $\alpha_1,\ldots,\alpha_K$ can be thought of as imaginary counts from our prior experience
- Equivalent sample size = $\alpha_1+\ldots+\alpha_K$
- The larger the equivalent sample size the more confident we are in our prior

Effect of Priors

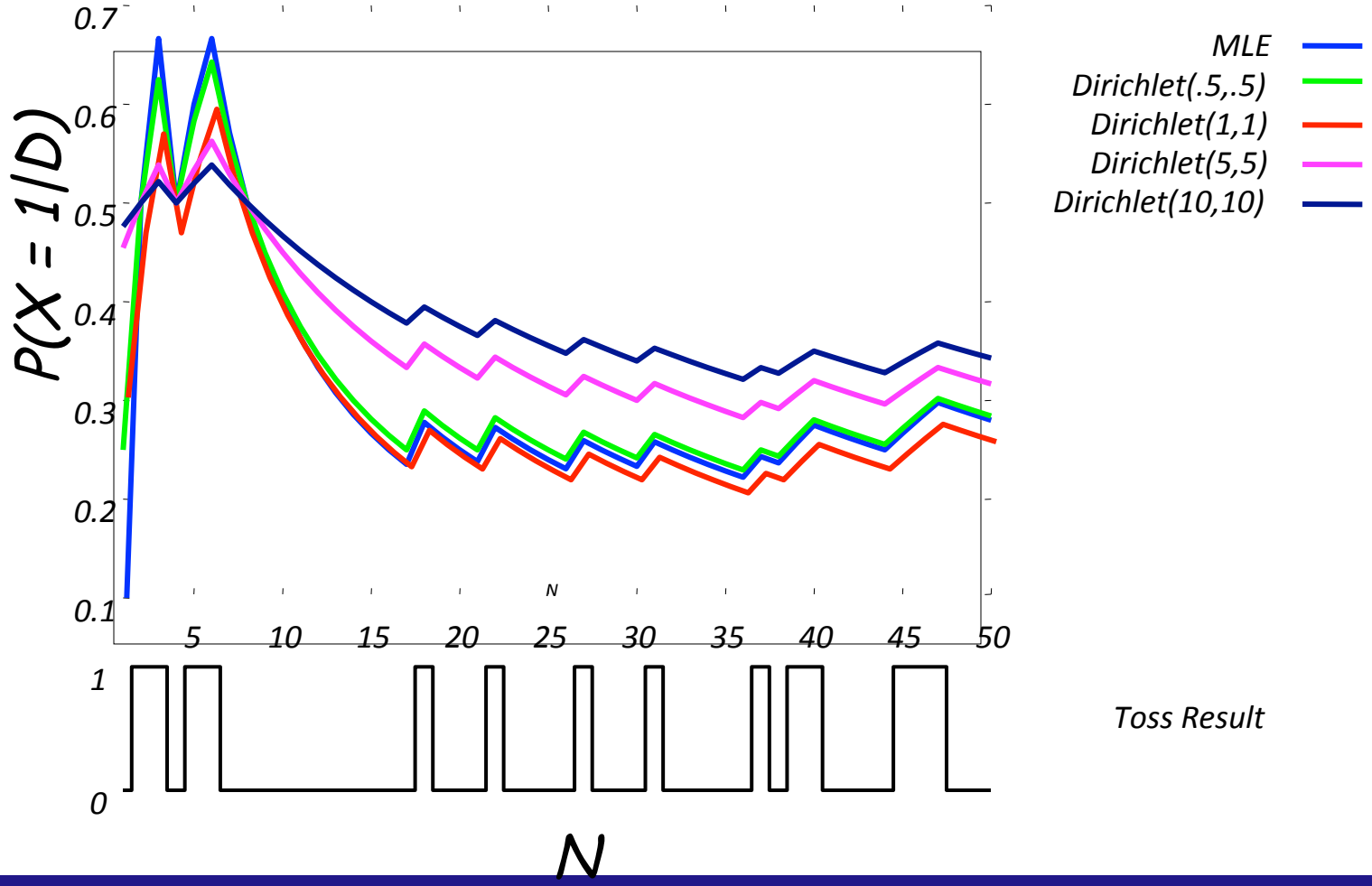
Prediction of $P(X=H)$ after seeing data with $N_H = 0.25 \bullet N_T$ for different sample sizes





Effect of Priors

- In real data, Bayesian estimates are less sensitive to noise in the data



Conjugate Families

- The property that the posterior distribution follows the same parametric form as the prior distribution is called conjugacy
  - Dirichlet prior is a conjugate family for the multinomial likelihood
- Conjugate families are useful because:
  - For many distributions we can represent them with hyperparameters
  - They allow for sequential update within the same representation
  - In many cases we have closed-form solution for prediction

Bayesian Estimation

$$P(x[M+1] \mid x[1],\ldots,x[M])$$
$$= \int P(x[M+1] \mid \theta, x[1],\ldots,x[M]) P(\theta \mid x[1],\ldots,x[M]) d\theta$$
$$= \int P(x[M+1] \mid \theta) P(\theta \mid x[1],\ldots,x[M]) d\theta$$

where

$$P(\theta \mid x[1],\ldots x[M]) = \frac{P(x[1],\ldots x[M] \mid \theta) P(\theta)}{P(x[1],\ldots x[M])}$$

Likelihood

Prior

Posterior

Probability of data

Summary of Bayesian estimation

- Treat the unknown parameters as random variables
- Assume a prior distribution for the unknown parameters
- Update the distribution of the parameters based on data – easy if we have conjugate priors
- Use Bayes rule to make prediction

Maximum a posteriori (MAP) estimates – A compromise between ML and Bayesian approaches

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

$$\begin{aligned}\Theta_{MAP} &= \arg\max_{\Theta} P(\Theta|D) \\ &= \arg\max_{\Theta} P(D|\Theta)P(\Theta) \\ &= \arg\max_{\Theta} P(\Theta)L(\Theta : D)\end{aligned}$$

Maximum a posteriori (MAP) estimates – A compromise between ML and Bayesian approaches

$$\Theta_{MAP} = \arg\max_{\Theta} P(\Theta) L(\Theta : D)$$

- Like in Bayesian estimation, we treat the unknown parameters as random variables
- But we estimate a single value for the parameter
  - the maximum a posteriori estimate that corresponds to the most probable value of the parameter
  - given the data <u>for a given choice of the prior</u>

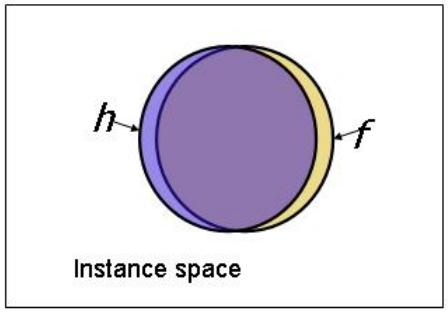
End of extra slides on estimation

Evaluating Classifier Performance

Vasant Honavar
Artificial Intelligence Research Laboratory
College of Information Sciences and Technology
Bioinformatics and Genomics Graduate Program
The Huck Institutes of the Life Sciences
Pennsylvania State University

vhonavar@ist.psu.edu
vhonavar.ist.psu.edu
faculty.ist.psu.edu/vhonavar

Estimating classifier performance



$h(x) = f(x)$

$$Domain(X) = \{a, b, c, d\}$$

$$D(X) = \left\{\frac{1}{8}, \frac{1}{2}, \frac{1}{8}, \frac{1}{4}\right\}$$

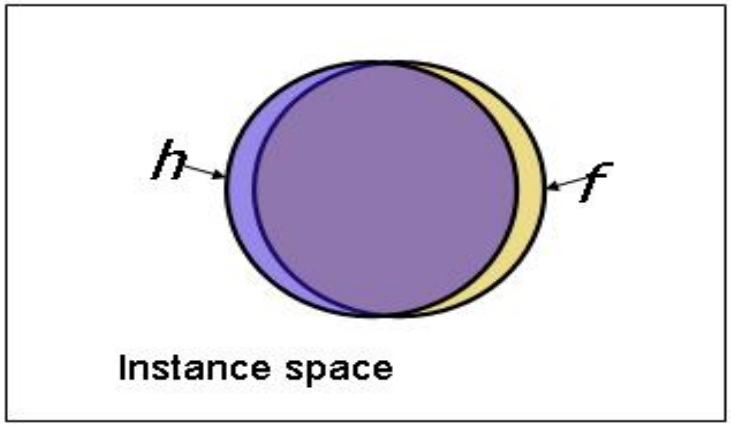
x	a	b	c	d
$h(x)$	0	1	1	0
$f(x)$	1	1	0	0

$$error_D(h) = \Pr_D[h(x) \neq f(x)]$$

$$= D(X = a) + D(X = c)$$

$$= \frac{1}{8} + \frac{1}{8} = \frac{1}{4}$$

Measuring classifier performance



$h(x) = f(x)$

$$Error_D(h) = \Pr_{x \in D}\left(f(x) \neq h(x)\right)$$

- We do not in general, know D, the distribution from which the data samples are drawn.
- So we estimate the error from the samples we have

Estimating Classifier Performance

N: Total number of instances in the data set

TP_j: Number of True positives for class j

FP_j : Number of False positives for class j

TN_j: Number of True Negatives for class j

FN_j: Number of False Negatives for class j

$$Accuracy_j = \frac{TP_j + TN_j}{N}$$
$$= P\left(class = c_j \wedge label = c_j\right)$$

Perfect classifier ←→ Accuracy =1

Popular measure

Biased in favor of the majority class!

Should be used with caution!

Measuring Classifier Performance: Sensitivity

$$Sensitivity_j = \frac{TP_j}{TP_j + FN_j}$$
$$= \frac{Count(label = c_j \wedge class = c_j)}{Count(class = c_j)}$$
$$= P(label = c_j \mid class = c_j)$$

Perfect classifier → Sensitivity = 1

Probability of correctly labeling members of the target class

Also called recall or hit rate

Measuring Classifier Performance: Specificity

$$Specificity_j = \frac{TP_j}{TP_j + FP_j}$$
$$= \frac{Count\left(label = c_j \wedge class = c_j\right)}{Count\left(label = c_j\right)}$$
$$= P\left(class = c_j \mid label = c_j\right)$$

Perfect classifier → Specificity = 1

Also called precision

Probability that a positive prediction is correct

Measuring Performance: Precision, Recall, and False Alarm Rate

$$Precision_j = Specificity_j = \frac{TP_j}{TP_j + FP_j}$$

$$Recall_j = Sensitivity_j = \frac{TP_j}{TP_j + FN_j}$$

Perfect classifier → Precision=1

Perfect classifier → Recall=1

$$FalseAlarm_j = \frac{FP_j}{TN_j + FP_j}$$

$$= \frac{Count\left(label = c_j \wedge class = \neg c_j\right)}{Count\left(label = \neg c_j\right)}$$

$$= P\left(label = c_j \mid class = \neg c_j\right)$$

Perfect classifier →
False Alarm Rate = 0

Classifier Learning -- Measuring Performance

Class → / Label ↓	C_1	$\neg C_1$
C_1	TP= 55	FP=5
$\neg C_1$	FN=10	TN=30

$$N = TP + FN + TN + FP = 100$$

$$sensitivity_1 = \frac{TP}{TP + FN} = \frac{55}{55 + 10} = \frac{55}{65}$$

$$specificity_1 = \frac{TP}{TP + FP} = \frac{55}{55 + 5} = \frac{55}{60}$$

$$accuracy_1 = \frac{TP + TN}{N} = \frac{55 + 30}{100} = \frac{85}{100}$$

$$falsealarm_1 = \frac{FP}{TN + FP} = \frac{5}{30 + 5} = \frac{5}{35}$$

Measuring Performance – Correlation Coefficient

$$CC_j = \frac{(TP_j \times TN_j) - (FP_j \times FN_j)}{\sqrt{(TP_J + FN_j)(TP_j + FP_j)(TN_j + FP_j)(TN_j + FN_j)}}$$

$$-1 \le CC_j \le 1$$

Perfect classifier ←→ $CC = 1$, Random guessing ←→ CC=0

Corresponds to the standard measure of correlation between two random variables *Label* and *Class* estimated from labels $\mathbf{L}$ and the corresponding class values $\mathbf{C}$ for the special case of binary (0/1) valued labels and classes

$$CC_j = \sum_{d_i \in D} \frac{(jlabel_i - \overline{jlabel})(jclass_i - \overline{jclass})}{\sigma_{JLABEL}\sigma_{JCLASS}}$$

$where\ jlabel_i = 1$ iff the classifier assigns d_i to class c_j

$jclass_i = 1$ iff the true class of d_i is class c_j

Beware of terminological confusion in the literature!

- Some bioinformatics authors use "accuracy" incorrectly to refer to recall i.e. sensitivity or precision i.e. specificity
- In medical statistics, specificity sometimes refers to sensitivity for the negative class i.e.

$$\frac{TN_j}{TN_j + FP_j}$$

- Some authors use false alarm rate to refer to the probability that a positive prediction is incorrect i.e.

$$\frac{FP_j}{FP_j + TP_j} = 1 - Precision_j$$

When you write

- provide the formula in terms of TP, TN, FP, FN

When you read

- check the formula in terms of TP, TN, FP, FN

Measuring Classifier Performance

- TP, FP, TN, FN provide the relevant information
- No single measure tells the whole story
- A classifier with 98% accuracy can be useless if 98% of the population does not have cancer and the 2% that do are misclassified by the classifier
- Use of multiple measures recommended
- Beware of terminological confusion!

Micro-averaged performance measures
Performance on a random instance

- Micro averaging gives equal importance to each instance
- Classes with large number of instances dominate

$$MicroAverage\ Precision = \frac{\sum_j TP_j}{\sum_j TP_j + \sum_j FP_j}$$

$$MicroAverage\ Recall = \frac{\sum_j TP_j}{\sum_j TP_j + \sum_j FN_j}$$

$$MicroAverage\ FalseAlarm = 1 - MicroAverage\ Precision$$

$$MicroAverage\ Accuracy = \frac{\sum_j TP_j}{N}$$

Etc.

$$MicroAverage\ CC = \frac{\left(\left(\sum_j TP_j\right) \times \left(\sum_j TN_j\right)\right) - \left(\left(\sum_j FP_j\right) \times \left(\sum_j FN_j\right)\right)}{\sqrt{\left(\sum_j TP_j + \sum_j FN_j\right)\left(\sum_j TP_j + \sum_j FP_j\right)\left(\sum_j TN_j + \sum_j FP_j\right)\left(\sum_j TN_j + \sum_j FN_j\right)}}$$

Macro-averaged performance measures

Macro averaging gives equal importance to each of the M classes

$$MacroAverage\ Sensitivity = \frac{1}{M}\sum_j Sensitivity_j$$

$$MacroAverageCorrelationCoeff = \frac{1}{M}\sum_j CorrelationCoeff_j$$

$$MacroAverage\ Specificity = \frac{1}{M}\sum_j Specificity_j$$

Receiver Operating Characteristic (ROC) Curve

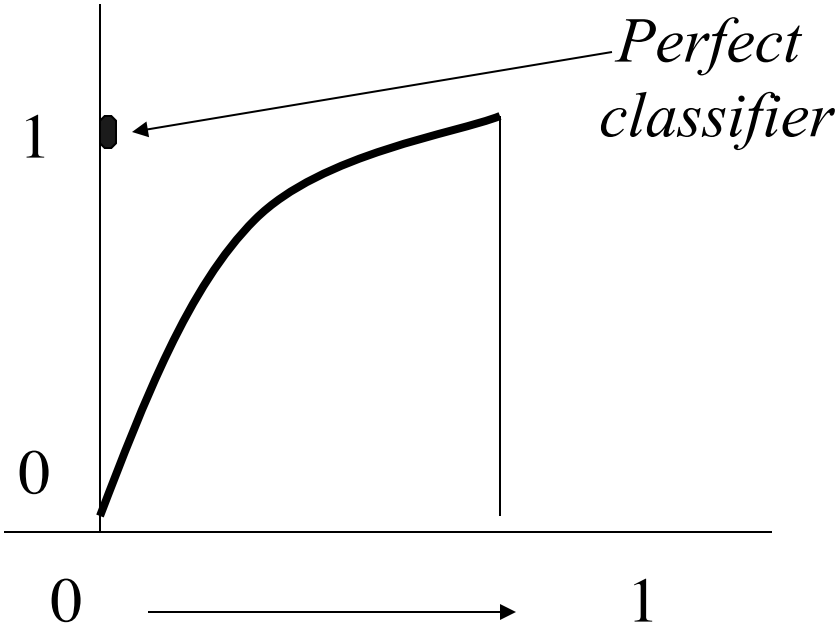
- We can often trade off recall versus precision – e.g., by adjusting classification threshold θ e.g.,

$$label = c_j \text{ if } \frac{P(c_j \mid X)}{P(\neg c_j \mid X)} > \theta$$

- ROC curve is a plot of Sensitivity against False Alarm Rate which characterizes this tradeoff for a given classifier

Receiver Operating Characteristic (ROC) Curve

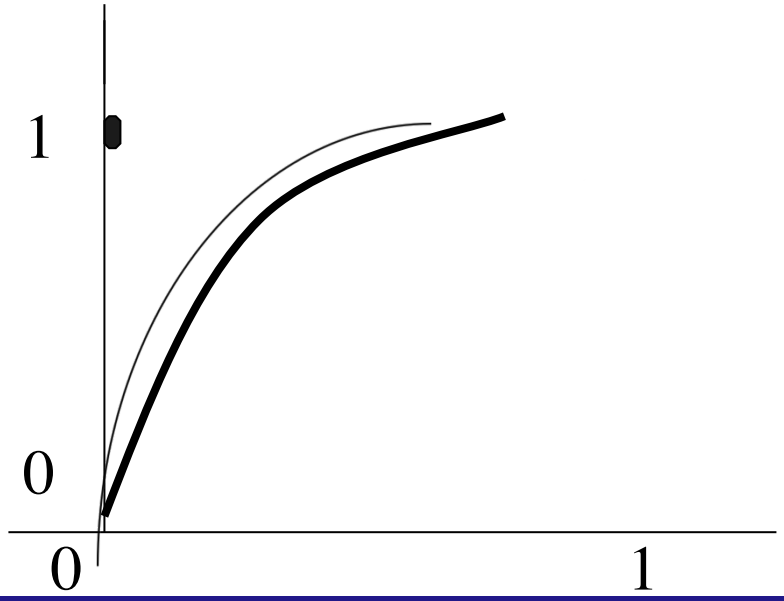
$$Sensitivity = \frac{TP}{TP + FN}$$



$$False\ Alarm\ Rate = \frac{FP}{FP + TN}$$

Measuring Performance of Classifiers – ROC curves

- ROC curves offer a more complete picture of the performance of the classifier as a function of the classification threshold
- A classifier *h* is better than another classifier *g* if ROC(*h*) dominates the ROC(*g*)
- ROC(*h*) dominates ROC(*g*) → AreaROC(*h*) > AreaROC(*g*)



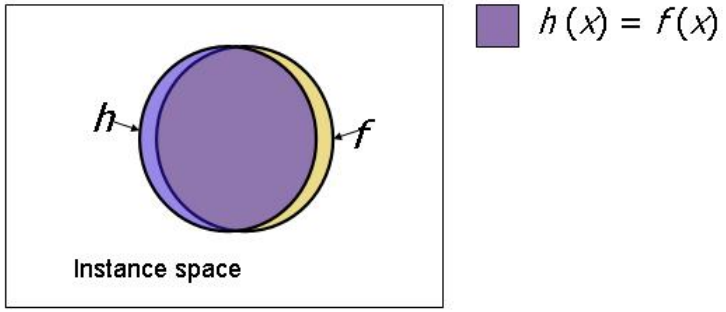
Evaluating a Classifier

- How well can a classifier be expected to perform on *novel* data?
- We can *estimate* the *performance* (e.g., accuracy, sensitivity) of the classifier using a test data set (not used for training)
- How close is the *estimated* performance to the *true* performance?

References:

- Evaluation of discrete valued hypotheses – Chapter 5, Mitchell
- Empirical Methods for Artificial Intelligence, Cohen

Estimating the performance of a classifier



The *true* error of a hypothesis *h* with respect to a target function *f* and an instance distribution *D* is

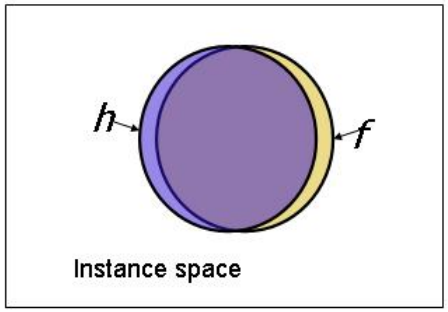
$$Error_D(h) \equiv \Pr_{x \in D}\left[f(x) \neq h(x)\right]$$

The sample error of a hypothesis *h* with respect to a target function *f* and an instance distribution *D* is

$$Error_S(h) \equiv \frac{1}{|S|}\sum_{x \in S} \delta(f(x) \neq h(x))$$

$$\delta(a,b) = 1 \text{ iff } a \neq b; \delta(a,b) = 0 \text{ otherwise}$$

Estimating classifier performance



$h(x) = f(x)$

$$Domain(X) = \{a, b, c, d\}$$

$$D(X) = \left\{\frac{1}{8}, \frac{1}{2}, \frac{1}{8}, \frac{1}{4}\right\}$$

x	a	b	c	d
$h(x)$	0	1	1	0
$f(x)$	1	1	0	0

$$error_D(h) = \Pr_D[h(x) \neq f(x)]$$

$$= D(X = a) + D(X = c)$$

$$= \frac{1}{8} + \frac{1}{8} = \frac{1}{4}$$

Evaluating the performance of a classifier

- Sample error estimated from training data is an *optimistic* estimate $$Bias = E\left[Error_S(h)\right] - Error_D(h)$$

- For an *unbiased* estimate, *h* must be evaluated on an independent sample *S* (which is not the case if *S* is the training set!)
- Even when the estimate is unbiased, it can *vary* across samples!
- If *h* misclassifies 8 out of 100 samples $$Error_S(h) = \frac{8}{100} = 0.08$$

How close is the *sample error* to the *true error*?

How close is the *estimated* error to the *true* error?

- Choose a sample *S* of size *n* according to distribution *D*
- Measure

$$Error_S(h)$$

$Error_S(h)$ is a random variable (outcome of a random experiment)

Given $Error_S(h)$, what can we conclude about $Error_D(h)$?

More generally, given the estimated performance of a hypothesis, what can we say about its actual performance?

Evaluation of a classifier with limited data

- There is extensive literature on how to estimate classifier performance from samples and how to assign confidence to estimates (See Mitchell, Chapter 5)
- Holdout method – use part of the data for training, and the rest for testing
- We may be unlucky – training data or test data may not be *representative*
- Solution – Run multiple experiments with disjoint training and test data sets in which each class is represented in roughly the same proportion as in the entire data set

Estimating the performance of the learned classifier

K-fold cross-validation

Partition the data (multi) set *S* into *K* equal parts $S_1 .. S_K$ with roughly the same class distribution as *S*.

Errorc = 0

For *i*=1 to *K* do

$$\{ \quad S_{Test} \leftarrow S_i \;\; S_{Train} \leftarrow S - S_i;$$
$$\alpha \leftarrow Learn(S_{Train})$$
$$Errorc \leftarrow Errorc + Error(\alpha, S_{Test}) \quad \}$$

$$Error \leftarrow \left(\frac{Errorc}{K} \right); \quad Output(Error)$$

Leave-one-out cross-validation

- *K*-fold cross validation with *K* = *n* where *n* is the total number of samples available
- *n* experiments – using *n*-1 samples for training and the remaining sample for testing
- Leave-one-out cross-validation does not guarantee the same class distribution in training and test data!

  Extreme case: 50% class 1, 50% class 2

  Predict majority class label in the training data

  True error – 50%;

  Leave-one-out error estimate – 100%!!!!!

Estimating classifier performance

Recommended procedure

- Use *K*-fold cross-validation (*K*=5 or 10) for estimating performance estimates (accuracy, precision, recall, points on ROC curve, etc.) and 95% confidence intervals around the mean
- Compute mean values of performance estimates and standard deviations of performance estimates
- Report mean values of performance estimates and their standard deviations or 95% confidence intervals around the mean
- Be skeptical – repeat experiments several times with different random splits of data into *K* folds!

Evaluating a hypothesis or a learning algorithm

How well can the decision tree be expected to perform on *novel* data?

We can *estimate* the *performance* (e.g., accuracy) of the decision tree using a test data set (not used for training)

How close is the *estimated* performance to the *true* performance?

Reference: Evaluation of discrete valued hypotheses – Chapter 5, Mitchell

Evaluating performance when we can afford to test on a large independent test set

The *true* error of a hypothesis *h* with respect to a target function *f* and an instance distribution *D* is

$$Error_D(h) \equiv \Pr_{x \in D}\left[f(x) \neq h(x)\right]$$

The sample error of a hypothesis h with respect to a target function f and an instance distribution D is

$$Error_S(h) \equiv \frac{1}{|S|}\sum_{x \in S} \delta(f(x) \neq h(x))$$

$$\delta(a,b) = 1 \text{ iff } a \neq b; \delta(a,b) = 0 \text{ otherwise}$$

Evaluating Classifier performance

$$Bias = E\left[Error_S(h)\right] - Error_D(h)$$

Sample error estimated from training data is an *optimistic* estimate

For an *unbiased* estimate, h must be evaluated on an independent sample S (which is not the case if S is the training set!)

Even when the estimate is unbiased, it can *vary* across samples!

If h misclassifies 8 out of 100 samples

$$Error_S(h) = \frac{8}{100} = 0.08$$

How close is the *sample error* to the *true error*?

How close is estimated error to its true value?

Choose a sample *S* of size *n* according to distribution *D*

Measure $Error_S(h)$

$Error_S(h)$ is a random variable (outcome of a random experiment)

Given $Error_S(h)$, what can we conclude about $Error_D(h)$?

More generally, given the estimated performance of a hypothesis, what can we say about its actual performance?

How close is estimated accuracy to its true value?

Question: How close is *p* (the true probability) to $\hat{p}$?

This problem is an instance of a well-studied problem in statistics – the problem of estimating the proportion of a population that exhibits some property, given the observed proportion over a random sample of the population. In our case, the property of interest is that *h* correctly (or incorrectly) classifies a sample.

Testing *h* on a single random sample *x* drawn according to *D* amounts to performing a random experiment which succeeds if *h* correctly classifies *x* and fails otherwise.

How close is estimated accuracy to its true value?

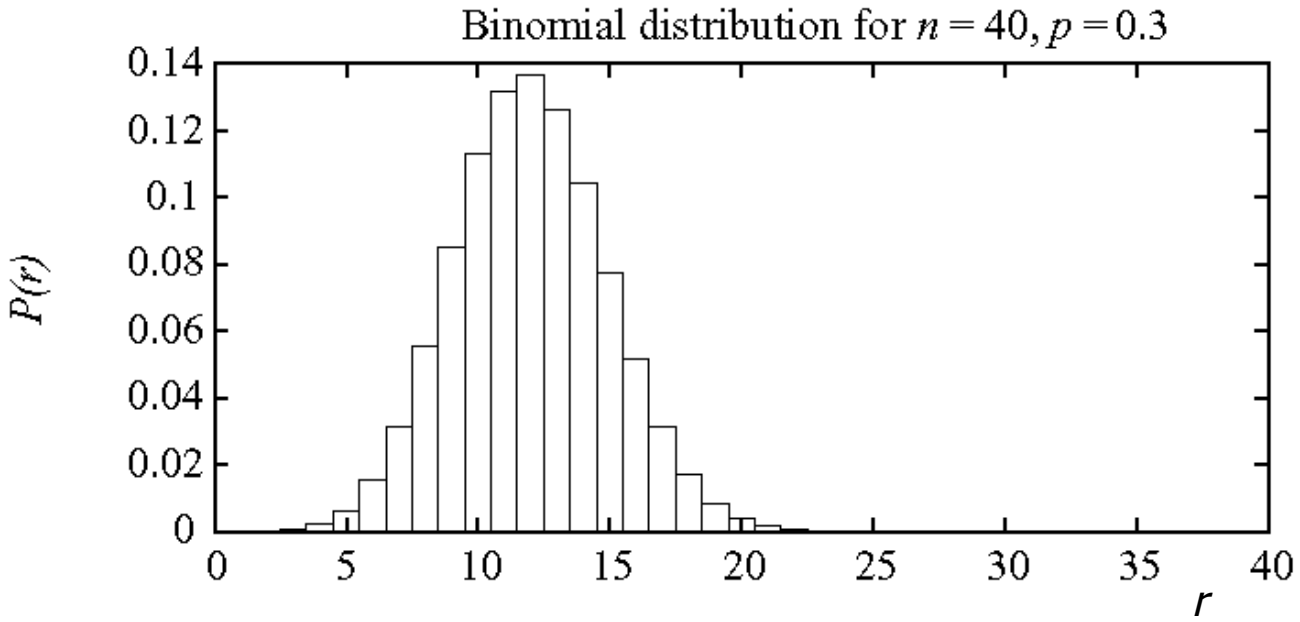
The output of a hypothesis whose true error is *p* as a binary *random variable* which corresponds to the outcome of a Bernoulli trial with a *success rate p* (the probability of correct prediction)

The *number* of *successes r* observed in *N* trials is a random variable Y which follows the Binomial distribution

$$P(r) = \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r}$$

$Error_S(h)$ is a Random Variable

Probability of observing r misclassified examples in a sample of size n:



$$P(r) = \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r}$$

Recall basic statistics

Consider a random experiment with discrete valued outcomes $y_1, y_2, \cdots y_M$

The expected value of the corresponding random variable *Y* is

$$E(Y) \equiv \sum_{i=1}^{M} y_i \Pr(Y = y_i)$$

The variance of *Y is*

$$Var(Y) \equiv E\left[(Y - E[Y])^2\right]$$

The standard deviation of *Y* is

$$\sigma_Y \equiv \sqrt{Var(Y)}$$

How close is estimated accuracy to its true value?

The *mean* of a Bernoulli trial with success rate $p = p$

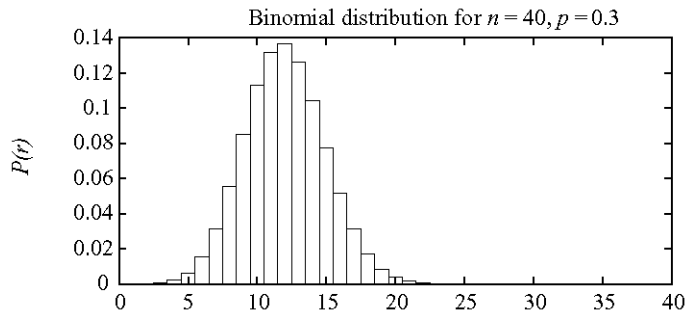
Variance = p (*1-p*)

If *N trials* are taken from the same Bernoulli process, the observed success rate $\hat{p}$ has the same mean p

and variance $\frac{p(1-p)}{N}$

For large *N*, the distribution of $\hat{p}$ follows a Gaussian distribution

Binomial Probability Distribution



$$P(r) = \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r}$$

Probability *P(r)* of *r* heads in *n* coin flips, if *p* = *Pr(heads)*

- Expected, or mean value of *X*, *E*[*X*], is

$$E[X] \equiv \sum_{i=0}^{N} iP(i) = np$$

- Variance of *X* is

$$Var(X) \equiv E[(X - E[X])^2] = np(1-p)$$

- Standard deviation of *X*, σ_X, is

$$\sigma_X \equiv \sqrt{E[(X - E[X])^2]} = \sqrt{np(1-p)}$$

Estimators, Bias, Variance, Confidence Interval

$$Error_S(h) = \frac{r}{n}$$

$$Error_D(h) = p$$

$$\sigma_{Error_S(h)} = \sqrt{\frac{p(1-p)}{n}}$$

$$\sigma_{Error_S(h)} = \sqrt{\frac{Error_D(h)(1-Error_D(h))}{n}}$$

$$\sigma_{Error_S(h)} \approx \sqrt{\frac{Error_S(h)(1-Error_S(h))}{n}}$$

An *N%* confidence interval for some parameter *p* that is the interval which is expected with probability *N% to contain p*

Normal distribution approximates binomial

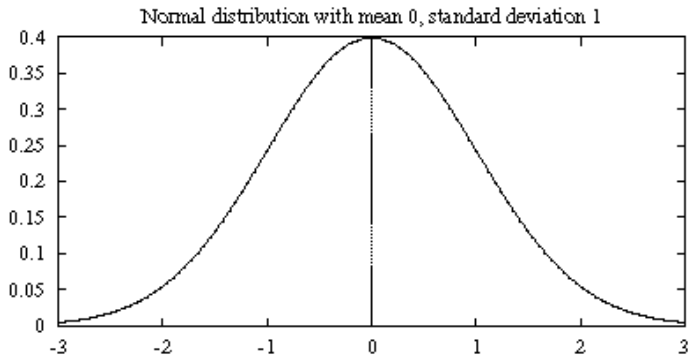
$Error_S(h)$ follows a Binomial distribution, with

- mean $\mu_{Error_S(h)} = Error_D(h)$
- standard deviation $\sigma_{Errors_S(h)} = \sqrt{\frac{Error_D(h)(1-Errors_D(h))}{n}}$

We can approximate this by a Normal distribution with the same mean and variance when $np(1-p) \geq 5$

Normal distribution

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$



The probability that *X* will fall in the interval (a, b) is given by $\int_a^b p(x)dx$

Expected, or mean value of *X* is given by E[X] = μ

Variance of *X* is given by Var(X) = σ^2

Standard deviation of *X* is given by $\sigma_X = \sigma$

How close is the estimated accuracy to its true value?

Let the probability that a Gaussian random variable X, with zero mean, takes a value between –z and z,

$Pr[-z \leq X \leq z] = c$

$Pr[X \geq z] = 5\%$ implies there is a 5% chance that X lies more than 1.65 standard deviations from the mean, or

$Pr [-1.65 \leq X \leq 1.65] = 90\%$

$Pr[X \geq z]$	z
0.001	3.09
0.005	2.58
0.01	2.33
0.05	1.65
0.10	1.28

How close is the estimated accuracy to its true value?

But $\hat{p}$ does not have zero mean and unit variance so we normalize to get

$$\Pr\left[-z < \frac{\hat{p} - p}{\sqrt{\frac{p(1-p)}{n}}} < z\right] = c$$

How close is the estimated accuracy to its true value?

To find confidence limits:

Given a particular confidence figure *c,* use the table to find the z corresponding to the probability ½ (*1-c*).

Use linear interpolation for values not in the table

$$p = \frac{\left[\hat{p} + \frac{z^2}{2n} \pm z\sqrt{\frac{\hat{p}}{n} - \frac{\hat{p}^2}{n} + \frac{z^2}{4n^2}}\right]}{\left[1 + \frac{z^2}{n}\right]}$$

How close is the estimated accuracy to its true value?

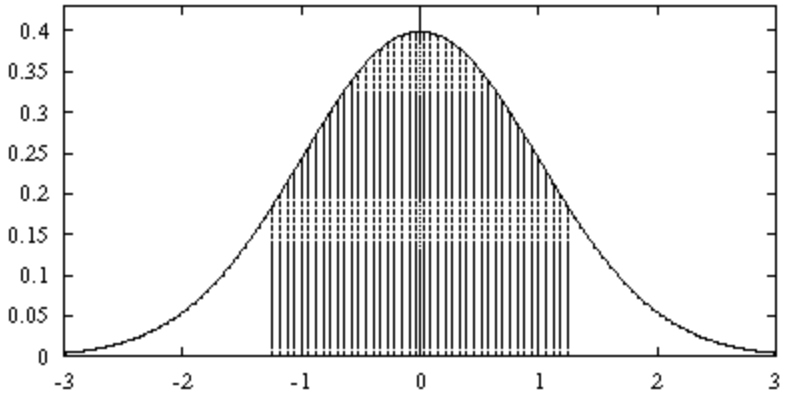
Example

$$\hat{p} = 0.75; \quad n = 1000; \quad c = 0.80; z = 1.28$$

Then with 80% confidence, we can say that the value of *p* lies in the interval [0.733,0.768]

Note: the normal distribution assumption is valid only for large *n* (i.e. $np(1-p) \geq 5$ or $n > 30$) so estimates based on smaller values of *n* should be taken with a generous dose of salt

Estimating confidence intervals



80% of area (probability) lies in $\mu \pm 1.28\sigma$

N% of area (probability) lies in $\mu \pm z_N\sigma$

$N\%$:	50%	68%	80%	90%	95%	98%	99%
z_N:	0.67	1.00	1.28	1.64	1.96	2.33	2.58

Confidence intervals

If S contains n examples, drawn independently of h and each other and $n \geq 30$ or $np(1-p) \geq 5$,

Then With approximately N% probability, $Error_S(h)$ lies in interval

$$Error_D(h) \pm Z_N \sqrt{\frac{Error_D(h)(1-Errors_D(h))}{n}}$$

equivalently, $Error_D(h)$ lies in interval

$$Errors_S(h) \pm Z_N \sqrt{\frac{Error_D(h)(1-Error_D(h))}{n}}$$

which is approximately

$$Errors_S(h) \pm Z_N \sqrt{\frac{Error_S(h)(1-Errors_S(h))}{n}}$$

One sided confidence intervals

What is the probability that $Error_D(h)$ is <u>at most U</u> ?

Symmetry of Gaussian distribution implies that confidence interval with $100(1-\alpha)\%$ confidence with lower bound L and upper bound U corresponds to a confidence interval with confidence $100\left(1-\alpha/2\right)\%$

and with upper bound U but no lower bound (or vice versa)

General approach to deriving confidence intervals

1. Identify the population parameter p to be estimated e.g., $Error_D(h)$
2. Define a suitable estimator W – preferably unbiased, minimum variance
3. Determine the distribution D_W obeyed by W , and the mean and variance of W
4. Determine the confidence interval by finding the thresholds L and U such that $N\%$ of the mass of the probability distribution D_Y *falls within the interval* $[L,U]$.

Central Limit Theorem Simplifies Confidence Interval Calculations

Consider a set of independent, identically distributed random variables $Y_1 \ldots Y_n$, all governed by an arbitrary probability distribution with mean μ and finite variance σ^2. Define the sample mean,

$$\bar{Y} \equiv \frac{1}{n} \sum_{i=1}^{n} Y_i$$

Central Limit Theorem As $n \rightarrow \infty$, the distribution governing $\bar{Y}$ approaches a Normal distribution, with mean μ and variance σ^2/n

Evaluation of a classifier with limited data

Holdout method – use part of the data for training, and the rest for testing

We may be unlucky – training data or test data may not be *representative*

Solution – Run multiple experiments with disjoint training and test data sets in which each class is represented in roughly the same proportion as in the entire data set

Estimating the performance of the learned classifier

K-fold cross-validation

Partition the data (multi) set *S* into *K* equal parts $S_1 .. S_K$

where each part has roughly the same class distribution as S.

$A = 0$

For i=1 to K do

$$\{ S_{Train} \leftarrow S - S_i; \qquad S_{Test} \leftarrow S_i$$

$$\alpha \leftarrow Learn(S_{Train})$$

$$A \leftarrow A + Accuracy(\alpha, S_{Test})$$

$$\}$$

Accuracy ← *A/K*; Output (*Accuracy*)

K-fold cross-validation

Recommended procedure for evaluating classifiers when data are limited

Use *K*-fold cross-validation (*K*=5 or 10)

Better still, repeat *K*-fold cross-validation *R* times and average the results

Difference in error between two hypotheses

We wish to estimate $d \equiv Error_D(h_1) - Error_D(h_2)$

Suppose h_1 has been tested on a sample S_1 of size n_1 drawn according to D *and* h_2 has been tested on a sample S_2 of size n_2 drawn according to D

An unbiased estimator $\hat{d} \equiv Error_{S_1}(h_1) - Error_{S_2}(h_2)$

For large n_1 and large n_2 the corresponding error estimates follow Normal distribution

Difference of two Normal distributions yields a normal distribution with variance equal to the sum of the variances of the individual distributions

Difference between errors of two hypotheses

$$d \equiv Error_D(h_1) - Error_D(h_2)$$

$$\hat{d} \equiv Errors_{S_1}(h_1) - Errors_{S_2}(h_2)$$

$$\sigma_{\hat{d}} \approx \sqrt{\frac{Errors_{S_1}(h_1)(1 - Error_{S_1}(h_1))}{n_1} + \frac{Error_{S_2}(h_2)(1 - Error_{S_2}(h_2))}{n_2}}$$

$$\hat{d} \pm z_N \sqrt{\frac{Error_{S_1}(h_1)(1 - Error_{S_1}(h_1))}{n_1} + \frac{Error_{S_2}(h_2)(1 - Error_{S_2}(h_2))}{n_2}}$$

When S_1=S_2, the variance of $\hat{d}$ is smaller and the confidence interval correct but overly conservative

Hypothesis testing

Is one hypothesis likely to be better than another?

What is the probability that $Error_D(h_1) > Error_D(h_2)$?

Suppose $Error_{S_1}(h_1) = 0.30;\ Error_{S_2}(h_2) = 0.20;\ \hat{d} = 0.10$

What is the probability that $d > 0$ given that $\hat{d} = 0.10$?

$$\Pr(d > 0 \mid \hat{d} = 0.10) = \Pr(\hat{d} < \mu_{\hat{d}} + 0.10)$$

Hypothesis testing

$$\text{If } n_1 = n_2 = 100, \sigma_{\hat{d}} \approx 0.061$$

$$\Pr(d > 0 \mid \hat{d} = 0.10) \approx \Pr(\hat{d} < \mu_{\hat{d}} + 1.64\sigma_{\hat{d}}) = 0.95$$

We accept the hypothesis that

$$Error_D(h_1) > Error_D(h_2)$$

with 95% confidence

Equivalently, we reject the opposite hypothesis –
the null hypothesis at a (1-0.95) = 0.05 level of significance

Comparing learning algorithms L_A and L_B

Which learning algorithm is better at learning f ?

Unlimited data –

Run L_A and L_B on *large* training set S_{train} drawn according to D

Test the resulting hypotheses on a *large independent* test set S_{Test} drawn according to D

Estimate $\Pr[Error_D(L_A(S_{Train})) > Error_D(L_B(S_{Train}))]$ Using

$Error_{S_{Test}}(L_A(S_{Train}))$ and $Error_{S_{Test}}(L_B(S_{Train}))$

Comparing learning algorithms L_A and L_B

Estimate the expected value of the difference in errors of L_A and L_B where expectation is taken over training sets S_{Train} drawn according to D

$$E_{S_{Train} \subset D}\left[Error_D(L_A(S_{Train})) - Error_D(L_B(S_{Train}))\right]$$

We have a limited data set S drawn from an unknown D *!!*

Comparing learning algorithms L_A and L_B

Limited data – Paired t-test

Run L_A and L_B on *large* training set S_{Train} drawn according to D

Test the resulting hypotheses on a *large independent* test set S_{Test} drawn according to D

Estimate

$$\Pr[Error_D(L_A(S_{Train})) > Error_D(L_B(S_{Train}))]$$ using

$$Error_{S_{Test}}(L_A(S_{Train}))$$ and $$Error_{S_{Test}}(L_B(S_{Train}))$$

Comparing learning algorithms L_A and L_B

Paired t-test

Partition S into k disjoint test sets $T_1, T_2, \ldots, T_k$ of equal size

For i from 1 to k do {

$S_{Test} \leftarrow T_i \; ; \; S_{Train} \leftarrow S - T_i$

$$\delta_i \leftarrow Error_{S_{Test}}(L_A(S_{Train})) - Error_{S_{Test}}(L_B(S_{Train}))$$

}

Return $$\bar{\delta} \equiv \frac{1}{k}\sum_{i=1}^{k}\delta_i$$

Comparing learning algorithms L_A *and* L_B

For large test sets, each δ_i has Normal distribution
$\bar{\delta}$ has Normal distribution if δ_i are independent
Can we estimate confidence interval for $\bar{\delta}$ as before?

δ_i are not exactly independent because of sampling from *S* as opposed to the distribution *D* (but we will pretend that they are)

We don't know the standard deviation of this distribution.
So we estimate it from sample ..But when the estimated variance is used, the distribution is no longer Normal unless *K* is large (which typically it is not)

Comparing learning algorithms L_A and L_B

Approximate N% confidence interval for

$$E_{S_{Train} \subset S}\left[Error_D(L_A(S_{Train})) - Error_D(L_B(S_{Train}))\right]$$

is given by $\bar{\delta} \pm t_{N,k-1}\psi_{\bar{\delta}}$

where $\psi_{\bar{\delta}} \equiv \sqrt{\frac{1}{k(k-1)}\sum_{i=1}^{k}(\delta_i - \bar{\delta})^2}$

is the estimate of standard deviation of the t distribution governing $\bar{\delta}$ and Z_N plays a role analogous to that of .

As $K \to \infty$, $t_{N,K-1} \to Z_N$ and $\psi_{\bar{\delta}} \to \sigma_{\bar{\delta}}$

Performance evaluation summary

- Rigorous statistical evaluation is extremely important in experimental computer science in general and machine learning in particular
  - How good is a learned hypothesis?
  - Is one hypothesis better than another?
  - Is one learning algorithm better than another on a particular learning task? (No learning algorithm outperforms all others on all tasks – No free lunch theorem)
- Different procedures for evaluation are appropriate under different conditions (large versus limited versus small sample) – Important to know when to use which evaluation method and be aware of pathological behavior (tendency to grossly overestimate or underestimate the target value under specific conditions

Modeling dependencies between attributes

- Naïve Bayes classifier assumes that the attributes are independent given the class
- What if the independence assumption does not hold?
  - We need more sophisticated models
    - Support Vector Machines
    - Higher order Markov models
    - Bayesian networks

Generative Versus Discriminative Models

- Generative models
  - Naïve Bayes, Bayes networks, etc.
- Discriminative models
  - Perceptron, Support vector machines, Logistic regression ..
- Relating generative and discriminative models
- Tradeoffs between generative and discriminative models
- Generalizations and extensions

Alternative realizations of the Bayesian recipe

Chef 1: Generative model

Note that $P(\omega_i \mid \mathbf{x}) = \dfrac{P(\mathbf{x}|\omega_i)P(\omega_i)}{P(\mathbf{x})}$

Model $P(\mathbf{x} \mid \omega_1)$, $P(\mathbf{x}|\omega_2)$, $P(\omega_1)$, and $P(\omega_2)$

Using Bayes rule, choose ω_1 if $P(\mathbf{x} \mid \omega_1)P(\omega_1) > P(\mathbf{x}|\omega_2)P(\omega_2)$

Otherwise choose ω_2

Chef 2: Discriminative Model

Model $P(\omega_1 \mid \mathbf{x}), P(\omega_2 \mid \mathbf{x})$, or the ratio $\dfrac{P(\omega_1 \mid \mathbf{x})}{P(\omega_2 \mid \mathbf{x})}$ directly

Choose ω_1 if $\dfrac{P(\omega_1 \mid \mathbf{x})}{P(\omega_2 \mid \mathbf{x})} > 1$

Otherwise choose ω_2

Generative vs. Discriminative Classifiers

Generative classifiers

- Assume some functional form for $P(\mathbf{X}|\mathrm{C})$, $P(\mathrm{C})$
- Estimate parameters of $P(\mathbf{X}|\mathrm{C})$, $P(\mathrm{C})$ directly from training data
- Use Bayes rule to calculate $P(\mathrm{C}|\mathbf{X}=\mathrm{x})$

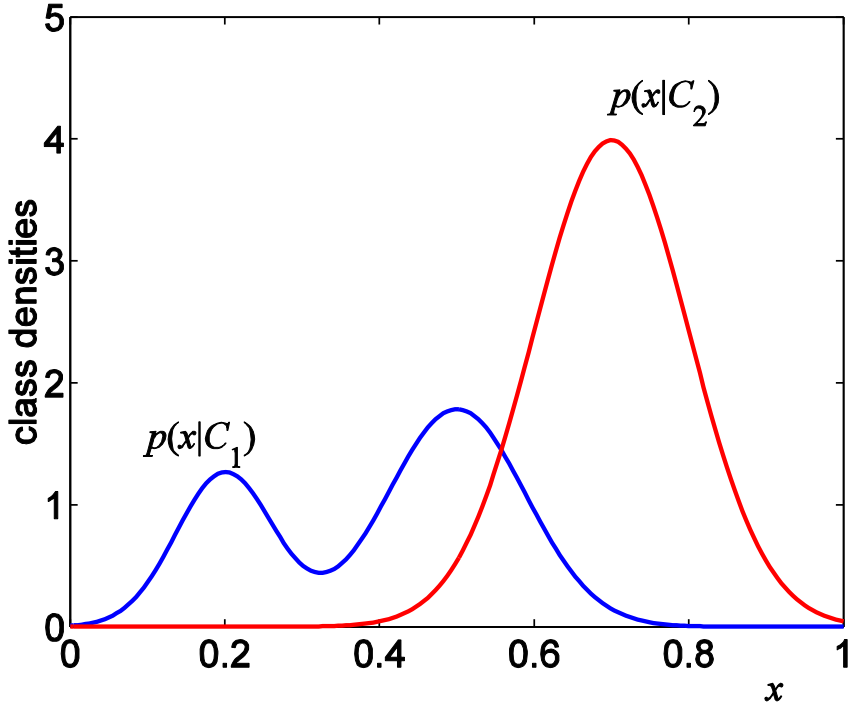
Discriminative classifiers

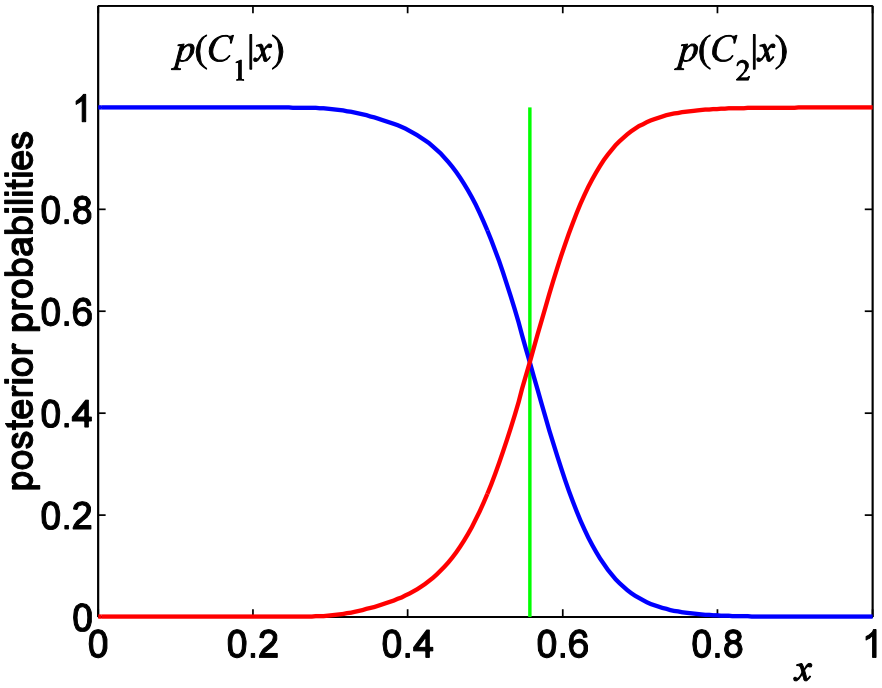
- Assume some functional form for $P(\mathrm{C}|\mathbf{X})$
- Estimate parameters of $P(\mathrm{C}|\mathbf{X})$ directly from training data

Discriminative classifiers – maximum margin version

- Assume a functional form $f(\mathbf{W})$ for the discriminant
- Find $\mathbf{W}$ that minimizes prediction error
- E.g., find $\mathbf{W}$ that maximizes the *margin of separation* between classes (e.g., SVM)

Generative vs. Discriminative Models





Which Chef cooks a better Bayesian recipe?

In theory, generative and conditional models produce identical results in the limit

- The classification produced by the generative model is the same as that produced by the discriminative model
- That is, given unlimited data, assuming that both approaches select the correct form for the relevant probability distributions or the model for the discriminant function, they will produce identical results (Why?)
- If the assumed form of the probability distributions is incorrect, then it is possible that the generative model might have a higher classification error than the discriminative model (Why?)

How about in practice?

Which Chef cooks a better Bayesian recipe?

In practice

- The error of the classifier that uses the discriminative model can be lower than that of the classifier that uses the generative model (Why?)
- Naïve Bayes is a generative model
- A perceptron is a discriminative model, and so is SVM
- An SVM can outperforms Naïve Bayes on classification

If the goal is classification, it might be useful to consider discriminative models that directly learn the classifier without going solving the harder intermediate problem of modeling the joint probability distribution of inputs and classes (Vapnik)

Neural Networks

- Decision trees are good at modeling nonlinear interactions among a small subset of attributes
- Sometimes we are interested in linear interactions among all attributes
- Simple neural networks are good at modeling such interactions
- The resulting models have close connections with naïve Bayes
  - Naïve Bayes can be seen as a special case

A simple discriminative model: Neural Networks

- Outline
- Background
- Threshold logic functions
- Connection to logic
- Connection to geometry
- Learning threshold functions – perceptron algorithm and its variants
- Perceptron convergence theorem

Background – Neural computation

- 1900 – Birth of neuroscience – Ramon Cajal et al.
- 1913 – Behaviorist or stimulus response psychology
- 1930-50: Theory of Computation, Church-Turing Thesis
- 1943: McCulloch & Pitts "A logical calculus of neuronal activity"
- 1949: Hebb – Organization of Behavior
- 1956 – Birth of Artificial Intelligence – "Computers and Thought"
- 1960-65: Perceptron model developed by Rosenblatt

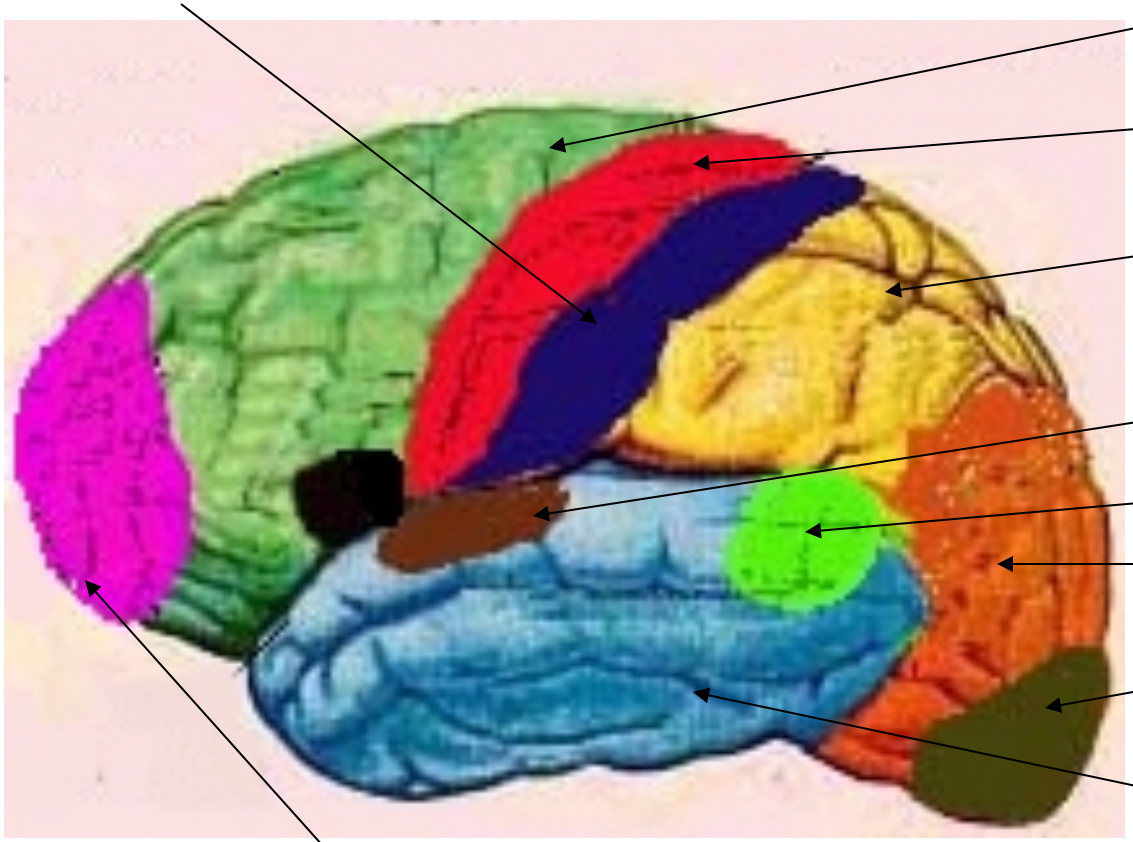
Background – Neural computation

- 1969: Minsky and Papert criticize Perceptron
- 1969: Chomsky argues for universal innate grammar
- 1970: Rise of cognitive psychology and knowledge-based AI
- 1975: Learning algorithms for multi-layer neural networks
- 1985: Resurgence of neural networks and machine learning
- 1988: Birth of computational neuroscience
- 1990: Successful applications (stock market, OCR, robotics)
- 1990-2000 New synthesis of behaviorist and cognitive or representational approaches in AI and psychology
- 2000-2010 Synthesis of logical and probabilistic approaches to representation and learning
- 2010- Data science, deep learning, big data …

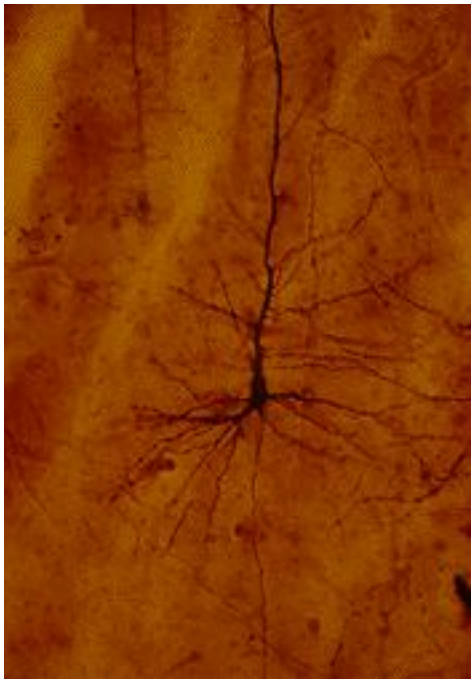
Background – Brains and Computers

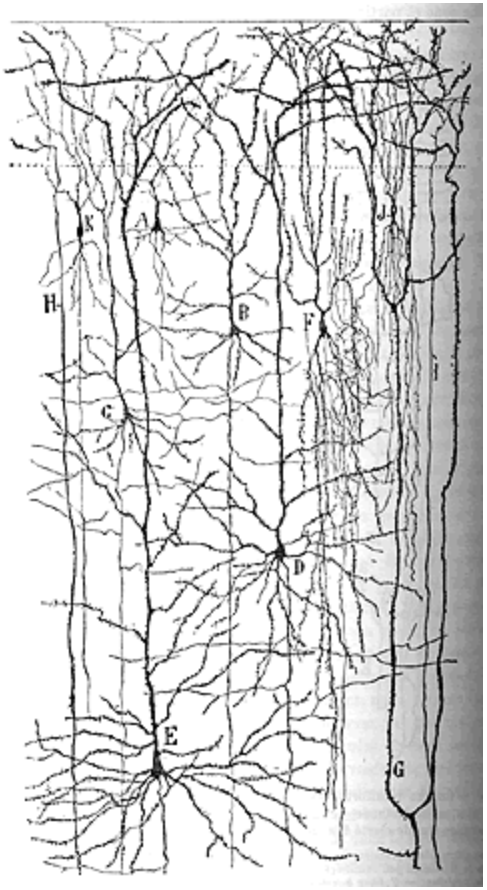
- Brain consists of 10^{11} neurons, each of which is connected to 10^4 neighbors
- Each neuron is slow (1 millisecond to respond to a stimulus) but the brain is astonishingly fast at perceptual tasks (e.g. face recognition)
- Brain processes and learns from multiple sources of sensory information (visual, tactile, auditory…)
- Brain is massively parallel, shallowly serial, modular and roughly hierarchical with recurrent and lateral connectivity within and between modules
- If cognition is -- or at least can be modeled by -- computation, it is natural to ask how and what brains compute

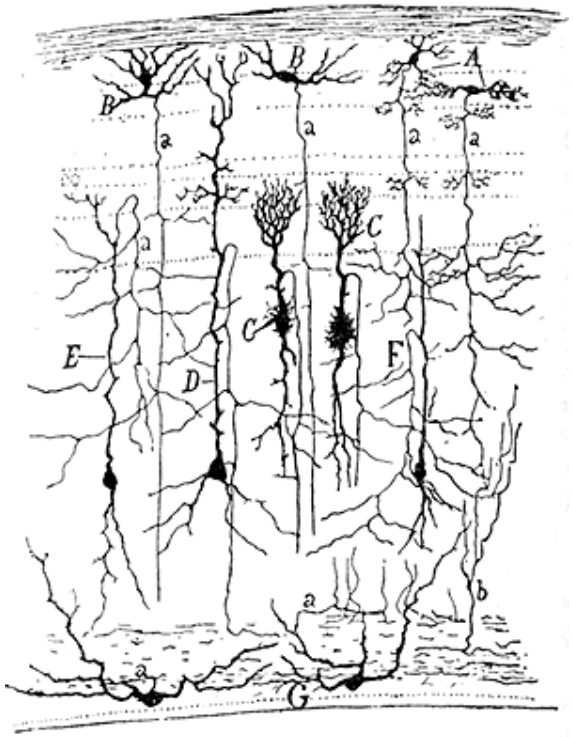
Brain and information processing



Neural Networks

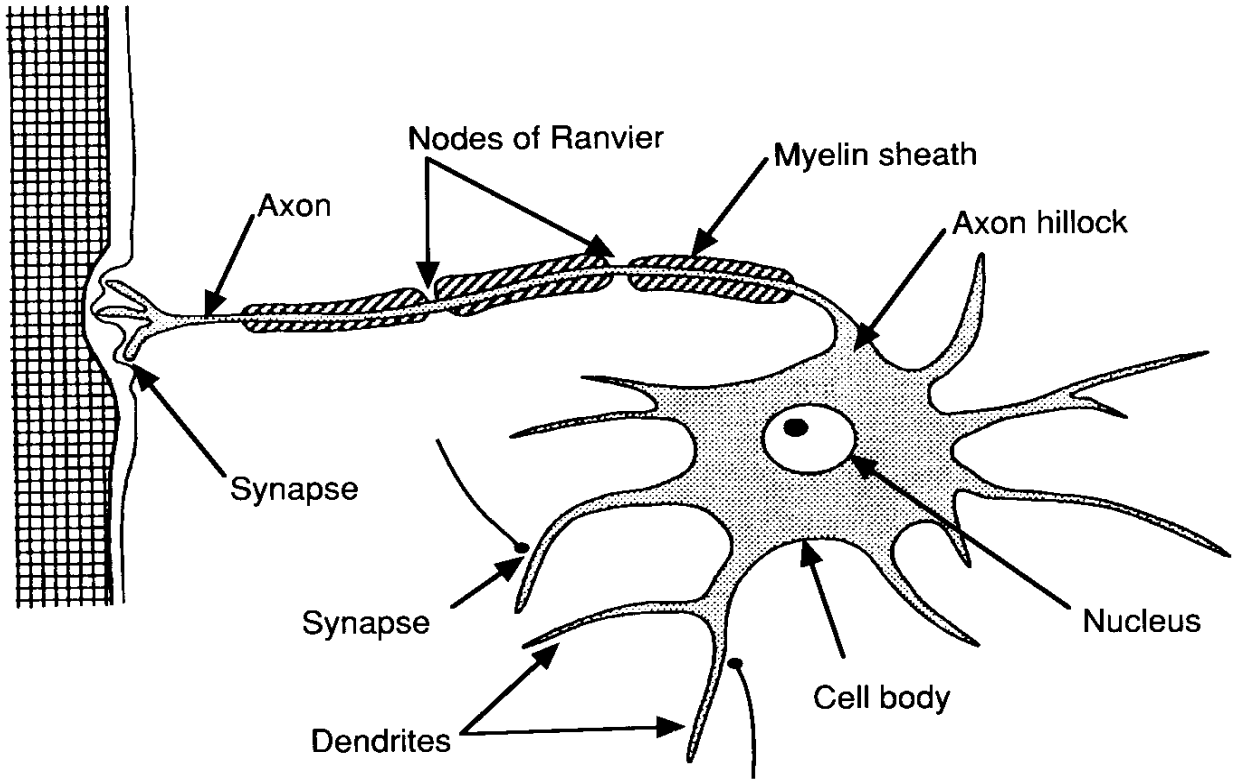




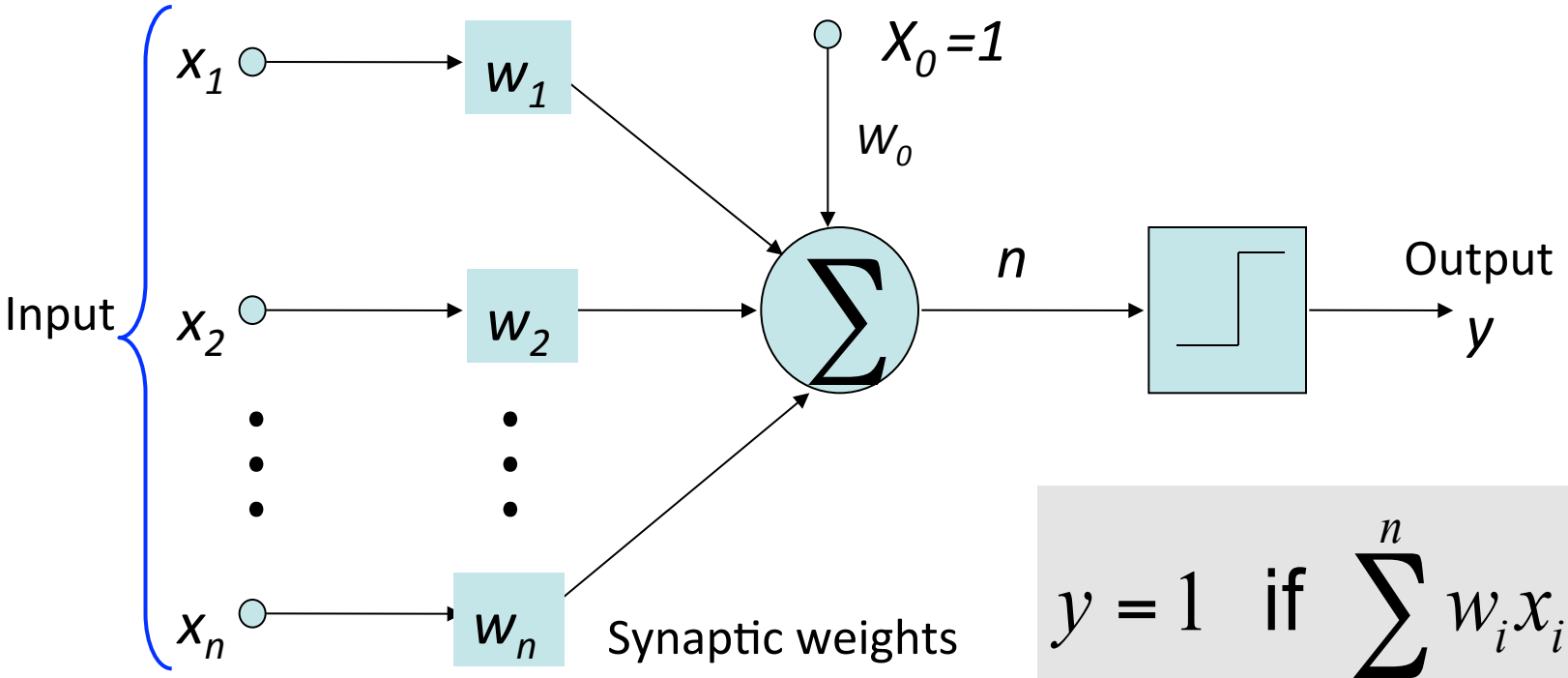


Ramon Cajal, 1900

Neurons and Computation



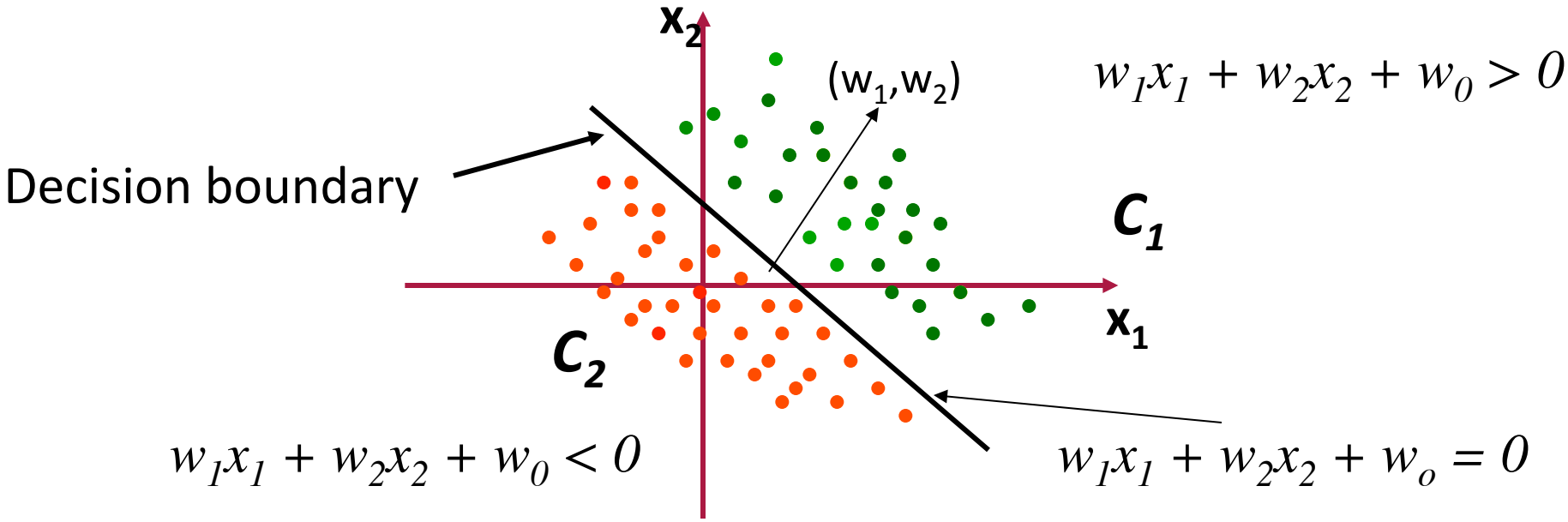
McCulloch-Pitts computational model of a neuron



$$y = 1 \quad \text{if} \quad \sum_{i=0}^{n} w_i x_i > 0$$

$$y = -1 \quad \text{otherwise}$$

Threshold neuron – Connection with Geometry



$\sum_{i=1}^{n} w_i x_i + w_0 = 0$ describes a hyperplane which divides the instance space $\Re^n$ into two half–spaces

$\chi_+ = \{\mathbf{X}_p \in \Re^n \mid \mathbf{W} \bullet \mathbf{X}_p + w_0 > 0\}$ and $\chi_- = \{\mathbf{X}_p \in \Re^n \mid \mathbf{W} \bullet \mathbf{X}_p + w_0 < 0\}$

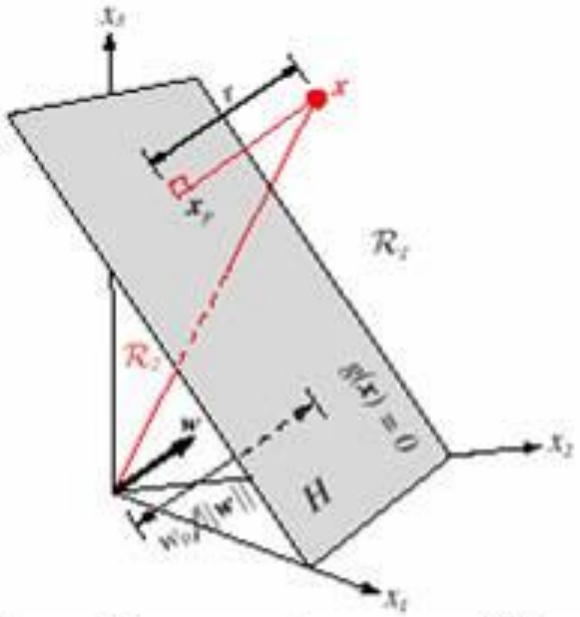
McCulloch-Pitts Neuron or Threshold Neuron

$$y = sign\left(\mathrm{W} \bullet \mathrm{X} + w_0\right)$$
$$= sign\left(\sum_{i=0}^{n} w_i x_i\right)$$
$$= sign\left(\mathrm{W}^T \mathrm{X} + w_0\right)$$

$$\mathrm{X} = \begin{bmatrix} x_1 \\ x_2 \\ \\ \\ x_n \end{bmatrix} \qquad \mathrm{W} = \begin{bmatrix} w_1 \\ w_2 \\ \\ \\ w_n \end{bmatrix}$$

$$sign\left(v\right) = 1 \text{ if } v > 0$$
$$= 0 \text{ otherwise}$$

Threshold neuron– Connection with Geometry



The (n-1)-dimensional hyperplane partitions the n-dimensional input space into two half spaces.

Threshold neuron – Connection with Geometry

Instance space $\Re^n$

Hypothesis space is the set of (n-1)-dimensional hyperplanes defined in the n-dimensional instance space

A hypothesis is defined by $\sum_{i=0}^{n} w_i x_i = 0$

- Orientation of the hyperplane is governed by $(w_1 \ldots w_n)^T$
- and the perpendicular distance of the hyperplane from the origin is given by $\left(\frac{|w_0|}{\sqrt{(w_1^2 + w_2^2 + \ldots + w_n^2)}} \right)$

Threshold neuron as a pattern classifier

- The threshold neuron can be used to classify a set of instances into one of two classes C_1, C_2
- If the output of the neuron for input pattern X_p is +1 then X_p is assigned to class C_1
- If the output is -1 then the pattern X_p is assigned to C_2

- Example

$$[w_0\ w_1\ w_2]^T = [-1\ -1\ 1]^T$$

$$\mathbf{X}_p^T = [1\ 0]^T \quad \mathbf{W} \bullet \mathbf{X}_p + w_0 = -1 + (-1) = -2$$

$$\mathbf{X}_p \text{ is assigned to class } C_2$$

Threshold neuron – Connection with Logic

- Suppose the input space is $\{0,1\}^n$
- Then threshold neuron computes a Boolean function $f : \{0,1\}^n \rightarrow \{-1,1\}$

Example

Let $w_0 = -1.5$; $w_1 = w_2 = 1$

In this case, the threshold neuron implements the logical AND function

x_1	x_2	g(X)	y
0	0	-1.5	-1
0	1	-0.5	-1
1	0	-0.5	-1
1	1	0.5	1

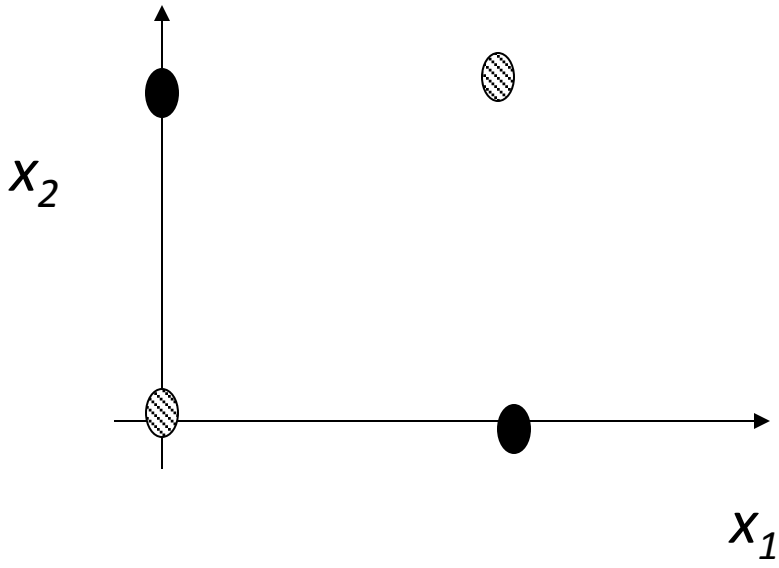
Threshold neuron – Connection with Logic

- A threshold neuron with the appropriate choice of weights can implement Boolean AND, OR, and NOT function
- Theorem: For any arbitrary Boolean function *f*, there exists a network of threshold neurons that can implement *f*.
- Theorem: Any arbitrary finite state automaton can be realized using threshold neurons and *delay* units
- Networks of threshold neurons, given access to unbounded memory, can compute any Turing-computable function
- Corollary: Brains if given access to enough working memory, can compute any computable function

Threshold neuron: Connection with Logic

Theorem: There exist functions that cannot be implemented by a single threshold neuron.

Example Exclusive OR



Why?

Threshold neuron – Connection with Logic

- Definition: A function that can be computed by a single threshold neuron is called a threshold function
- Of the 16 2-input Boolean functions, 14 are Boolean threshold functions
- As *n* increases, the number of Boolean threshold functions becomes an increasingly small fraction of the total number of *n*-input Boolean functions

$$N_{Threshold}(n) \leq 2^{n^2}$$

$$N_{Boolean}(n) = 2^{2^n}$$

Terminology and Notation

- Synonyms: Threshold function, Linearly separable function, linear discriminant function
- Synonyms: Threshold neuron, McCulloch-Pitts neuron, Perceptron, Threshold Logic Unit (TLU)
- We often include w_0 as one of the components of W and incorporate x_0 as the corresponding component of X with the understanding that x_0 =1. Then y=1 if W.X > 0 and y=-1 otherwise.

Learning Threshold functions

A training example E_k is an ordered pair (X_k, d_k) where

$$\mathbf{X}_k = \left[x_{0k} \; x_{1k} \; .... \; x_{nk} \right]^T$$

is an (n+1) dimensional input pattern, $d_k = f(\mathbf{X}_k) \in \{-1, 1\}$

is the desired output of the classifier and f is an unknown target function to be learned.

A training set E is simply a multi-set of examples.

Learning Threshold functions

$$S^+ = \{X_k | (X_k, d_k) \in E \text{ and } d_k = 1\}$$
$$S^- = \{X_k | (X_k, d_k) \in E \text{ and } d_k = -1\}$$

We say that a training set E is linearly separable if and only if

$$\exists W^* \text{ such that } \forall X_p \in S^+, W^* \bullet X_p > 0$$
$$\text{and } \forall X_p \in S^-, W^* \bullet X_p < 0$$

Learning Task: Given a linearly separable training set *E,* find a solution

$$W^* \text{ such that } \forall X_p \in S^+, W^* \bullet X_p > 0 \text{ and } \forall X_p \in S^-, W^* \bullet X_p < 0$$

Rosenblatt's Perceptron Learning Algorithm

Initialize $\mathbf{W} = [0\ 0 \ldots\ldots 0]^T$ Set learning rate $\eta > 0$

Repeat until a complete pass through E results in no weight updates

For each training example $E_k \in E$

{

$$y_k \leftarrow sign(\mathbf{W} \bullet \mathbf{X}_k)$$

$$\mathbf{W} \leftarrow \mathbf{W} + \eta(d_k - y_k)\mathbf{X}_k$$

}

$$\mathbf{W}^* \leftarrow \mathbf{W}; \quad \text{Return}(\mathbf{W}^*)$$

Perceptron learning algorithm –Example

Let

$S^+ = \{(1, 1, 1), (1, 1, -1), (1, 0, -1)\}$

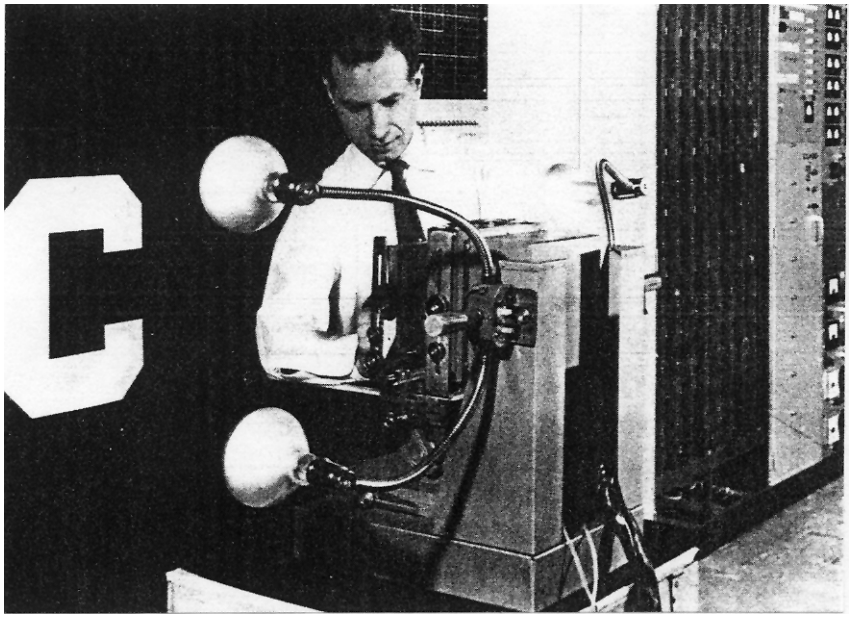
$S^- = \{(1,-1, -1), (1,-1, 1), (1,0, 1)\}$

W= (0 0 0);

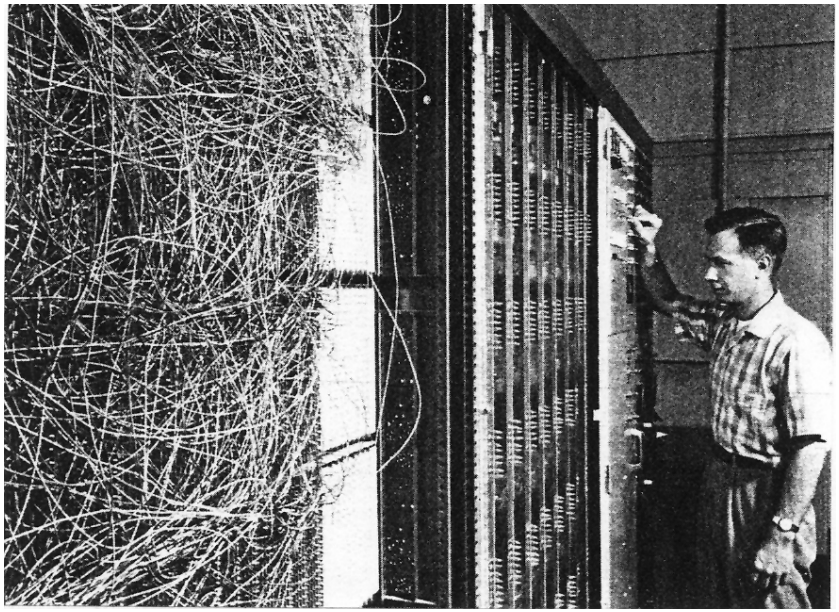
$$\eta = \frac{1}{2}$$

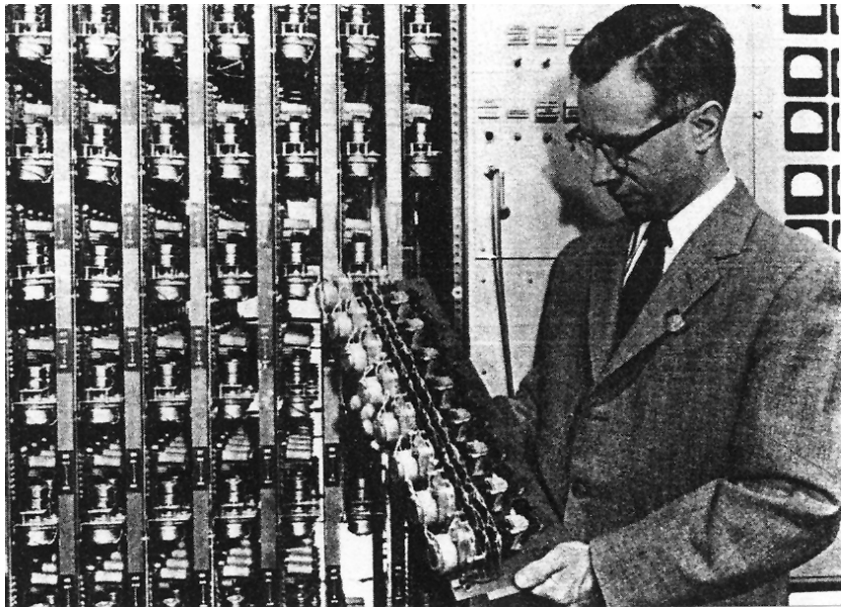
X_k	d_k	W	$W.X_k$	y_k	Update?	Updated W
(1, 1, 1)	1	(0, 0, 0)	0	-1	Yes	(1, 1, 1)
(1, 1, -1)	1	(1, 1, 1)	1	1	No	(1, 1, 1)
(1,0, -1)	1	(1, 1, 1)	0	-1	Yes	(2, 1, 0)
(1, -1, -1)	-1	(2, 1, 0)	1	1	Yes	(1, 2, 1)
(1,-1, 1)	-1	(1, 2, 1)	0	-1	No	(1, 2, 1)
(1,0, 1)	-1	(1, 2, 1)	2	1	Yes	(0, 2, 0)
(1, 1, 1)	1	(0, 2, 0)	2	1	No	(0, 2, 0)

Perceptron (1957)



Perceptron





Perceptron Convergence Theorem (Novikoff)

Theorem Let $E = \{(\mathbf{X}_k, d_k)\}$ be a training set where $\mathbf{X}_k \in \{1\} \times \Re^n$ and $d_k \in \{-1, 1\}$

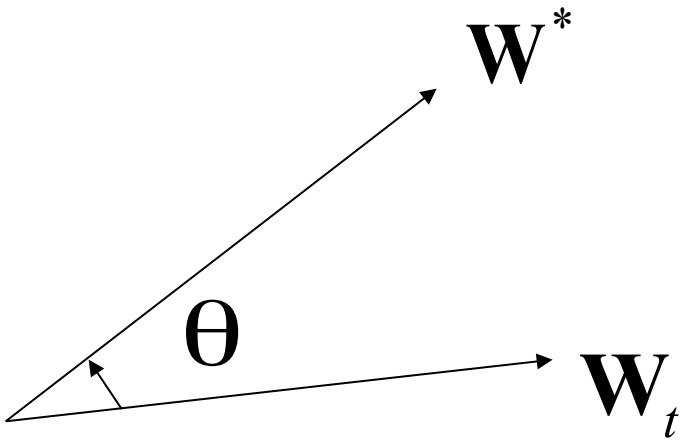
Let $S^+ = \{\mathbf{X}_k | (\mathbf{X}_k, d_k) \in E \ \& \ d_k = 1\}$ and $S^- = \{\mathbf{X}_k | (\mathbf{X}_k, d_k) \in E \ \& \ d_k = -1\}$

The perceptron algorithm is guaranteed to terminate after a bounded number t of weight updates with a weight vector $\mathbf{W}^*$ such that $\forall \mathbf{X}_k \in S^+, \mathbf{W}^* \bullet \mathbf{X}_k \geq \delta$ and $\forall \mathbf{X}_k \in S^-, \mathbf{W}^* \bullet \mathbf{X}_k \leq -\delta$ for *some* $\delta > 0$, whenever such $\mathbf{W}^* \in \Re^{n+1}$ and $\delta > 0$ exist -- that is, *E is linearly separable.* The bound on the number *t* of weight updates is given by

$$t \leq \left(\frac{\|\mathbf{W}^*\| L}{\delta} \right)^2 \quad \text{where } L = \max_{\mathbf{X}_k \in S} \|\mathbf{X}_k\| \quad \text{and } S = S^+ \bigcup S^-$$

Proof of Perceptron Convergence Theorem

Let $\mathbf{W}_t$ be the weight vector after t weight *updates*.



$$\text{Invariant}: \ \forall\theta \quad |\cos\theta| \leq 1$$

Proof of Perceptron Convergence Theorem

Let W^* be such that

$$\forall \mathrm{X}_k \in S^+, \mathrm{W}^* \bullet \mathrm{X}_k \geq \delta \text{ and } \forall \mathrm{X}_k \in S^-, \mathrm{W}^* \bullet \mathrm{X}_k \leq -\delta$$

WLOG assume that $\mathrm{W}^* \bullet \mathrm{X} = 0$ passes through the origin.

Let $\forall \mathrm{X}_k \in S^+, Z_k = \mathrm{X}_k,$

$$\forall \mathrm{X}_k \in S^-, Z_k = -\mathrm{X}_k,$$

$$Z = \{Z_k\}$$

$$\left(\forall \mathrm{X}_k \in S^+, \mathrm{W}^* \bullet \mathrm{X}_k \geq \delta \ \& \ \forall \mathrm{X}_k \in S^-, \mathrm{W}^* \bullet \mathrm{X}_k \leq -\delta\right)$$

$$\Leftrightarrow \left(\forall Z_k \in Z, \mathrm{W}^* \bullet Z_k \geq \delta\right).$$

Let $E' = \{(Z_k, 1)\}$

Proof of Perceptron Convergence Theorem

$$W_{t+1} = W_t + \eta(d_k - y_k)Z_k$$

where $W_0 = [0\,0\,....\,0]^T$ and $\eta > 0$

[Weight update based on example $(Z_k, 1)$]

$$\Leftrightarrow [(d_k = 1) \wedge (y_k = -1)]$$

$$\therefore W^* \bullet W_{t+1} = W^* \bullet (W_t + 2\eta Z_k)$$

$$= (W^* \bullet W_t) + 2\eta(W^* \bullet Z_k)$$

Since $\forall Z_k \in Z, (W^* \bullet Z_k \geq \delta), W^* \bullet W_{t+1} \geq W^* \bullet W_t + 2\eta\delta$

$\therefore \forall t \quad W^* \bullet W_t \geq 2t\eta\delta$ ....(a)

Proof of Perceptron Convergence Theorem

$$\|W_{t+1}\|^2 \equiv W_{t+1} \bullet W_{t+1}$$
$$= (W_t + 2\eta Z_k) \bullet (W_t + 2\eta Z_k)$$
$$= (W_t \bullet W_t) + 4\eta (W_t \bullet Z_k) + 4\eta^2 (Z_k \bullet Z_k)$$

Note weight update based on $Z_k \Leftrightarrow (W_t \bullet Z_k \leq 0)$

$$\therefore \|W_{t+1}\|^2 \leq \|W_t\|^2 + 4\eta^2 \|Z_k\|^2 \leq \|W_t\|^2 + 4\eta^2 L^2$$

Hence $\|W_t\|^2 \leq 4t\eta^2 L^2$

$$\therefore \forall t \ \|W_t\| \leq 2\eta L\sqrt{t} \quad \text{.....(b)}$$

Proof of Perceptron Convergence Theorem

From (a) we have : $\forall t \quad \left(\mathbf{W}^* \bullet \mathbf{W}_t\right) \geq 2t\eta\delta$

$$\Rightarrow \left\{\forall t \quad 2t\eta\delta \leq \left(\mathbf{W}^* \bullet \mathbf{W}_t\right)\right\} \Rightarrow \left\{\forall t \quad 2t\eta\delta \leq \left\|\mathbf{W}^*\right\|\left\|\mathbf{W}_t\right\|\cos\theta\right\}$$

$$\Rightarrow \left\{\forall t \quad 2t\eta\delta \leq \left\|\mathbf{W}^*\right\|\left\|\mathbf{W}_t\right\|\right\} \because \forall \theta \quad \cos\theta \leq 1,$$

Substituting for an upper bound on $\left\|\mathbf{W}_t\right\|$ from (b),

$$\forall t \quad \left\{2t\eta\delta \leq \left\|\mathbf{W}^*\right\| 2\eta L\sqrt{t}\right\} \Rightarrow \left\{\forall t \quad \left(\delta\sqrt{\mathrm{t}} \leq \left\|\mathbf{W}^*\right\| L\right)\right\}$$

$$\Rightarrow t \leq \left(\frac{\left\|\mathbf{W}^*\right\| L}{\delta}\right)^2$$

Notes on the Perceptron Convergence Theorem

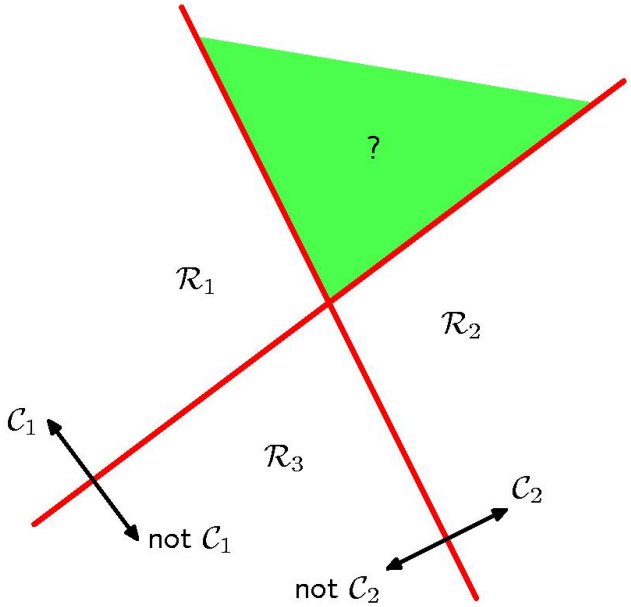
- The bound on the number of weight updates does not depend on the learning rate
- The bound is not useful in determining when to stop the algorithm because it depends on the norm of the unknown weight vector and delta
- The convergence theorem offers no guarantees when the training data set is not linearly separable

Exercise: Prove that the perceptron algorithm is robust with respect to fluctuations in the learning rate

$$0 < \eta_{\min} \leq \eta_t \leq \eta_{\max} < \infty$$

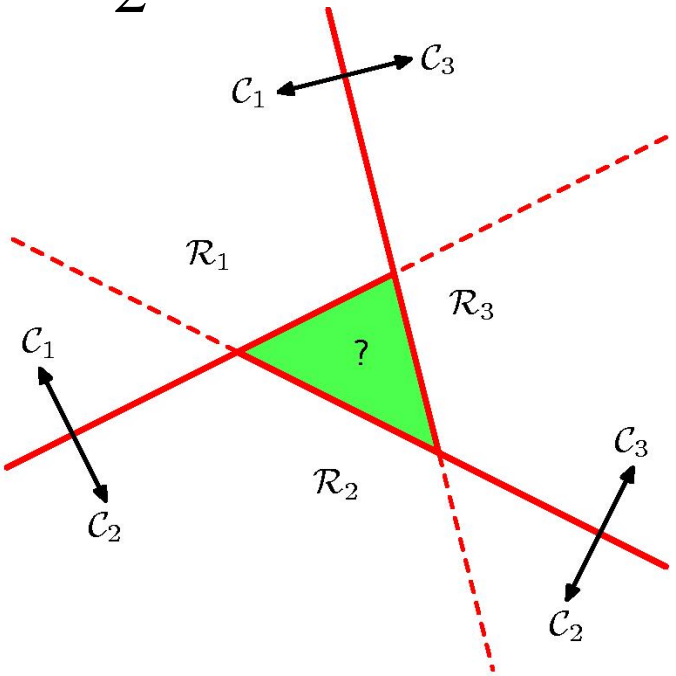
Multiple classes

$K-1$ binary classifiers



One-versus-rest

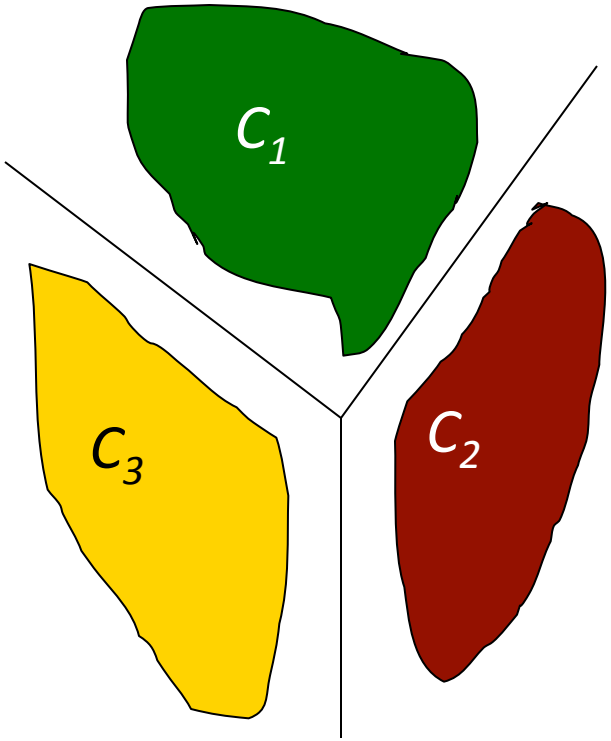
$\frac{K(K-1)}{2}$ binary classifiers



One-versus-one

Problem: Green region has ambiguous class membership

Multi-category classifiers



Define K linear functions of the form:

$$y_k(\mathbf{X}) = \mathbf{W}_k{}^T \mathbf{X} + w_{k0}$$

$$h(\mathbf{X}) = \arg\max_k y_k(\mathbf{X})$$

$$= \arg\max_k \left(\mathbf{W}_k{}^T \mathbf{X} + w_{k0}\right)$$

Decision surface between class C_k and C_j

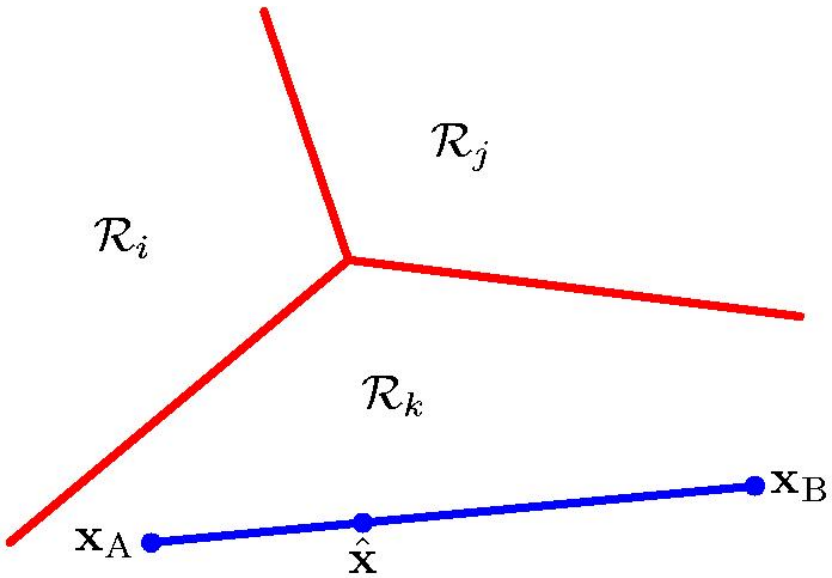
$$\left(\mathbf{W}_k - \mathbf{W}_j\right)^T \mathbf{X} + \left(w_{k0} - w_{j0}\right) = 0$$

Linear separator for K classes

- Decision regions defined by

$$\left(\mathbf{W}_k - \mathbf{W}_j\right)^T \mathbf{X} + \left(w_{k0} - w_{j0}\right) = 0$$

are singly connected and convex



For any points $\mathbf{X}_A, \mathbf{X}_B \in R_k$,

any $\hat{\mathbf{X}}$ that lies on the line connecting $\mathbf{X}_A$ and $\mathbf{X}_B$

$\hat{\mathbf{X}} = \lambda \mathbf{X}_A + (1-\lambda)\mathbf{X}_B$ where $0 \le \lambda \le 1$

also lies in R_k

Winner-Take-All Networks

$y_{ip} = 1$ iff $\mathbf{W}_i \bullet \mathbf{X}_p > \mathbf{W}_j \bullet \mathbf{X}_p \quad \forall j \neq i$

$y_{ip} = 0$ otherwise

Note: $\mathbf{W}_j$ are augmented weight vectors

$$\mathbf{W}_1 = [1\ -1\ -1]^T, \mathbf{W}_2 = [1\ 1\ 1]^T, \mathbf{W}_3 = [2\ 0\ 0]^T$$

			$W_1.X_p$	$W_2.X_p$	$W_3.X_p$	y_1	y_2	y_3
1	-1	-1	3	-1	2	1	0	0
1	-1	+1	1	1	2	0	0	1
1	+1	-1	1	1	2	0	0	1
1	+1	+1	-1	3	2	0	1	0

What does neuron 3 compute?

Linear separability of multiple classes

Let $S_1, S_2, S_3 \ldots S_M$ be multisets of instances

Let $C_1, C_2, C_3 \ldots C_M$ be disjoint classes

$\forall i \;\; S_i \subseteq C_i$

$\forall i \neq j \;\; C_i \bigcap C_j = \varnothing$

We say that the sets $S_1, S_2, S_3 \ldots S_M$ are linearly separable iff $\exists$ weight vectors $\mathrm{W}_1^*, \mathrm{W}_2^*, .. \mathrm{W}_M^*$ such that

$$\forall i \;\; \left\{ \forall \mathrm{X}_p \in S_i, \left(\mathrm{W}_i^* \bullet \mathrm{X}_p > \mathrm{W}_j^* \bullet \mathrm{X}_p \right) \forall j \neq i \right\}$$

Training WTA Classifiers

$d_{kp} = 1$ iff $\mathbf{X}_p \in C_k$; $d_{kp} = 0$ otherwise

$y_{kp} = 1$ iff $\mathbf{W}_k \bullet \mathbf{X}_p > \mathbf{W}_j \bullet \mathbf{X}_p \quad \forall k \neq j$

Suppose $d_{kp} = 1, y_{jp} = 1$ and $y_{kp} = 0$

$\mathbf{W}_k \leftarrow \mathbf{W}_k + \eta \mathbf{X}_p$; $\mathbf{W}_j \leftarrow \mathbf{W}_j - \eta \mathbf{X}_p$;

All other weights are left unchanged.

Suppose $d_{kp} = 1, y_{jp} = 0$ and $y_{kp} = 1$.

The weights are unchanged.

Suppose $d_{kp} = 1, \forall j \; y_{jp} = 0$ (there was a tie)

$\mathbf{W}_k \leftarrow \mathbf{W}_k + \eta \mathbf{X}_p$

All other weights are left unchanged.

WTA Convergence Theorem

Given a linearly separable training set, the WTA learning algorithm is guaranteed to converge to a solution within a finite number of weight updates.

Proof Sketch: Transform the WTA training problem to the problem of training a single perceptron using a suitably transformed training set. Then the proof of WTA learning algorithm reduces to the proof of perceptron learning algorithm

WTA Convergence Theorem

Let $\mathbf{W}^T = [\mathbf{W}_1 \mathbf{W}_2 .... \mathbf{W}_M]^T$ be a concatenation of the weight vectors associated with the M neurons in the WTA group. Consider a multi-category training set $E = \{(\mathbf{X}_p, f(\mathbf{X}_p))\}$ where $\forall \mathbf{X}_p \ f(\mathbf{X}_p) \in \{C_1, ... C_M\}$

Let $\mathbf{X}_p \in C_1$. Generate $(M\text{-}1)$ training examples using $\mathbf{X}_p$ for an $M(n+1)$ input perceptron :

$\mathbf{X}_{p12} = [\mathbf{X}_p \ -\mathbf{X}_p \ \phi \ \ \phi ... \phi]$

$\mathbf{X}_{p13} = [\mathbf{X}_p \ \ \phi \ -\mathbf{X}_p \ \phi ... \phi]$

...

$\mathbf{X}_{p1M} = [\mathbf{X}_p \ \phi \ \ \phi ... \phi \ -\mathbf{X}_p]$

where ϕ is an all zero vector with the same dimension as $\mathbf{X}_p$ and set the desired output of the corresponding perceptron to be 1 in each case.

Similarly, from each training example for an $(n+1)$ – input WTA, we can generate $(M-1)$ examples for an $M(n+1)$ input single neuron.

Let the union of the resulting $|E|(M-1)$ examples be E'

WTA Convergence Theorem

By construction, there is a one-to-one correspondence between the weight vector $\mathbf{W}^T = [\mathbf{W}_1 \mathbf{W}_2 .... \mathbf{W}_M]^T$ that results from training an M-neuron WTA on the multi-category set of examples E and the result of training an $M(n+1)$ input perceptron on the transformed training set E'. Hence the convergence proof of WTA learning algorithm follows from the perceptron convergence theorem.

Weight space representation

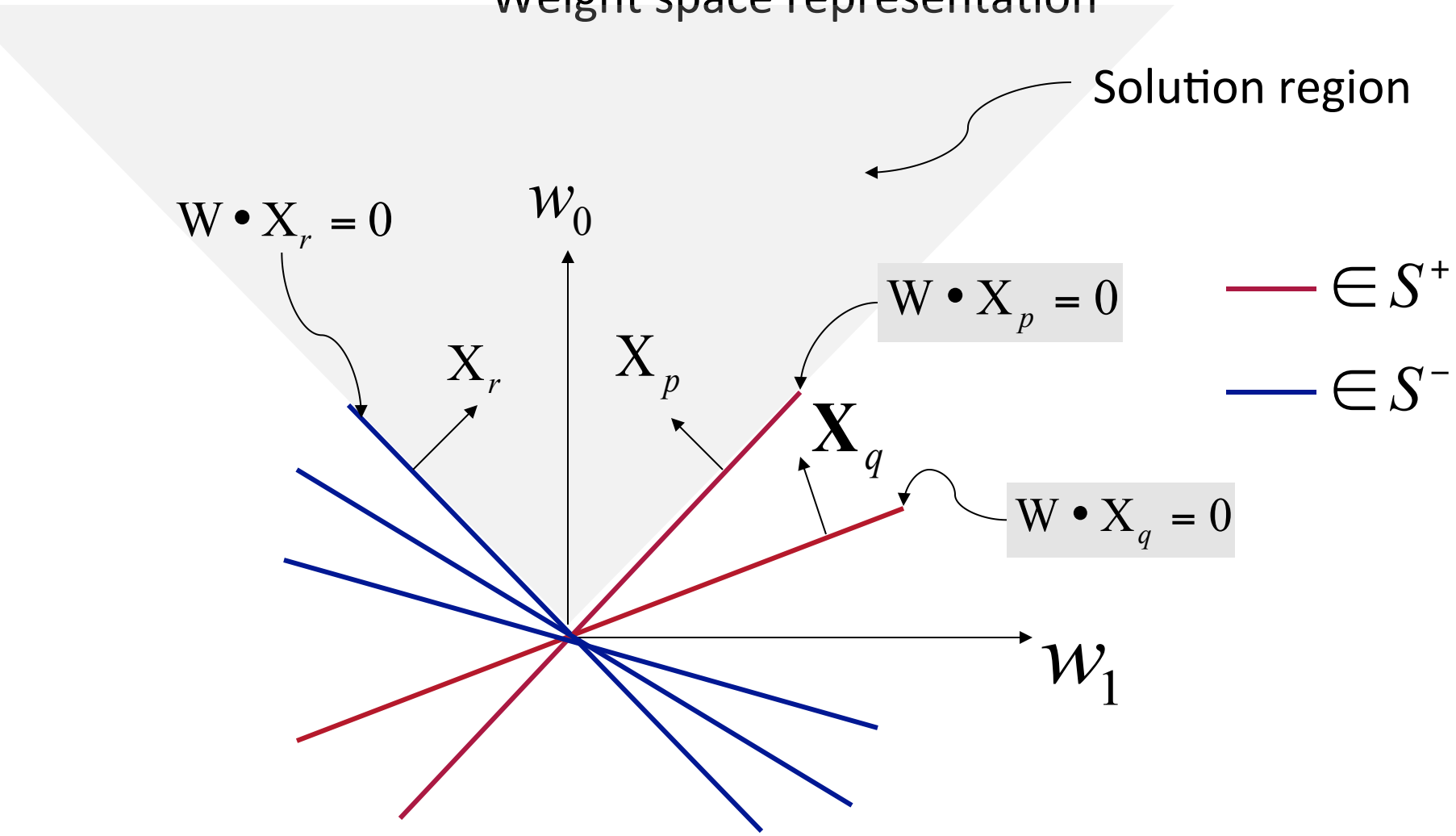
Pattern space representation

- Coordinates of space correspond to attributes (features)
- A point in the space represents an instance
- Weight vector W_v defines a hyperplane W_v .X=0

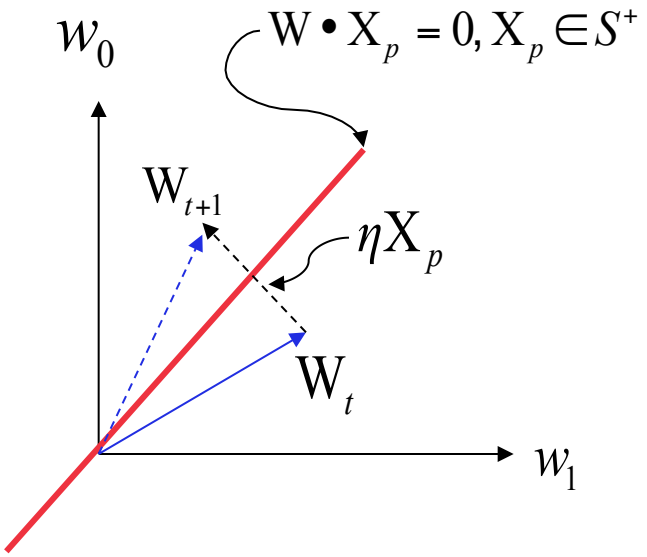
Weight space (dual) representation

- Coordinates define a weight space
- A point in the space represents a choice of weights W_v
- An instance X_p defines a hyperplane W. X_p=0

Weight space representation



Weight space representation



$$W_{t+1} \leftarrow W_t + \eta X_p$$

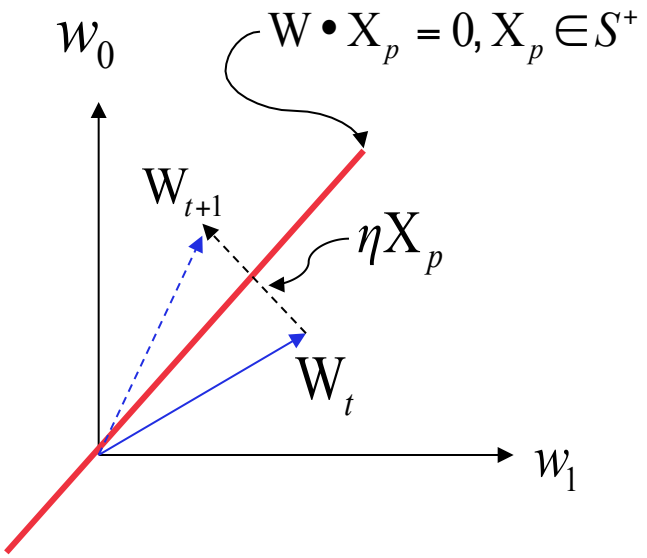
Fractional correction rule

$$\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t + \lambda \left(\frac{\left| \mathbf{W}_t \bullet \mathbf{X}_p \right| + \varepsilon}{\mathbf{X}_p \bullet \mathbf{X}_p + \varepsilon} \right) \left(d_p - y_p \right) \mathbf{X}_p$$

$0 < \lambda < 1; \lambda \neq 0.5$ when $d_p, y_p \in \{-1, 1\}$

$\varepsilon > 0$ is a constant (to handle the case when the dot product $\mathbf{W}_t \bullet \mathbf{X}_p$ or $\mathbf{X}_p \bullet \mathbf{X}_p$ (or both) approach zero.

Weight space representation



$$W_{t+1} \leftarrow W_t + \eta X_p$$

The Perceptron Algorithm Revisited

The perceptron works by adding misclassified positive or subtracting misclassified negative examples to an arbitrary weight vector, which (without loss of generality) we assumed to be the zero vector. So the final weight vector is a linear combination of training points

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i,$$

where, since the sign of the coefficient of $\mathbf{x}_i$ is given by label y_i, the α_i are positive values, proportional to the number of times, misclassification of has caused the weight to be updated. It is called the embedding strength of the pattern .

$\mathbf{x}_i$

$\mathbf{x}_i$

Dual Representation

The decision function can be rewritten as:

$$h(\mathbf{x}) = \text{sgn}\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right) = \text{sgn}\left(\left\langle \left(\sum_{j=1}^{l} \alpha_j y_j \mathbf{x}_j \right), \mathbf{x} \right\rangle + b\right)$$

$$= \text{sgn}\left(\left(\sum_{j=1}^{l} \alpha_j y_j \right) \langle \mathbf{x}_j, \mathbf{x} \rangle + b\right)$$

The update rule is:

$$\text{on training example } (\mathbf{x}_i, y_i)$$

$$\text{if } \; y_i \left(\left(\sum_{j=1}^{l} \alpha_j y_j \right) \langle \mathbf{x}_j, \mathbf{x}_i \rangle + b \right) \leq 0,$$

$$\text{then } \; \alpha_i \leftarrow \alpha_i + \eta$$

WLOG, we can take $\eta = 1.$

Limitations of perceptrons

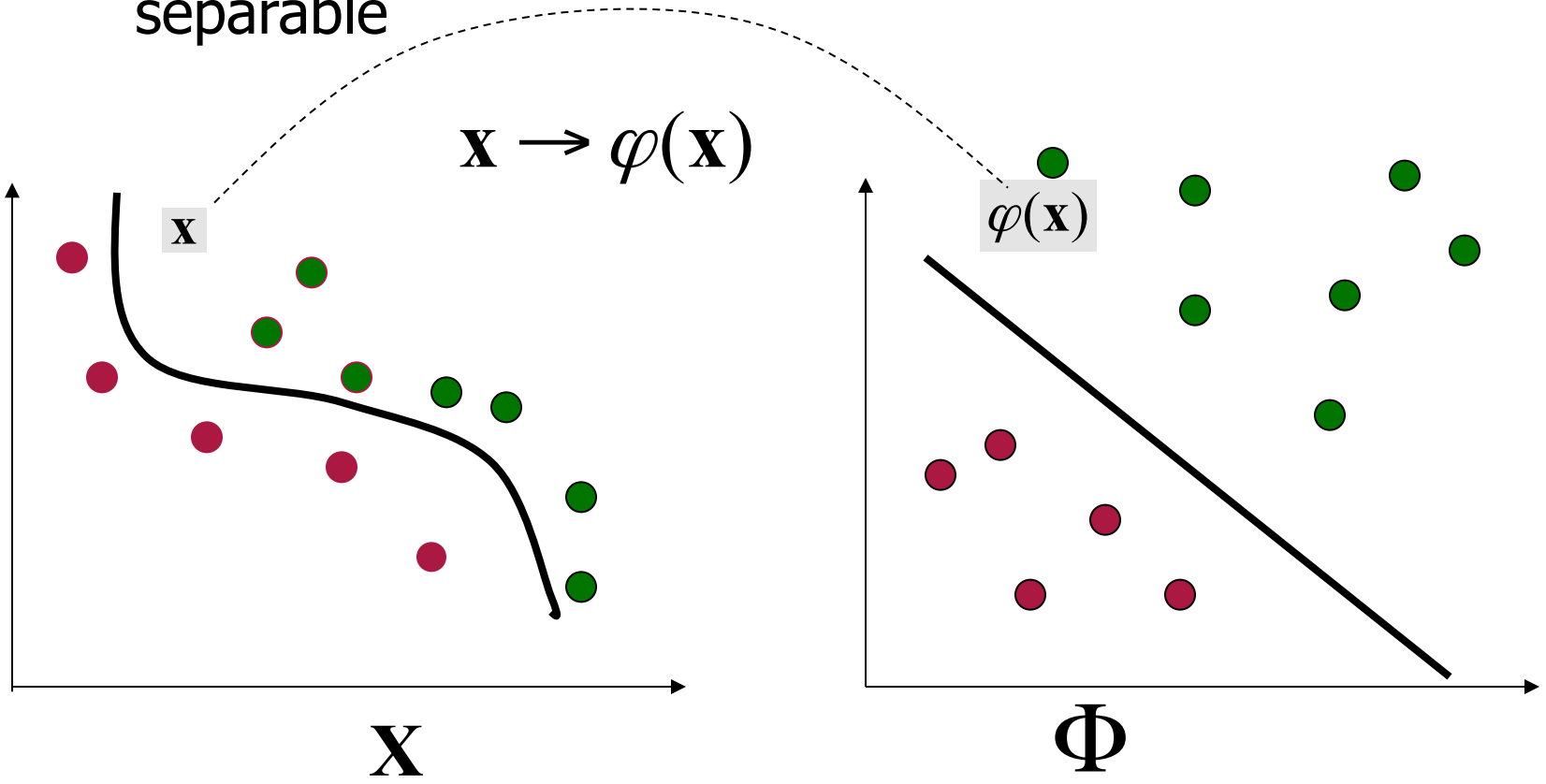
- Perceptrons can only represent threshold functions
- Perceptrons can only learn linear decision boundaries

What if the data are not linearly separable?

- More complex networks?
- Non-linear transformations into a feature space where the data become separable?

Extending Linear Classifiers Learning in feature spaces

Map data into a feature space where they are linearly separable



Exclusive OR revisited

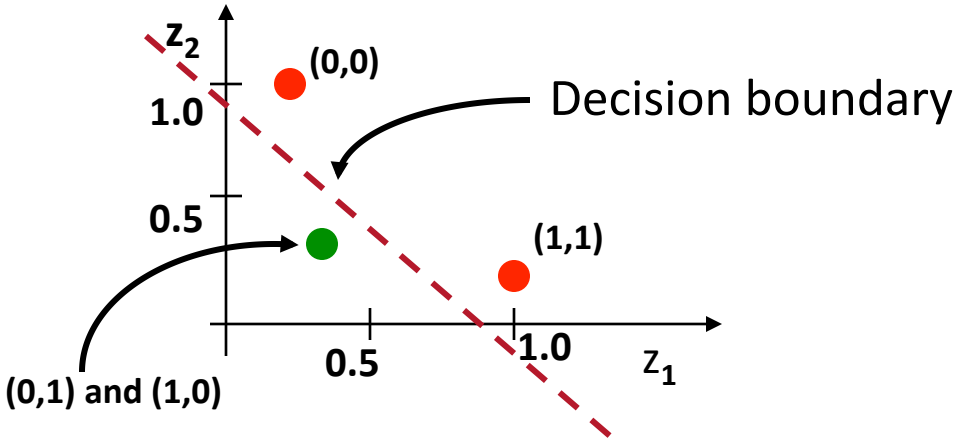
In the feature (hidden) space:

$$\varphi_1(x_1, x_2) = e^{-||\mathbf{X}-\mathbf{W}_1||^2} = z_1$$

$$\varphi_2(x_1, x_2) = e^{-||\mathbf{X}-\mathbf{W}_2||^2} = z_2$$

$$\mathbf{W}_1 = [1,1]^T$$

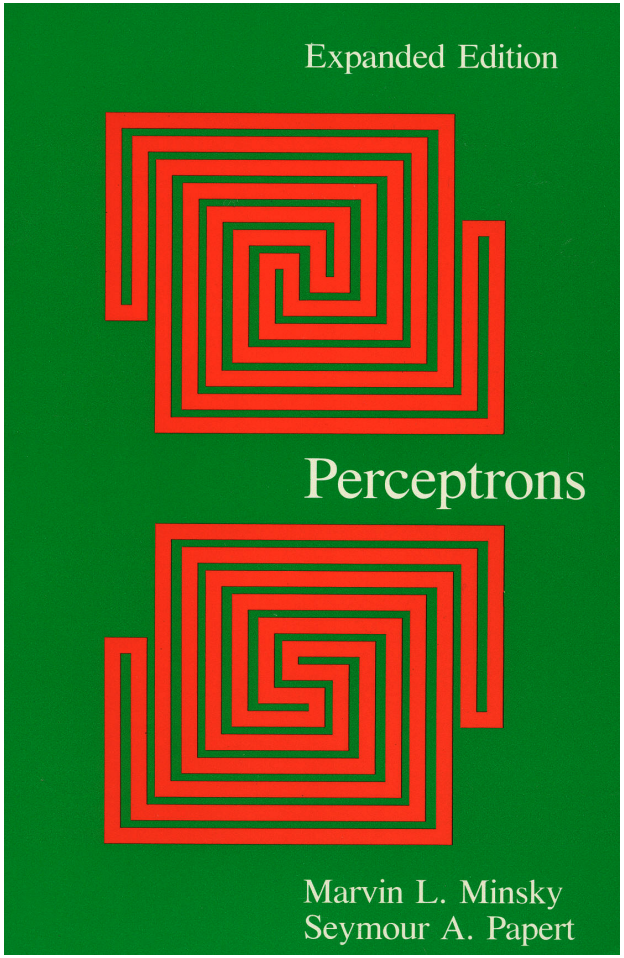
$$\mathbf{W}_2 = [0,0]^T$$



When mapped into the feature space < z_1 , z_2 >, C1 and C2 become *linearly separable.* So a linear classifier with $\varphi_1(x)$ and $\varphi_2(x)$ as inputs can be used to solve the XOR problem.

"Perceptrons" (1969)

"The perceptron […] has many features that attract attention: its linearity, its intriguing learning theorem; its clear paradigmatic simplicity as a kind of parallel computation. *There is no reason to suppose that any of these virtues carry over to the many-layered version.* Nevertheless, *we consider it to be an important research problem to elucidate (or reject) our intuitive judgement that the extension is sterile.*"
[pp. 231 – 232]



Learning in the Feature Spaces

High dimensional Feature spaces

$$\mathbf{x} = (x_1, x_2 \ldots . x_n) \rightarrow \varphi(\mathbf{x}) = (\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots . \varphi_d(\mathbf{x}))$$

where typically $d >> n$ solve the problem of expressing complex functions

But this introduces a

- computational problem (working with very large vectors)
  - Solved using the kernel trick – implicit feature spaces
- generalization problem (curse of dimensionality)
  - Solved by maximizing the margin of separation – first implemented in SVM (Vapnik)

We will return to SVM later

Linear Classifiers – Linear discriminant functions

- Perceptron implements a linear discriminant function – a linear decision surface given by

$$\mathrm{X} = \begin{bmatrix} x_1 \\ x_2 \\ \\ x_n \end{bmatrix} \qquad \mathrm{W} = \begin{bmatrix} w_1 \\ w_2 \\ \\ w_n \end{bmatrix}$$

$$y(x) = \mathbf{W}^T \mathbf{X} + w_0 = 0$$

The solution hyper-plane simply has to separate the classes

We can consider alternative criteria for separating hyper-planes

Least Squares for Classification

$$P(\omega_k \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid \omega_k) P(\omega_k)}{\sum_{l=1}^{M} P(\mathbf{X} \mid \omega_l) P(\omega_l)}$$

$$\left.\begin{aligned} t_k(\mathbf{X}_p) = t_{kp} &= 1 \text{ if } \mathbf{X}_p \in \omega_k \\ t_k(\mathbf{X}_p) = t_{kp} &= 0 \text{ if } \mathbf{X}_p \notin \omega_k \end{aligned}\right\} k\text{th target output}$$

$$g_k(\mathbf{X}_p; \mathbf{W}) = k\text{th output for input } \mathbf{X}_p$$

$$E_S(\mathbf{W}) = \sum_{p=1}^{P} \left(g_k(\mathbf{X}_p; \mathbf{W}) - t_{kp}\right)^2$$

$$= \sum_{\mathbf{X}_p \in \omega_k} \left(g_k(\mathbf{X}_p; \mathbf{W}) - 1\right)^2 + \sum_{\mathbf{X}_p \notin \omega_k} \left(g_k(\mathbf{X}_p; \mathbf{W}) - 0\right)^2$$

Least Squares for Classification

$$\lim_{|S|\to\infty} \frac{1}{|S|} E_S(\mathbf{W}) = P(\omega_k)\int \left(g_k(\mathbf{X};\mathbf{W})-1\right)^2 P(\mathbf{X}\mid\omega_k)d\mathbf{X} + P(\omega_{i\neq k})\int {g_k}^2(\mathbf{X};\mathbf{W})P(\mathbf{X}\mid\omega_{i\neq k})d\mathbf{X}$$

$$= \int {g_k}^2(\mathbf{X};\mathbf{W})P(\mathbf{X})d\mathbf{X} - 2\int g_k(\mathbf{X};\mathbf{W})P(\mathbf{X},\omega_k)d\mathbf{X} + \int P(\mathbf{X},\omega_k)d\mathbf{X}$$

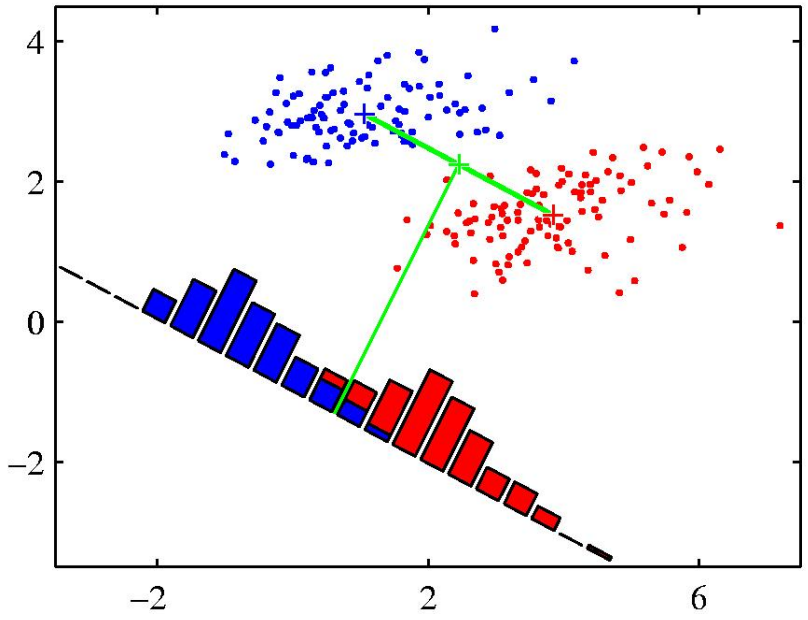
$$= \int \left(g_k(\mathbf{X};\mathbf{W}) - P(\omega_k\mid\mathbf{X})\right)^2 P(\mathbf{X})d\mathbf{X} + \underbrace{\int P(\omega_k\mid\mathbf{X})P(\omega_{i\neq k}\mid\mathbf{X})P(\mathbf{X})d\mathbf{X}}_{\text{independent of } \mathbf{W}}$$

Because least square criterion minimizes this quantity with respect to **W**, we have $g_k(\mathbf{X};\mathbf{W}) \approx P(\omega_k\mid\mathbf{X})$

assuming that the functions $g_k(\mathbf{X};\mathbf{W})$ are expressive enough to represent $P(\omega_k\mid\mathbf{X})$

Exercise: Show that Fisher discriminant is a special case of Least Squares classification

Project data onto a line joining the means of the two classes



Measure of separation of classes

- separation of the projected means

$$m_2 - m_1 = \mathbf{W}^T(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$$

Problems:

- Separation can be made arbitrarily large by increasing the magnitude of $\mathbf{W}$ – constrain $\mathbf{W}$ to be of unit length
- Classes that are well separated in the original space can have non trivial overlap in the projection –
  - Maximize between class variance in the projection

Fisher's Linear Discriminant

Given two classes, find the linear discriminant $\mathbf{W} \in \Re^n$ that maximizes Fisher's discriminant ratio:

$$f(\mathbf{W};\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_2,\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2) = \frac{(\mathbf{W}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2))^2}{\mathbf{W}^T(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)\ \mathbf{W}}$$

$$= \frac{\mathbf{W}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\mathbf{W}}{\mathbf{W}^T(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)\ \mathbf{W}}$$

Set $\dfrac{\partial f(\mathbf{W};\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_2,\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2)}{\partial \mathbf{W}} = 0$

Fisher's Linear Discriminant

$$f(\mathbf{W};\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_2,\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2) = \frac{\mathbf{W}^T(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\mathbf{W}}{\mathbf{W}^T(\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2)\ \mathbf{W}}$$

$$\frac{\partial f(\mathbf{W};\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_2,\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2)}{\partial \mathbf{W}} = \frac{\left(\mathbf{W}^T(\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2)\ \mathbf{W}\right)\frac{\partial\left(\mathbf{W}^T(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\mathbf{W}\right)}{\partial \mathbf{W}} - \left(\mathbf{W}^T(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\mathbf{W}\right)\frac{\partial}{\partial \mathbf{W}}\left(\mathbf{W}^T(\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2)\ \mathbf{W}\right)}{\left(\mathbf{W}^T(\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2)\ \mathbf{W}\right)^2}$$

$$\left(\mathbf{W}^T(\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2)\ \mathbf{W}\right)\frac{\partial\left(\mathbf{W}^T(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\mathbf{W}\right)}{\partial \mathbf{W}} - \left(\mathbf{W}^T(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\mathbf{W}\right)\frac{\partial}{\partial \mathbf{W}}\left(\mathbf{W}^T(\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2)\ \mathbf{W}\right) = 0$$

$$\left(\mathbf{W}^T(\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2)\ \mathbf{W}\right)2(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\mathbf{W} - \left(\mathbf{W}^T(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\mathbf{W}\right)2(\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2)\ \mathbf{W} = 0$$

$$(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\mathbf{W} = k(\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2)\ \mathbf{W} \quad (k = \text{const})$$

$$(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2) = k'(\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2)\ \mathbf{W} \qquad (k' = \text{const}.. \because (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\mathbf{W} \text{ has the same direction as } (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2))$$

$$\boxed{\mathbf{W}^* \propto (\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2)^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)}$$

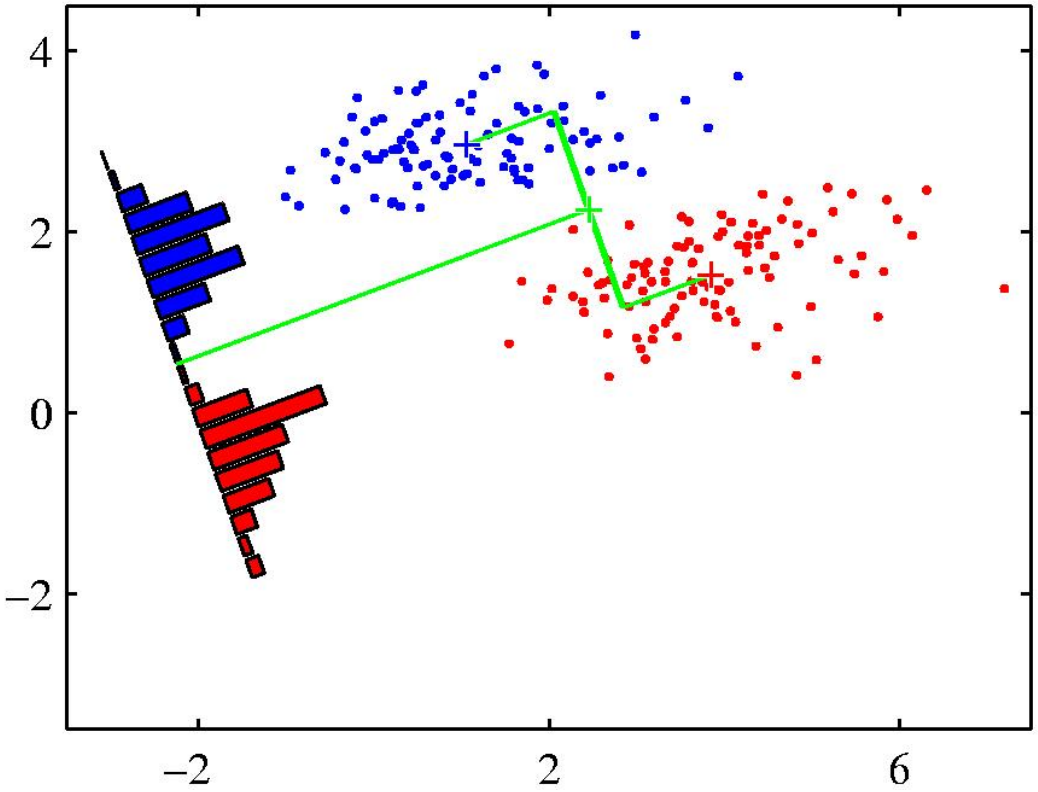
Fisher's Linear Discriminant

$\mathbf{W} \in \Re^n$ that maximizes Fisher's discriminant ratio:

$$\mathbf{W}^* = (\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

- Unique solution
- Easy to compute
- Has a probabilistic interpretation
- Can be updated incrementally as new data become available
- Naturally extends to K-class problems
- Can be generalized (using kernel trick) to handle non linearly separable class boundaries

Project data based on Fisher discriminant



Fisher's linear discriminant

- Can be shown to maximize between class separation
- If the samples in each class have Gaussian distribution, then classification using the Fisher discriminant can be shown to yield minimum error classifier
- If $\sum_1$ and $\sum_2$ are proportional to the identity matrix $\mathbf{I}$, $\mathbf{W}$ corresponding to the Fisher discriminant is proportional to the difference between the class means $\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right)$
- Can be generalized to K classes
- A special case of least squares classification (next)

Generative Versus Discriminative Models

- Generative models
  - Naïve Bayes
- Discriminative models
  - Perceptron, Support vector machines, Logistic regression ..
- Relating generative and discriminative models
- Tradeoffs between generative and discriminative models
- Generalizations and extensions

Relating Generative and Discriminative Models

Chef 1: Generative model

Note that $P(\omega_i \mid \mathbf{x}) = \dfrac{P(\mathbf{x}|\omega_i)P(\omega_i)}{P(\mathbf{x})}$

Model $P(\mathbf{x} \mid \omega_1)$, $P(\mathbf{x}|\omega_2)$, $P(\omega_1)$, and $P(\omega_2)$

Using Bayes rule, choose ω_1 if $P(\mathbf{x} \mid \omega_1)P(\omega_1) > P(\mathbf{x}|\omega_2)P(\omega_2)$

Otherwise choose ω_2

Chef 2: Discriminative Model

Model $P(\omega_1 \mid \mathbf{x}), P(\omega_2 \mid \mathbf{x})$, or the ratio $\dfrac{P(\omega_1 \mid \mathbf{x})}{P(\omega_2 \mid \mathbf{x})}$ directly

Choose ω_1 if $\dfrac{P(\omega_1 \mid \mathbf{x})}{P(\omega_2 \mid \mathbf{x})} > 1$

Otherwise choose ω_2

Review of Matrix Algebra

Basic concepts

- *Vector* in R^n is an ordered set of n real numbers.
  - e.g. v = (1,6,3,4) is in R^4
  - "(1,6,3,4)" is a column vector:
  - as opposed to a row vector:
- m-by-n *matrix* is an object with m rows and n columns, each entry filled with a real number:

$$\begin{pmatrix} 1 \\ 6 \\ 3 \\ 4 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 6 & 3 & 4 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & 8 \\ 4 & 78 & 6 \\ 9 & 3 & 2 \end{pmatrix}$$

Basic concepts

- Transpose: $\begin{pmatrix} a \\ b \end{pmatrix}^T = \begin{pmatrix} a & b \end{pmatrix}$ $\quad \begin{pmatrix} a & b \\ c & d \end{pmatrix}^T = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$

$(Ax)^T = x^T A^T$ We will define matrix multiplication shortly

L_p norm of v=$(v_1 \cdots v_k)$ is $\left(\sum_i |v_i|^p \right)^{\frac{1}{p}}$

L_1 norm of v=$(v_1 \cdots v_k)$ is $\left(\sum_i |v_i| \right)$

L_2 norm of v=$(v_1 \cdots v_k)$ is $\sqrt{\sum_i |v_i|^2}$

L_∞ norm of v=$(v_1 \cdots v_k)$ is $\max_i |v_i|$

Basic concepts

- Vector dot product:
  - Note dot product of u with itself is the square of the length of u.

$$u \bullet v = \begin{pmatrix} u_1 & u_2 \end{pmatrix} \bullet \begin{pmatrix} v_1 & v_2 \end{pmatrix} = u_1 v_1 + u_2 v_2$$

- Matrix product:

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$AB = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

Basic concepts

- Vector products:
  - Dot product:

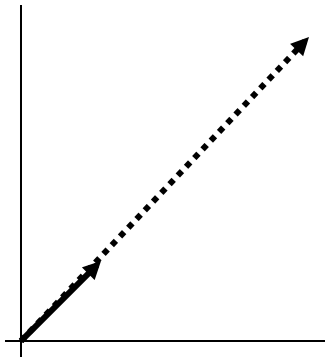
$$u \bullet v = u^T v = \begin{pmatrix} u_1 & u_2 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = u_1 v_1 + u_2 v_2$$

  - Outer product:

$$uv^T = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \begin{pmatrix} v_1 & v_2 \end{pmatrix} = \begin{pmatrix} u_1 v_1 & u_1 v_2 \\ u_2 v_1 & u_2 v_2 \end{pmatrix}$$

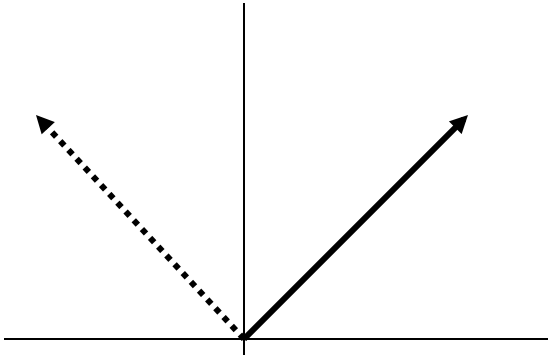
Matrices as linear transformations

$$\begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 5 \end{pmatrix}$$



(stretching)

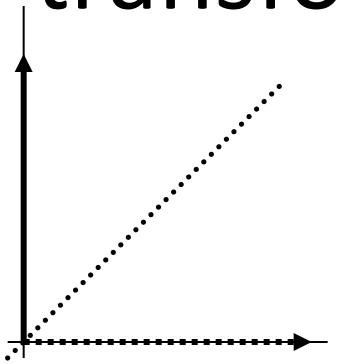
$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$



(rotation)

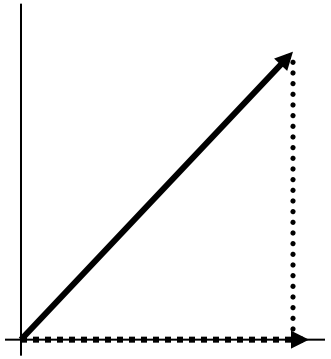
Matrices as linear transformations

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$



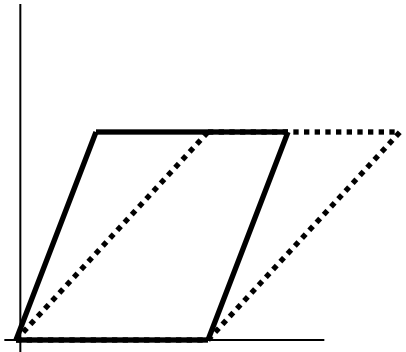
(reflection)

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$



(projection)

$$\begin{pmatrix} 1 & c \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x + cy \\ y \end{pmatrix}$$



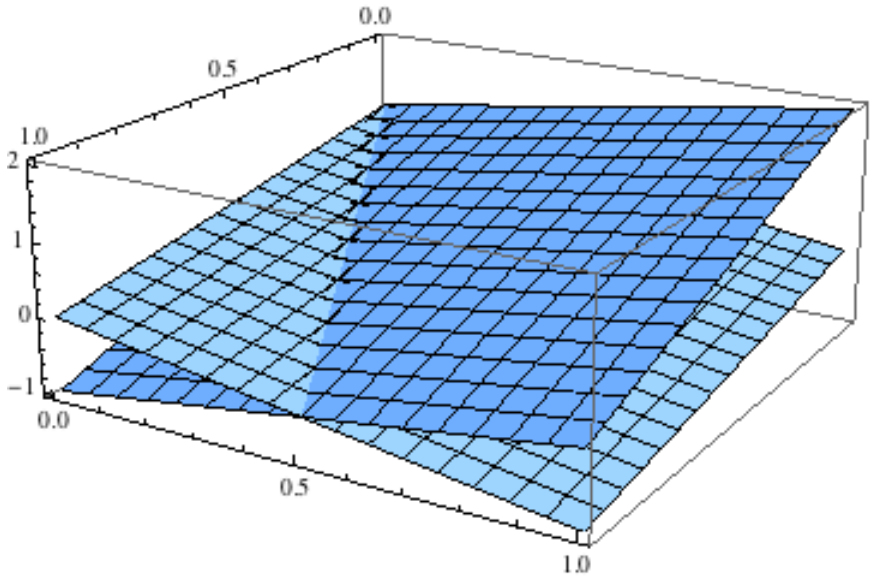
(shearing)

Using matrices to express as sets of (linear) constraints

$$x + y + z = 1$$

$$2x - y + z = 2$$

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & -1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$



Special matrices

$$\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix}$$ *diagonal*

$$\begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix}$$ *upper-triangular*

$$\begin{pmatrix} a & b & 0 & 0 \\ c & d & e & 0 \\ 0 & f & g & h \\ 0 & 0 & i & j \end{pmatrix}$$ *tri-diagonal*

$$\begin{pmatrix} a & 0 & 0 \\ b & c & 0 \\ d & e & f \end{pmatrix}$$ *lower-triangular*

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$ *I (identity matrix)*

Matrix inversion

- To solve Ax=b, we can write a closed-form solution if we can find a matrix A^{-1}

  s.t. $AA^{-1} = A^{-1}A = I$ (identity matrix)
- Then Ax=b iff $x=A^{-1}b$:

  $x = Ix = A^{-1}Ax = A^{-1}b$
- A is *non-singular* iff A^{-1} exists iff Ax=b has a unique solution.
- Note: If A^{-1}, B^{-1} exist, then $(AB)^{-1} = B^{-1}A^{-1}$,

  and $(A^T)^{-1} = (A^{-1})^T$

Special matrices

- Matrix A is *symmetric* if $A = A^T$
- A is *positive definite* if $x^TAx>0$ for all non-zero x (*positive semi-definite* if inequality is not strict)

$$\begin{pmatrix} a & b & c \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = a^2 + b^2 + c^2$$

$$\begin{pmatrix} a & b & c \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = a^2 - b^2 + c^2$$

Special matrices

- Matrix A is *symmetric* if $A = A^T$
- A is *positive definite* if $x^TAx>0$ for all non-zero x (*positive semi-definite* if inequality is not strict)

- Useful fact: Any matrix of form A^TA is positive semi-definite.

To see this, $x^T(A^TA)x = (x^TA^T)(Ax) = (Ax)^T(Ax) \geq 0$

Determinants

- If det(A) = 0, then A is singular.
- If det(A) ≠ 0, then A is invertible.
- To compute:
  - Simple example:

$$\det\begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$$

  - Matlab: det(A)

Determinants

- m-by-n matrix A is *rank-deficient* if it has rank r < m (≤ n)
- Thm: rank(A) < r iff

  det(A) = 0 for all t-by-t submatrices,

  r ≤ t ≤ m

Eigenvalues & eigenvectors

- How can we characterize matrices?
- The solutions to $Ax = \lambda x$ in the form of eigenpairs (λ, x) = (eigenvalue,eigenvector) where x is non-zero
- To solve this, $(A - \lambda I)x = 0$
- λ is an eigenvalue iff $\det(A - \lambda I) = 0$

Eigenvalues & eigenvectors

$(A - \lambda I)x = 0$

λ is an eigenvalue iff $\det(A - \lambda I) = 0$

Example:

$$A = \begin{pmatrix} 1 & 4 & 5 \\ 0 & 3/4 & 6 \\ 0 & 0 & 1/2 \end{pmatrix}$$

$$\det(A - \lambda I) = \begin{pmatrix} 1-\lambda & 4 & 5 \\ 0 & 3/4-\lambda & 6 \\ 0 & 0 & 1/2-\lambda \end{pmatrix} = (1-\lambda)(3/4-\lambda)(1/2-\lambda)$$

$$\lambda = 1, \lambda = 3/4, \lambda = 1/2$$

From generative to discriminative models

- Assume classes are binary $y \in \{0,1\}$
- Suppose we model the class by a binomial distribution with parameter q

$$p(y \mid q) = q^{y}(1-q)^{(1-y)}$$

- Assume each component X_j of input $\mathbf{X}$ each have Gaussian distributions with parameters Θ_j and are independent given the class

$$p(x, y \mid \Theta) = P(y \mid q)\prod_{j=1}^{n} p(x_j \mid y, \theta_j)$$

$$\text{where } \Theta = (q, \theta_1, \ldots . \theta_n)$$

From generative to discriminative models

$$p\left(x_j \mid y=0,\Theta_j\right)=\frac{1}{\left(2\pi\sigma_j^2\right)^{1/2}}\exp\left\{-\frac{1}{2\sigma_j^2}\left(x_j-\mu_{0j}\right)^2\right\}$$

$$p\left(x_j \mid y=1,\Theta_j\right)=\frac{1}{\left(2\pi\sigma_j^2\right)^{1/2}}\exp\left\{-\frac{1}{2\sigma_j^2}\left(x_j-\mu_{1j}\right)^2\right\}$$

where $\Theta_j = \left(\mu_{0j},\mu_{1j},\sigma_j\right)$

(Note:we have assumed that $\forall j\ \sigma_{0j}=\sigma_{1j}=\sigma_j$)

From generative to discriminative models

The calculation of the posterior probability $p(Y=1|x, \Theta)$ is simplified if we use matrix notation

$$p(x|y=1,\Theta) = \prod_{j=1}^{j=n} \left(\frac{1}{(2\pi)^{1/2}\sigma_j} \exp\left\{ -\frac{1}{2}\left(\frac{x_1 - \mu_{1j}}{\sigma_j} \right)^2 \right\} \right)$$

$$= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left\{ -\frac{1}{2}(x-\mu_1)^T \Sigma^{-1}(x-\mu_1) \right\}$$

where $\mu_1 = (\mu_{11}....\mu_{1n})^T$; and $\Sigma = diag(\sigma_1^2....\sigma_n^2) = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & . & 0 \\ 0 & 0 & \sigma_n^2 \end{bmatrix}$

From generative to discriminative models

$$p(y=1 \mid x,\Theta) = \frac{p(x \mid y=1,\Theta)p(y=1 \mid q)}{p(x \mid y=1,\Theta)p(y=1 \mid q) + p(x \mid y=0,\Theta)p(y=0 \mid q)}$$

$$= \frac{q \exp\left\{-\frac{1}{2}(x-\mu_1)^T \Sigma^{-1}(x-\mu_1)\right\}}{q \exp\left\{-\frac{1}{2}(x-\mu_1)^T \Sigma^{-1}(x-\mu_1)\right\} + (1-q)\exp\left\{-\frac{1}{2}(x-\mu_0)^T \Sigma^{-1}(x-\mu_0)\right\}}$$

$$= \frac{1}{1+\exp\left\{-\log\left(\frac{q}{1-q}\right) + \frac{1}{2}(x-\mu_1)^T \Sigma^{-1}(x-\mu_1) - \frac{1}{2}(x-\mu_0)^T \Sigma^{-1}(x-\mu_0)\right\}}$$

$$= \frac{1}{1+\exp\left\{-\underbrace{(\mu_1-\mu_0)^T \Sigma^{-1}}_{\beta^T} x + \underbrace{\frac{1}{2}(\mu_1-\mu_0)^T \Sigma^{-1}(\mu_1+\mu_0) - \log\left(\frac{q}{1-q}\right)}_{-\gamma}\right\}}$$

$$= \frac{1}{1+\exp\left(-\beta^T x - \gamma\right)}$$

where we have used $A^T DA - B^T DB = (A+B)^T D(A-B)$ for a symmetric matrix D

From generative to discriminative models

$$p(y=1 \mid x, \Theta) = \frac{1}{1+\exp\left(-\beta^T x - \gamma\right)}$$

The posterior probability that Y=1 takes the form

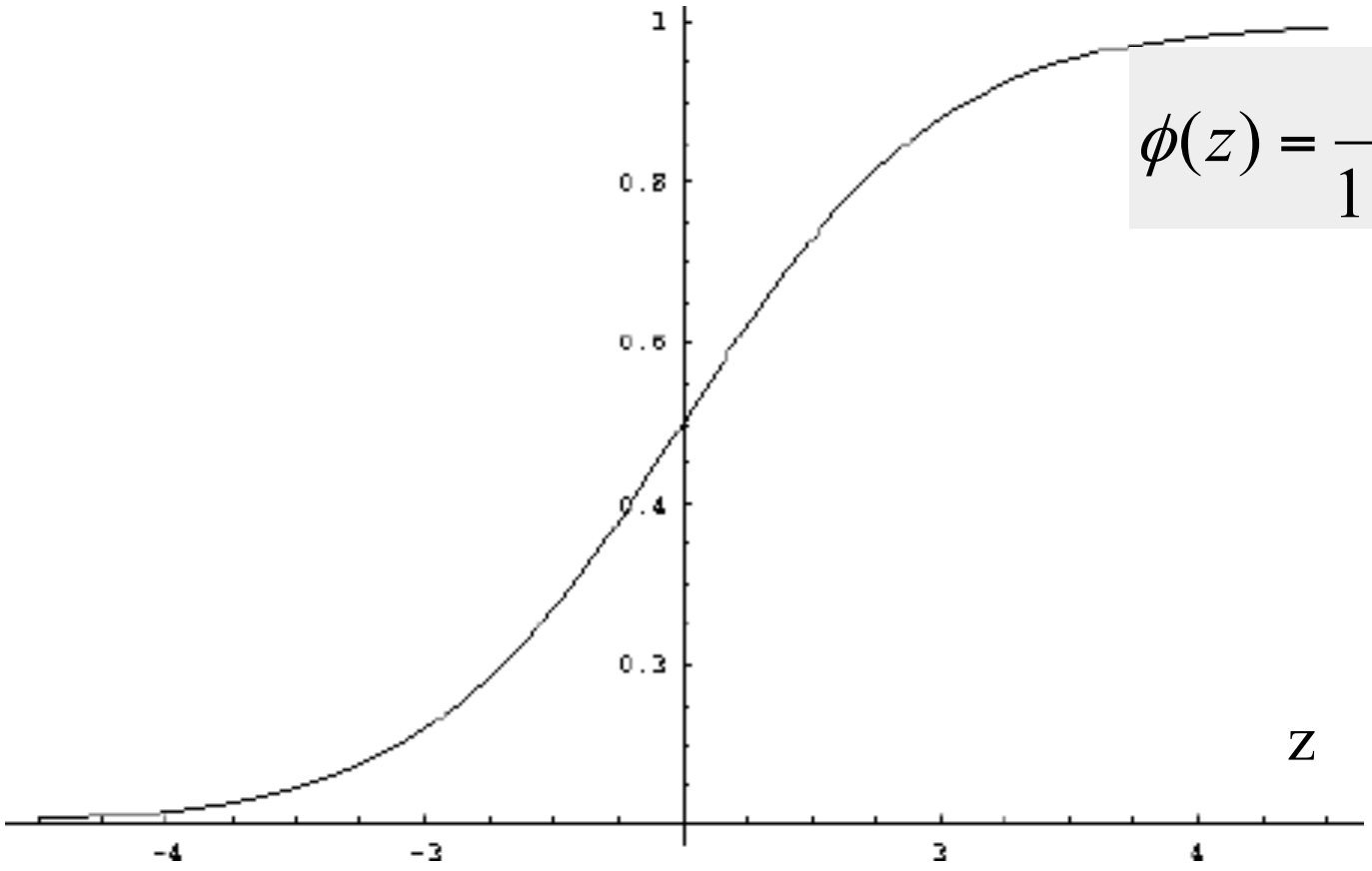
where $$\phi(z) = \frac{1}{1+e^{-z}}$$

$$z = \beta^T x + \gamma$$
$$= \beta \bullet x + \gamma$$

is an affine function of x

Sigmoid or Logistic Function

$$\phi(z)$$

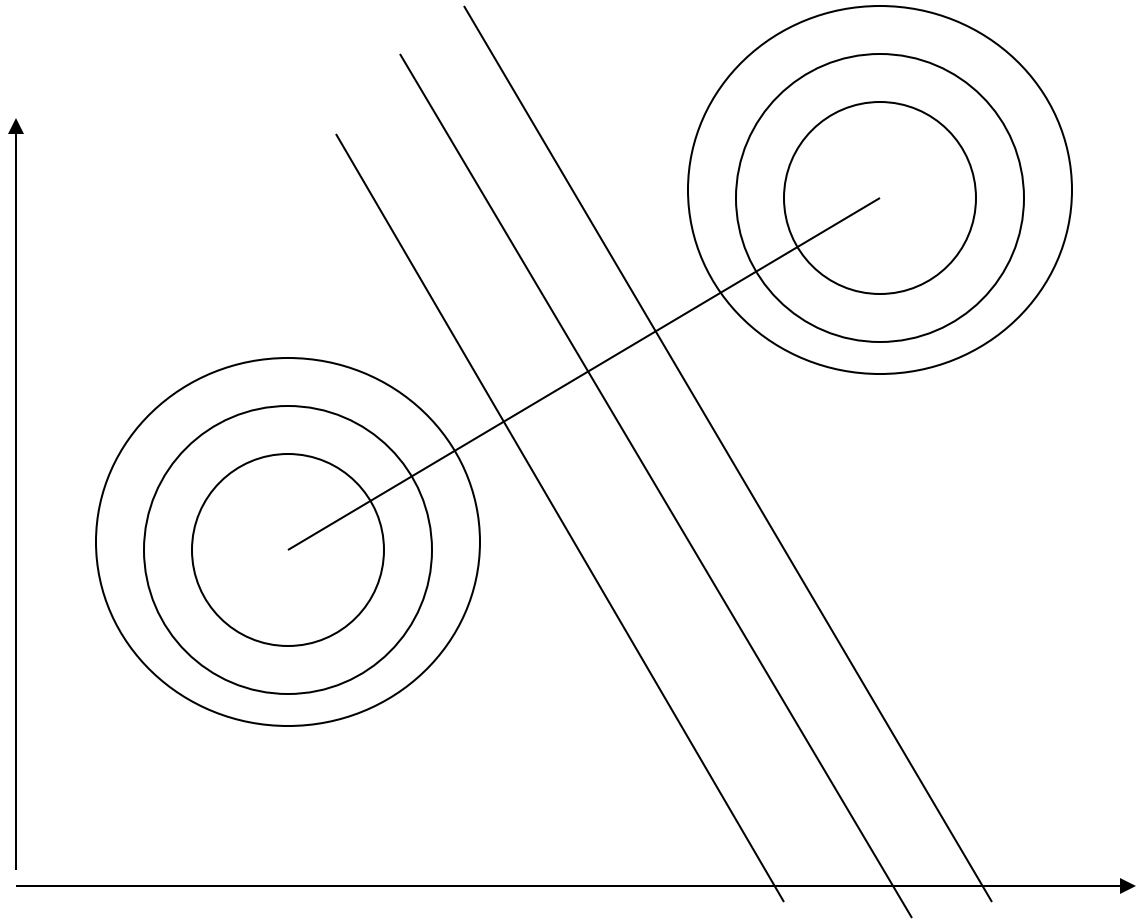


$$\phi(z) = \frac{1}{1+e^{-z}}$$

Implications of the logistic posterior

- Posterior probability of Y is a logistic function of an affine function of $\mathbf{x}$
- Contours of equal posterior probability are lines in the input space
- $\beta^T\mathbf{x}$ is proportional to the projection of $\mathbf{x}$ on β and this projection is equal for all vectors $\mathbf{x}$ that lie along a line that is orthogonal to β
- Special case
  - variances of Gaussians = 1
  - the contours of equal posterior probability are lines that are orthogonal to the difference vector between the means of the two classes
- Equal posterior for the two classes when $z=0$

Geometric interpretation (diagonal $\sum$)
Contour plot



Geometric interpretation

$$p(y=1 \mid x,\Theta) = \frac{p(x \mid y=1,\Theta)p(y=1 \mid q)}{p(x \mid y=1,\Theta)p(y=1 \mid q) + p(x \mid y=0,\Theta)p(y=0 \mid q)}$$

$$= \frac{1}{1+\exp\left\{-\underbrace{(\mu_1-\mu_0)^T\Sigma^{-1}}_{\beta^T}x + \underbrace{\frac{1}{2}(\mu_1-\mu_0)^T\Sigma^{-1}(\mu_1+\mu_0) - \log\left(\frac{q}{1-q}\right)}_{-\gamma}\right\}}$$

$$= \frac{1}{1+\exp(-\beta^T x-\gamma)} = \frac{1}{1+e^{-z}}$$

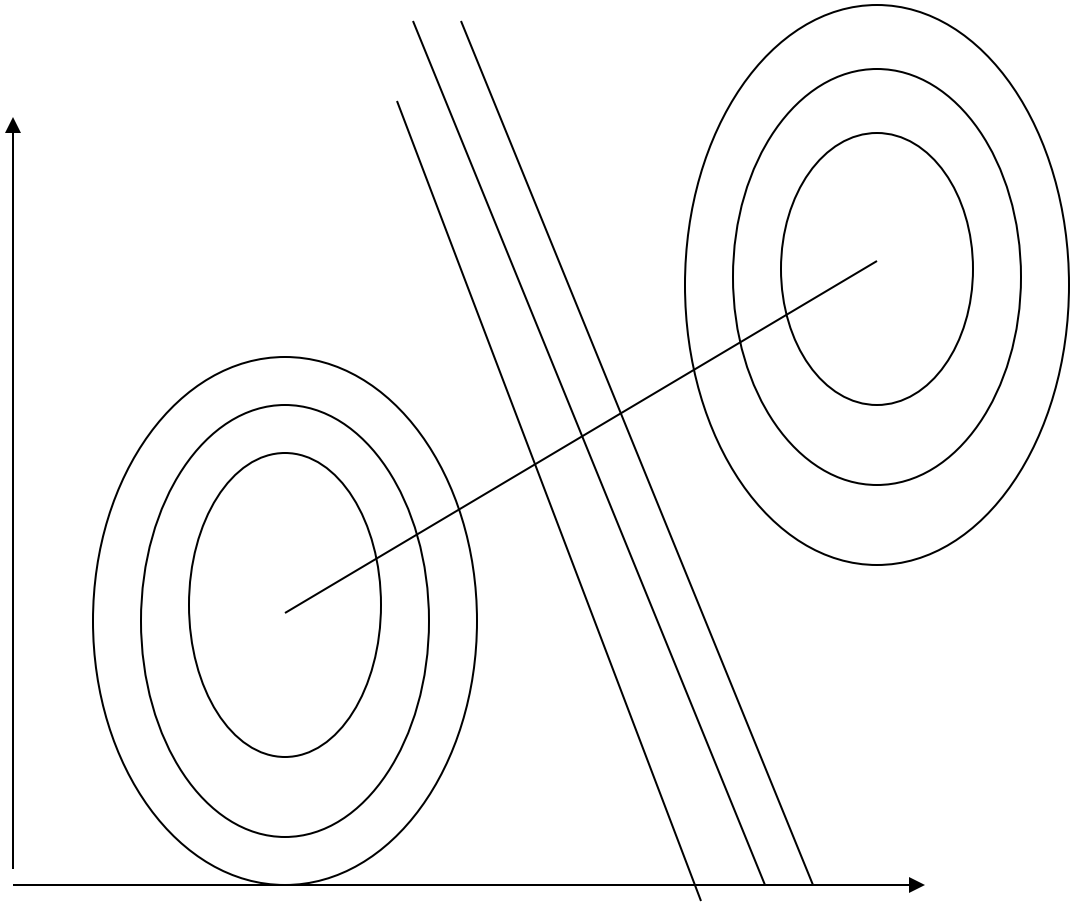
when $q = 1\text{-}q$, $z = (\mu_1-\mu_0)^T\Sigma^{-1}\left(x - \frac{(\mu_1+\mu_0)}{2}\right)$

In this case, the posterior probabilities for the two classes are equal when x is equidistant from the two means

Geometric interpretation

- If the prior probabilities of the classes are such that $q > 0.5$ the effect is to shift the logistic function to the left resulting in a larger value for the posterior probability for Y=1 for any given point in the input space.
- $q < 0.5$ results in a shift of the logistic function to the right resulting in a smaller value for the posterior probabilty for Y=1 (or larger value for the posterior probability for Y=0)

Geometric interpretation (general $\sum$)



Now the equi-probability contours are still lines in the input space although the lines are no longer orthogonal to the difference in means of the two classes

Generalization to multiple classes – Softmax function

- *Y* is a multinomial variable which takes on one of *K* values

$$q_k = p(y = k \mid q) = p(y^k = 1 \mid q)$$
$$\text{where } (y = k) \equiv (y^k = 1)$$
$$q = (q_1 \quad q_2 \quad \cdots \quad q_K)$$

- As before, $\mathbf{x}$ is a multivariate Gaussian

$$p(\mathbf{x} \mid y_k = 1, \Theta) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \right\}$$

$$\text{where } \boldsymbol{\mu}_k = (\mu_{k1} \ldots \mu_{kn})^T; \text{ and } \forall k \ \boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$$

(covariance matrix is assumed to be same for each class)

Generalization to multiple classes – Softmax function

Posterior probability for class k is obtained via Bayes rule

$$p\left(y^k = 1 \mid \mathbf{x}, \Theta\right) = \frac{p\left(\mathbf{x} \mid y^k = 1, \Theta\right) p\left(y^k = 1 \mid q\right)}{\sum_{l=1}^{K} p\left(\mathbf{x} \mid y^k = 1, \Theta\right) p\left(y^k = 1 \mid q\right)}$$

$$= \frac{q_k \exp\left\{-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}_k\right)^T \boldsymbol{\Sigma}^{-1}\left(\mathbf{x}-\boldsymbol{\mu}_k\right)\right\}}{\sum_{l=1}^{K} q_l \exp\left\{-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}_l\right)^T \boldsymbol{\Sigma}^{-1}\left(\mathbf{x}-\boldsymbol{\mu}_l\right)\right\}}$$

$$= \frac{\exp\left\{\boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log q_k\right\}}{\sum_{l=1}^{K} \exp\left\{\boldsymbol{\mu}_l^T \boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_l^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_l + \log q_l\right\}}$$

Generalization to multiple classes – Softmax function

We have shown that

$$p\left(y^k = 1 \mid \mathbf{x}, \Theta\right) = \frac{\exp\left\{\boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log q_k\right\}}{\sum_{l=1}^{K} \exp\left\{\boldsymbol{\mu}_l^T \boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_l^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_l + \log q_l\right\}}$$

Defining parameter vectors and augmenting the input Vector $\mathbf{x}$ by adding a constant input of 1 we have

$$\boldsymbol{\beta}_k = \begin{bmatrix} -\frac{1}{2}\boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log q_k \\ \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k \end{bmatrix}$$

$$p\left(y^k = 1 \mid \mathbf{x}, \Theta\right) = \frac{e^{\boldsymbol{\beta}_k^T \mathbf{x}}}{\sum_{l=1}^{K} e^{\boldsymbol{\beta}_l^T \mathbf{x}}} = \frac{e^{\langle \boldsymbol{\beta}_k, \mathbf{x} \rangle}}{\sum_{l=1}^{K} e^{\langle \boldsymbol{\beta}_l, \mathbf{x} \rangle}}$$

Generalization to multiple classes – Softmax function

$$p\left(y^k = 1 \mid \mathbf{x}, \Theta\right) = \frac{e^{\boldsymbol{\beta}_k^T \mathbf{x}}}{\sum_{l=1}^{K} e^{\boldsymbol{\beta}_l^T \mathbf{x}}} = \frac{e^{\langle \boldsymbol{\beta}_k, \mathbf{x} \rangle}}{\sum_{l=1}^{K} e^{\langle \boldsymbol{\beta}_l, \mathbf{x} \rangle}}$$

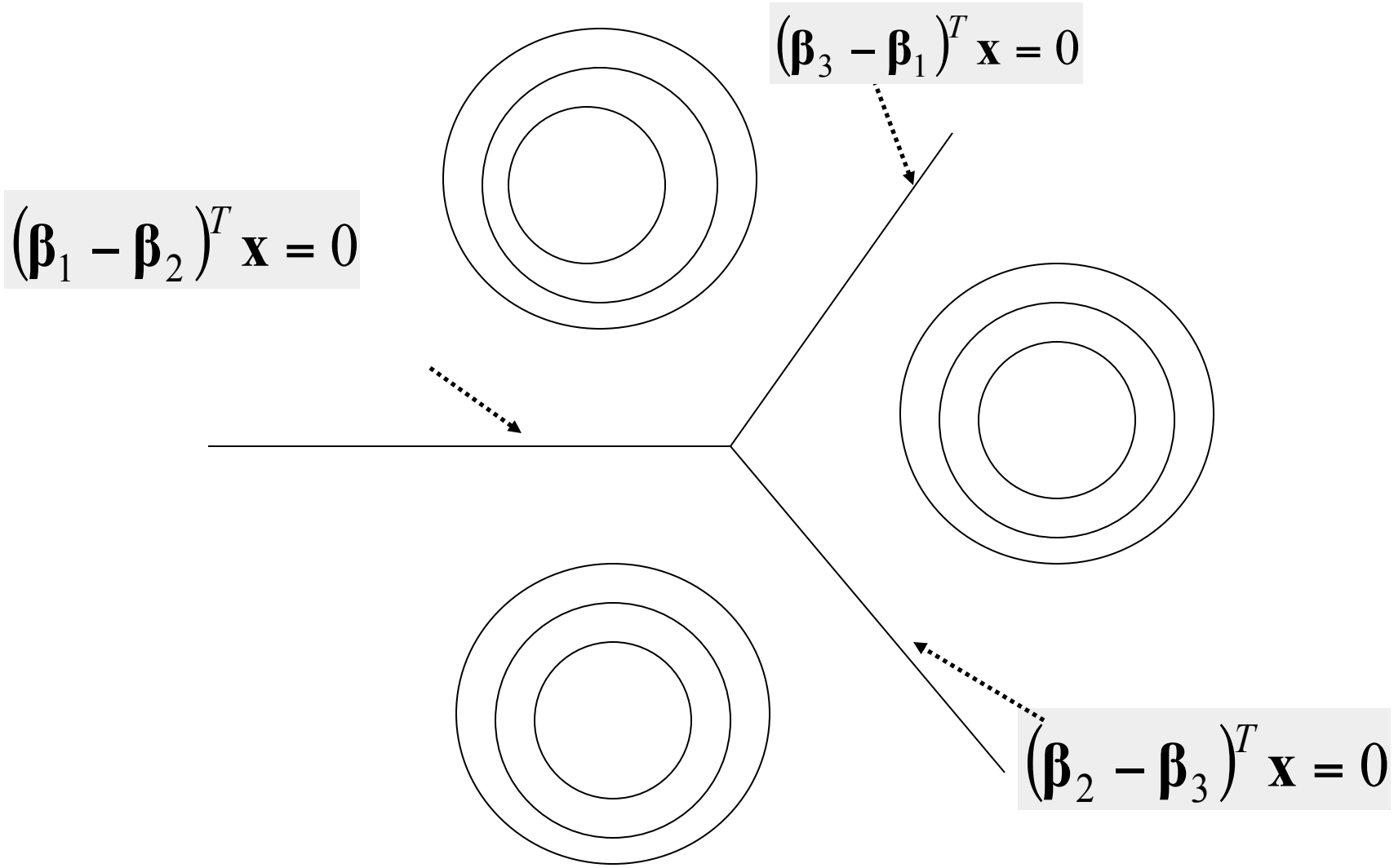
corresponds to the decision rule:

$$h(\mathbf{x}) = \underset{j}{\text{argmax}}\ p\left(y^k = 1 \mid \mathbf{x}, \Theta\right) = \underset{j}{\text{argmax}}\ e^{\langle \boldsymbol{\beta}_j, \mathbf{x} \rangle} = \underset{j}{\text{argmax}}\ \langle \boldsymbol{\beta}_j, \mathbf{x} \rangle$$

Consider the ratio of posterior prob. for classes k and $j \neq k$

$$\frac{p\left(y^k = 1 \mid \mathbf{x}, \Theta\right)}{p\left(y^j = 1 \mid \mathbf{x}, \Theta\right)} = \frac{e^{\langle \boldsymbol{\beta}_k, \mathbf{x} \rangle}}{\sum_{l=1}^{K} e^{\langle \boldsymbol{\beta}_l, \mathbf{x} \rangle}} \frac{\sum_{l=1}^{K} e^{\langle \boldsymbol{\beta}_l, \mathbf{x} \rangle}}{e^{\langle \boldsymbol{\beta}_j, \mathbf{x} \rangle}} = \frac{e^{\langle \boldsymbol{\beta}_k, \mathbf{x} \rangle}}{e^{\langle \boldsymbol{\beta}_j, \mathbf{x} \rangle}} = e^{\langle (\boldsymbol{\beta}_k - \boldsymbol{\beta}_j), \mathbf{x} \rangle}$$

Equi-probability contours of the softmax function



Naïve Bayes classifier with discrete attributes and K classes

q_k= prior probability of class k

η_{kji}= probability that x_j (the jth component of $\mathbf{x}$) takes the ith value in its domain when $\mathbf{x}$ belongs to class k.

$$p(\mathbf{x}, \mathbf{y} \mid \Theta) = p(\mathbf{y} \mid q) \prod_{j=1}^{n} p(x_j \mid \mathbf{y}, \theta_j)$$

$$q_k = p(y^k = 1 \mid q);$$

$$\eta_{kji} = p(x_j^i = 1 \mid y^k = 1, \eta)$$

$$p(y^k = 1 \mid \mathbf{x}, \eta) = \frac{q_k \prod_j \prod_i (\eta_{kji})^{x_j^i}}{\sum_l q_l \prod_j \prod_i (\eta_{kji})^{x_j^i}}$$

Naïve Bayes classifier with discrete attributes and K classes

$$q_k = p\left(y^k = 1 \mid q\right); \quad \eta_{kji} = p\left(x_j^i = 1 \mid y^k = 1, \eta\right)$$

$$p\left(y^k = 1 \mid \mathbf{x}, \eta\right) = \frac{q_k \prod_j \prod_i \left(\eta_{kji}\right)^{x_j^i}}{\sum_l q_l \prod_j \prod_i \left(\eta_{kji}\right)^{x_j^i}}$$

$$= \frac{\exp\left\{\log q_k + \sum_{j=1}^{n} \sum_{i=1}^{N_j} x_j^i \log \eta_{kji}\right\}}{\sum_{l=1}^{K} \exp\left\{\log q_l + \sum_{j=1}^{n} \sum_{i=1}^{N_j} x_j^i \log \eta_{lji}\right\}}$$

$$= \frac{e^{\boldsymbol{\beta}_k^T \mathbf{x}}}{\sum_{l=1}^{K} e^{\boldsymbol{\beta}_l^T \mathbf{x}}} = \frac{e^{\langle \boldsymbol{\beta}_k, \mathbf{x} \rangle}}{\sum_{l=1}^{K} e^{\langle \boldsymbol{\beta}_l, \mathbf{x} \rangle}}$$

From generative to discriminative models

- A curious fact about all of the generative models we have considered so far is that
  - The posterior probability of class can be expressed in the form of a logistic function in the case of a binary classifier and a softmax function in the case of a K-class classifier

From generative to discriminative models

- For multinomial and Gaussian class conditional densities (in the case of the latter, with equal but otherwise arbitrary covariance matrices)
  - the contours of equal posterior probabilities of classes are hyperplanes in the input (feature) space.
- The result is a simple linear classifier analogous to the perceptron (for binary classification) or winner-take-all network (for K-ary classification)
- Next, we see that these results hold for a more general class of distributions

Digression: The exponential family of distributions

The exponential family is specified by

$$p(\mathbf{x} \mid \boldsymbol{\eta}) = h(\mathbf{x}) e^{\left\{\boldsymbol{\eta}^T G(\mathbf{x}) - A(\boldsymbol{\eta})\right\}}$$

where $\boldsymbol{\eta}$ is a parameter vector and $A(\boldsymbol{\eta})$, $h(\mathbf{x})$ and $G(\mathbf{x})$ are appropriately chosen functions.

- Gaussian, Binomial, and multinomial (and many other "textbook") distributions belong to the exponential family
- Likelihood function for exponential family is provably convex
- Maximum entropy estimate of unknown probability distributions under moment constraints yields an exponential form

The Bernoulli distribution belongs to the exponential family

$$p(\mathbf{x} \mid \boldsymbol{\eta}) = h(\mathbf{x}) e^{\left\{\boldsymbol{\eta}^T G(\mathbf{x}) - A(\boldsymbol{\eta})\right\}}$$

Bernoulli distribution with success rate q is given by

$$p(x \mid q) = q^x (1-q)^{1-x} = \exp\left\{\log\left(\frac{q}{1-q}\right)x + \log(1-q)\right\}$$

We can see that Bernoulli distribution belongs to the exponential family by choosing

$$\eta = \log\left(\frac{q}{1-q}\right);\ G(x) = x;\ h(x) = 1$$
$$A(\eta) = -\log(1-q) = \log(1+e^{\eta})$$

The Gaussian distribution belongs to the exponential family

$$p(\mathbf{x} \mid \boldsymbol{\eta}) = h(\mathbf{x}) e^{\{\boldsymbol{\eta}^T G(\mathbf{x}) - A(\boldsymbol{\eta})\}}$$

Univariate Gaussian distribution can be written as

$$p(x \mid \mu, \sigma^2) = \frac{1}{(2\pi)^{1/2}\sigma} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

$$= \frac{1}{(2\pi)^{1/2}} \exp\left\{\frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}x^2 - \frac{1}{2\sigma^2}\mu^2 - \ln\sigma\right\}$$

We see that Gaussian distribution belongs to the exponential family by choosing

$$\eta = \begin{bmatrix} \mu/\sigma^2 \\ -1/2\sigma^2 \end{bmatrix}; \quad A(\eta) = \frac{\mu^2}{2\sigma^2} + \ln\sigma$$

$$G(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}; \quad h(x) = \frac{1}{(2\pi)^{1/2}}$$

The exponential family

The exponential family which is given by

$$p(\mathbf{x} \mid \boldsymbol{\eta}) = h(\mathbf{x}) e^{\left\{\boldsymbol{\eta}^T G(\mathbf{x}) - A(\boldsymbol{\eta})\right\}}$$

where $\boldsymbol{\eta}$ is a parameter vector and $A(\boldsymbol{\eta})$, $h(\mathbf{x})$ and $G(\mathbf{x})$ are appropriately chosen functions – can be shown to include several additional distributions such as the multinomial, the Poisson, the Gamma, the Dirichlet, among others.

Exercise: Show that the multinomial distribution belongs to the exponential family.

From generative to discriminative models

- In the case of the generative models we have seen
- The posterior probability of class can be expressed in the form of a logistic function in the case of a binary classifier and a softmax function in the case of a K-class classifier
- The contours of equal posterior probabilities of classes are hyperplanes in the input (feature) space yielding a linear classifier for binary classification) or winner-take-all network (for K-ary classification).

From generative to discriminative models

- We just showed that the probability distributions underlying the generative models considered belong to the exponential family
- What can we say about the classifiers when the underlying generative models are distributions from the exponential family?

Classification problem for generic class conditional density from the exponential family $p(\mathbf{x} \mid \boldsymbol{\eta}) = h(\mathbf{x}) e^{\{\boldsymbol{\eta}^T G(\mathbf{x}) - A(\boldsymbol{\eta})\}}$

Consider Binary classification task with density for class 0 and class 1 parameterized by η_0 and η_1. Further assume $G(\mathbf{x})$ is a linear function of $\mathbf{x}$ (before augmenting $\mathbf{x}$ with a 1)

$$p(y=1 \mid \mathbf{x}, \boldsymbol{\eta}) = \frac{p(\mathbf{x} \mid y=1, \boldsymbol{\eta}) p(y=1 \mid q)}{p(\mathbf{x} \mid y=1, \boldsymbol{\eta}) p(y=1 \mid q) + p(\mathbf{x} \mid y=0, \boldsymbol{\eta}) p(y=0 \mid q)}$$

$$= \frac{\exp\left\{\boldsymbol{\eta}_1^T G(\mathbf{x}) - A(\boldsymbol{\eta}_1)\right\} h(\mathbf{x}) q_1}{\exp\left\{\boldsymbol{\eta}_1^T G(\mathbf{x}) - A(\boldsymbol{\eta}_1)\right\} h(\mathbf{x}) q_1 + \exp\left\{\boldsymbol{\eta}_0^T G(\mathbf{x}) - A(\boldsymbol{\eta}_0)\right\} h(\mathbf{x}) q_0}$$

$$p(y=1 \mid \mathbf{x}, \boldsymbol{\eta}) = \frac{1}{1 + \exp\left\{-(\boldsymbol{\eta}_0 - \boldsymbol{\eta}_1)^T G(\mathbf{x}) - A(\boldsymbol{\eta}_0) + A(\boldsymbol{\eta}_1) + \log \frac{q_0}{q_1}\right\}}$$

Note that this is a logistic function of a linear function of $\mathbf{x}$

Classification problem for generic class conditional density from the exponential family

Consider K-ary classification task; Suppose $G(\mathbf{x})$ is a linear function of $\mathbf{x}$

$$p(\mathbf{x} \mid \boldsymbol{\eta}) = h(\mathbf{x}) e^{\left\{\boldsymbol{\eta}^T G(\mathbf{x}) - A(\boldsymbol{\eta})\right\}}$$

$$p(y^k = 1 \mid \mathbf{x}, \boldsymbol{\eta}) = \frac{\exp\left\{\boldsymbol{\eta}_k{}^T G(\mathbf{x}) - A(\boldsymbol{\eta}_k)\right\} q_k}{\sum_{l=1}^{K} \exp\left\{\boldsymbol{\eta}_l{}^T G(\mathbf{x}) - A(\boldsymbol{\eta}_l)\right\} q_l}$$

$$= \frac{\exp\left\{\boldsymbol{\eta}_k{}^T G(\mathbf{x}) - A(\boldsymbol{\eta}_k) + \log q_k\right\}}{\sum_{l=1}^{K} \exp\left\{\boldsymbol{\eta}_l{}^T G(\mathbf{x}) - A(\boldsymbol{\eta}_l) + \log q_l\right\}}$$

which is a softmax function of a linear function of $\mathbf{x}$!!

Summary

- A variety of class conditional densities all yield the same logistic-linear or softmax-linear (with respect to parameters) form for the posterior probability
- In practice, choosing a class conditional density can be difficult – especially in high dimensional spaces – e.g., multi-variate Gaussian where the covariance matrix grows quadratically in the number of dimensions!
- The invariance of the functional form of the posterior probability with respect to the choice of the distribution is good news!
- It is not necessary to specify the class conditional density at all if we can work directly with the posterior – which brings us to discriminative models!

Discriminative Models

- We saw that under fairly general assumptions concerning the underlying generative model, the posterior probability of class given **x** can be expressed in the form of a logistic function of an affine or polynomial (in the simplest case, linear) function of **x** in the case of a binary classification task.

$$P(y=1 \mid \mathbf{x}) = \frac{1}{1+e^{-\langle \mathbf{w}, G(\mathbf{x}) \rangle}} = \frac{1}{1+e^{-\eta(\mathbf{x})}} = \mu(\mathbf{x})$$

where $\eta(\mathbf{x}) = \mathbf{w}^T G(\mathbf{x}) = \langle \mathbf{w}, G(\mathbf{x}) \rangle$

- In the discriminative setting, we simply assume this form and proceed without regard to details of the underlying generative model

Discriminative Models

Note that the posterior probability of $Y=1$ is same as the conditional expectation of y given $\mathbf{x}$:

$$E(y \mid \mathbf{x}) = 1 \cdot P(y=1 \mid \mathbf{x}) + 0 \cdot P(y=0 \mid \mathbf{x})$$
$$= P(y=1 \mid \mathbf{x}) = \mu(\mathbf{x}) = (\mu(\mathbf{x}))^{y} (1-\mu(\mathbf{x}))^{1-y}$$

where

$$\mu(\mathbf{x}) = \frac{1}{1+e^{-\eta(\mathbf{x})}} = \frac{1}{1+e^{-\mathbf{w}^T G(\mathbf{x})}}$$

Hence estimating $\mathrm{P}(Y=1|\mathbf{x})$ is equivalent to performing logistic regression

Some Properties of the Logistic Function

$$\mu = \frac{1}{1+e^{-\eta}}; \quad \eta = \log\left(\frac{\mu}{1-\mu}\right)$$

$$\frac{d\eta}{d\mu} = \frac{d}{d\mu}\log\left(\frac{\mu}{1-\mu}\right) = \left(\frac{1-\mu}{\mu}\right)\frac{d}{d\mu}\left(\frac{\mu}{1-\mu}\right) = \left(\frac{1-\mu}{\mu}\right)\left(\frac{(1-\mu)\frac{d}{d\mu}(\mu) - \mu\frac{d}{d\mu}(1-\mu)}{(1-\mu)^2}\right) = \frac{1}{\mu(1-\mu)}$$

$$\frac{d\mu}{d\eta} = \mu(1-\mu)$$

Maximum likelihood estimation of $\mathbf{w}$

$$D = \{(x_n, y_n); X_n \in Domain(\mathbf{x}); \quad y_n \in \{0,1\}; \; n = 1..N\}$$

$$\eta_n = \mathbf{w}^T x_n; \quad \mu_n = \frac{1}{1+e^{-\eta_n}}$$

Likelihood $$P(y_1 \ldots y_N \mid x_1 \ldots x_N, \mathbf{w}) = \prod_{n=1}^{N} (\mu_n)^{y_n} (1-\mu_n)^{(1-y_n)}$$

Log likelihood

$$LL(\mathbf{w} : D) = \sum_{n=1}^{N} \{y_n \log \mu_n + (1-y_n) \log(1-\mu_n)\}$$

We need to find $\mathbf{w}$ that maximizes log likelihood

Digression – Minimizing / Maximizing Functions

Consider $f(x)$, a function of a scalar variable x with domain D_x

$f(x)$ is convex over some sub-domain $D \subseteq D_x$ if $\forall X_1, X_2 \in D$,
the chord joining the points $f(X_1)$ and $f(X_2)$ lies above
the graph of $f(x)$

$f(x)$ has a local minimum at $x = X_a$ if $\exists$ neighborhood $U \subseteq D_x$ around X_a such that $\forall x \in U, f(x) > f(X_a)$

We say that $\lim_{x \to a} f(x) = A$ if, for any $\varepsilon > 0$, $\exists \delta > 0$ such that $|f(x) - A| < \varepsilon$
$\forall x$ such that $|x-a| < \delta$

Minimizing/Maximizing Functions

We say that $f(x)$ is continuous at $x = a$ if $\lim_{\varepsilon \to 0} \left\{ \lim_{x \to a + \in} f(x) \right\} = \lim_{\varepsilon \to 0} \left\{ \lim_{x \to a - \in} f(x) \right\}$

The derivative of the function $f(x)$ is defined as

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{(\Delta x)}$$

$$\left. \frac{df}{dx} \right|_{x = X_0} = 0 \text{ if } X_0 \text{ is a local maximum or a local minimum}$$

Minimizing/Maximizing Functions

$$\frac{d(u+v)}{dx} = \frac{du}{dx} + \frac{dv}{dx}$$

$$\frac{d(uv)}{dx} = u\frac{dv}{dx} + v\frac{du}{dx}$$

$$\frac{d\left(\frac{u}{v}\right)}{dx} = \frac{v\left(\frac{du}{dx}\right) - u\left(\frac{dv}{dx}\right)}{v^2}$$

Taylor Series Approximation of Functions

Taylor series approximation of $f(x)$

If $f(x)$ is differentiable i.e., its derivatives

$\frac{df}{dx}, \frac{d^2 f}{dx^2} = \frac{d}{dx}\left(\frac{df}{dx}\right), \ldots \frac{d^n f}{dx^n}$ exist at $x = X_0$ and

$f(x)$ is continuous in the neighborhood of $x = X_0$, then

$$f(x) = f(X_0) + \left(\frac{df}{dx}\Big|_{x=X_0}\right)(x - X_0) + \ldots\ldots + \frac{1}{n!}\left(\frac{df^n}{dx^n}\Big|_{x=X_0}\right)(x - X_0)^n$$

$$f(x) \approx f(X_0) + \left(\frac{df}{dx}\Big|_{x=X_0}\right)(x - X_0)$$

Chain rule

Let $f(\mathbf{X}) = f(x_0, x_1, x_2, \ldots\ldots x_n)$

$\frac{\partial f}{\partial x_i}$ is obtained by treating all $x_i \mid i \neq j$ as constant.

Chain rule

Let $z = \varphi(u_1 \ldots . u_m)$

Let $u_i = f_i(x_0, x_1 \ldots\ldots x_n)$

Then $\forall k \quad \frac{\partial z}{\partial x_k} = \sum_{i=1}^{m} \left(\frac{\partial z}{\partial u_i} \right) \left(\frac{\partial u_i}{\partial x_k} \right)$

Taylor Series Approximation of Multivariate Functions

Let $f(\mathbf{X}) = f(x_{0,} x_1, x_2, \ldots\ldots x_n)$ be differentiable and continuous at $\mathbf{X}_0 = (x_{00,} x_{10}, x_{20}, \ldots\ldots x_{n0})$

Then

$$f(\mathbf{X}) \approx f(\mathbf{X}_0) + \sum_{i=0}^{n} \left(\frac{\partial f}{\partial x} \right)\Big|_{\mathbf{X}=\mathbf{X}_0} (x_i - x_{i0})$$

Minimizing / Maximizing Multivariate Functions

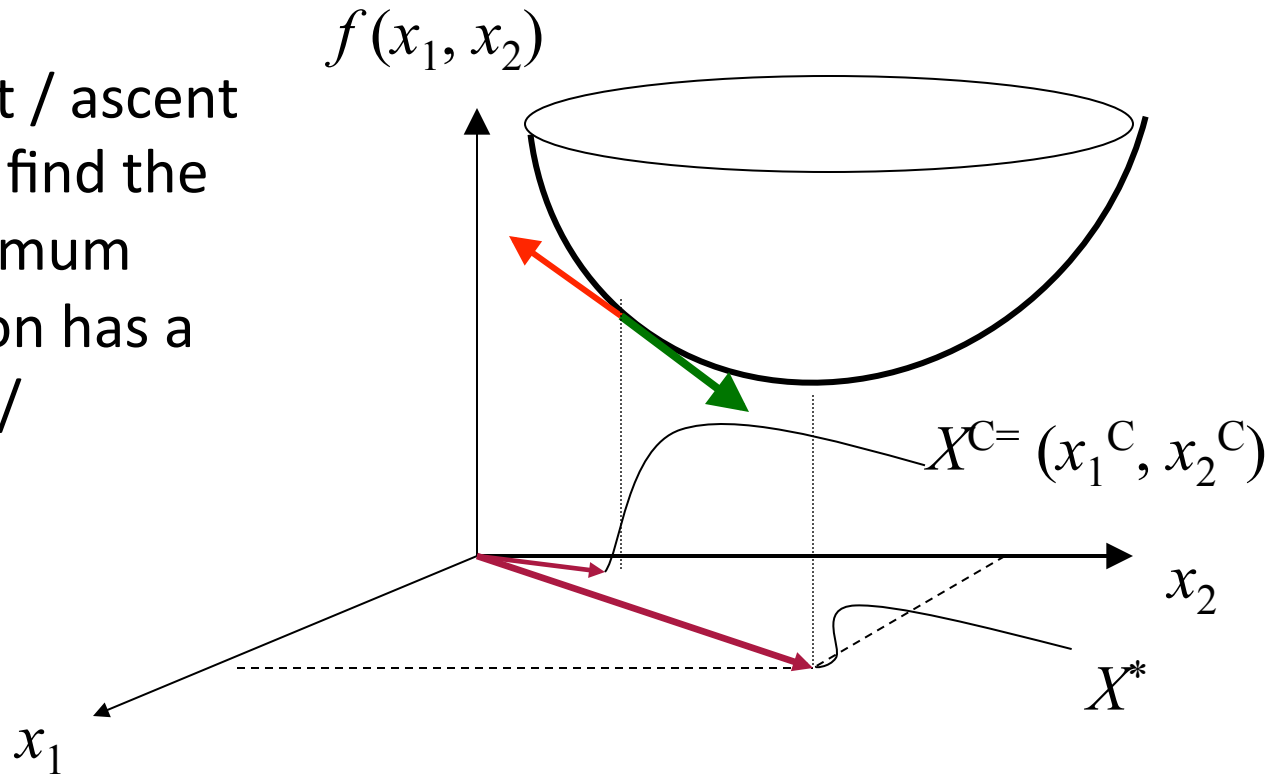
To find $\mathbf{X}^*$ that minimizes $f(\mathbf{X})$, we change current guess $\mathbf{X}^C$ in the direction of the negative gradient of $f(\mathbf{X})$ evaluated at $\mathbf{X}^C$

$$\mathbf{X}^C \leftarrow \mathbf{X}^C - \eta \left(\frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial x_1} \ldots\ldots\ldots\ldots \frac{\partial f}{\partial x_n} \right) \Bigg|_{\mathbf{X}=\mathbf{X}^C}$$ (why?)

for small (ideally infinitesimally small)

Minimizing / Maximizing Functions

Gradient descent / ascent is guaranteed to find the minimum / maximum when the function has a single minimum / maximum



Maximum likelihood estimation of $\mathbf{w}$

$$LL(\mathbf{w}:D) = \sum_{n=1}^{N} \{y_n \log \mu_n + (1-y_n)\log(1-\mu_n)\}$$

$$\frac{\partial LL(\mathbf{w}:D)}{\partial \mathbf{w}} = \sum_{n=1}^{N} \left(\frac{y_n}{\mu_n} - \frac{(1-y_n)}{(1-\mu_n)}\right)\left(\frac{\partial \mu_n}{\partial \eta_n}\right)\left(\frac{\partial \eta_n}{\partial \mathbf{w}}\right)$$

$$= \sum_{n=1}^{N} \left(\frac{y_n - \mu_n}{\mu_n(1-\mu_n)}\right)\mu_n(1-\mu_n)x_n$$

$$= \sum_{n=1}^{N} (y_n - \mu_n)x_n$$

$$G(\mathbf{x}) = \mathbf{x}$$
$$\eta(\mathbf{x}) = \mathbf{W}^T\mathbf{x}$$

Simple gradient ascent learning algorithm

$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) + \rho \left.\frac{\partial LL(\mathbf{w}:D)}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}(t)}$$

$$\rho > 0$$

$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) + \rho_t (y_n - \mu_n)x_n$$

$$\lim_{t\to\infty} \rho_t = 0; \quad \sum_{t=0}^{\infty} \rho_t = \infty; \quad \sum_{t=0}^{\infty} \rho_t^2 < \infty$$

Maximum likelihood estimation of $\mathbf{w}$

Simple gradient ascent algorithm can be quite slow and has little to recommend it in practice

The momentum trick provides a simple approach to speeding up the simple gradient ascent algorithm

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta\mathbf{w}(t)$$

$$\Delta\mathbf{w}(t) = \rho \left.\frac{\partial LL(\mathbf{w}:D)}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}(t)} + \alpha \Delta w_i(t-1) \text{ where } 0 < \alpha < 1$$

$$= \sum_{\tau=0}^{t} \alpha^{t-\tau} \left.\frac{\partial LL(\mathbf{w}:D)}{\partial \mathbf{w}}\right|_{w_i = w_i(\tau)}$$

Maximum likelihood estimation of $\mathbf{w}$

The momentum trick can also be applied in the on line version

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta\mathbf{w}(t)$$

$$\Delta\mathbf{w}(t) = \rho_t (y_n - \mu_n) x_n \big|_{\mathbf{w}=\mathbf{w}(t)} + \alpha \Delta w_i(t-1) \text{ where } 0 < \alpha < 1$$

$$= \sum_{\tau=0}^{t} \alpha^{t-\tau} \rho_t (y_n - \mu_n) x_n \big|_{w_i = w_i(\tau)}$$

Maximum likelihood estimation of $\mathbf{w}$

- More sophisticated optimization algorithms – line search, conjugate gradient, Newton-Raphson, iteratively reweighted least squares, and related methods can be used to maximize the log likelihood function which although not quadratic, is approximately quadratic.
- For details, see standard texts on optimization.
- When the form of the underlying generative model is known, we can initialize the parameter vector $\mathbf{w}$ based on the maximum likelihood estimates for which often closed form solutions are available and then run a few iterations of gradient ascent to improve classification accuracy.

Multi-class Discriminative Model

Softmax-linear model is the multi-class generalization of the logistic-linear model

$$p\left(y^k = 1 \mid \mathbf{x}\right) = \frac{e^{\boldsymbol{\theta}_k^T \mathbf{x}}}{\sum_{l=1}^{K} e^{\boldsymbol{\theta}_l^T \mathbf{x}}}$$

In the discriminative setting, we simply assume this form and proceed without regard to details of the underlying generative model

Multi-class Discriminative Model

Let $$p\left(y^k = 1 \mid x_n\right) = \frac{e^{\boldsymbol{\theta}_k^T x_n}}{\sum_{l=1}^{K} e^{\boldsymbol{\theta}_l^T x_n}} = \mu_n^k$$

Let $$\eta_n^k = \boldsymbol{\theta}_k^T x_n$$

$$\eta_n = \left[\eta_n^1 \ldots \eta_n^k \ldots \eta_n^K\right]$$

$$\mu_n = \left[\mu_n^1 \ldots \mu_n^k \ldots \mu_n^K\right]$$

Then $$\mu_n^k = \frac{e^{\eta_n^k}}{\sum_{l=1}^{K} e^{\eta_n^l}} \; ; \quad \mu^k = \frac{e^{\eta^k}}{\sum_{l=1}^{K} e^{\eta^l}}$$

Some properties of the softmax function

Softmax-linear function is invertible up to an additive constant.

$$\mu^k = \frac{e^{\eta^k}}{\sum_{l=1}^{K} e^{\eta^l}}$$

$$\eta^k = \log \mu^k + C$$

$$C = \log\left(\sum_{l=1}^{K} e^{\eta^l} \right)$$

Some properties of the softmax function

$$\mu^k = \frac{e^{\eta^k}}{\sum_{l=1}^{K} e^{\eta^l}}; \qquad \eta^k = \log \mu^k + C; \qquad C = \log\left(\sum_{l=1}^{K} e^{\eta^l} \right)$$

$$\frac{\partial \mu^k}{\partial \eta^j} = \frac{\left(\sum_{l=1}^{K} e^{\eta^l} \right) e^{\eta^k} \delta_{kj} - e^{\eta^k} e^{\eta^j}}{\sum_{l=1}^{K} e^{\eta^l}}$$

$$= \frac{e^{\eta^k}}{\sum_{l=1}^{K} e^{\eta^l}} \left(\delta_{kj} - \frac{e^{\eta^j}}{\sum_{l=1}^{K} e^{\eta^l}} \right) = \mu^k \left(\delta_{kj} - \mu^j \right)$$

Maximum likelihood estimation of $\mathbf{w}$

$$D = \left\{ \left(x_n, y_n \right); x_n \in Domain\left(\mathbf{x} \right); \quad y_n \in \left\{ y_n^1 \ldots y_n^K \right\}; \; n = 1..N; \right\}$$

$$P\left(y_n \mid x_n, \boldsymbol{\theta} \right) = \prod_{k=1}^{K} \left(\mu_n^k \right)^{y_n^k}$$

$$L\left(\boldsymbol{\theta} : D \right) = P\left(y_1 \ldots y_N \mid x_1 \ldots x_N, \boldsymbol{\theta} \right) = \prod_{n=1}^{N} \prod_{k=1}^{K} \left(\mu_n^k \right)^{y_n^k}$$

$$LL\left(\boldsymbol{\theta}_1 \ldots \boldsymbol{\theta}_K : D \right) = \sum_{n=1}^{N} \sum_{k=1}^{K} y_n^k \log \mu_n^k$$

We need to find parameters that maximize log likelihood

Maximum likelihood estimation of $\mathbf{w}$

$$\eta_n^k = \boldsymbol{\theta}_k^T x_n$$

$$LL(\boldsymbol{\theta} : D) = \sum_{n=1}^{N} \sum_{k=1}^{K} y_n^k \log \mu_n^k$$

$$\nabla_{\boldsymbol{\theta}_i} LL(\boldsymbol{\theta} : D) = \sum_{n=1}^{N} \sum_{k=1}^{K} \left(\frac{\partial LL(\boldsymbol{\theta} : D)}{\partial \mu_n^k} \right) \left(\frac{\partial \mu_n^k}{\partial \eta_n^i} \right) \left(\frac{\partial \eta_n^i}{\partial \boldsymbol{\theta}_i} \right)$$

$$= \sum_{n=1}^{N} \sum_{k=1}^{K} y_n^k \left(\delta_{ik} - \mu_n^i \right) x_n$$

$$= \sum_{n=1}^{N} \left(y_n^i - \mu_n^i \right) x_n$$

Where we have used the chain rule and the fact that

$$\forall n \quad \left(\sum_{k=1}^{K} y_n^k = 1 \right)$$

Maximum likelihood estimation of $\mathbf{w}$

$$\nabla_{\boldsymbol{\theta}_i} LL(\boldsymbol{\theta} : D) = \sum_{n=1}^{N} \left(y_n^i - \mu_n^i \right) x_n$$

Basic gradient ascent update rule is given by

$$\boldsymbol{\theta}_i(t+1) \leftarrow \boldsymbol{\theta}_i(t) + \rho \left. \frac{\partial LL(\boldsymbol{\theta} : D)}{\partial \boldsymbol{\theta}_i} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}(t)}$$

$$\rho > 0$$

which we can be speed up using the momentum trick as before

Maximum likelihood estimation of $\mathbf{w}$

The momentum trick provides a simple approach to speeding up the simple gradient ascent algorithm

$$\boldsymbol{\theta}_i(t+1) = \boldsymbol{\theta}_i(t) + \Delta\boldsymbol{\theta}_i(t)$$

$$\Delta\boldsymbol{\theta}_i(t) = \rho \left.\frac{\partial LL(\boldsymbol{\theta}:D)}{\partial \boldsymbol{\theta}_i}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}(t)} + \alpha\Delta\boldsymbol{\theta}_i(t-1) \text{ where } 0 < \alpha < 1$$

$$= \sum_{\tau=0}^{t} \alpha^{t-\tau} \left.\frac{\partial LL(\boldsymbol{\theta}:D)}{\partial \boldsymbol{\theta}_i}\right|_{\boldsymbol{\theta}_i=\boldsymbol{\theta}_i(\tau)}$$

Maximum likelihood estimation of $\mathbf{w}$

$$\nabla_{\boldsymbol{\theta}_i} LL(\boldsymbol{\theta} : D) = \sum_{n=1}^{N} \left(y_n^i - \mu_n^i\right) x_n$$

Basic online gradient ascent update rule is given by

$$\boldsymbol{\theta}_i(t+1) \leftarrow \boldsymbol{\theta}_i(t) + \rho_t \left(y_n^i - \mu_n^i\right) x_n$$

$$\lim_{t \to \infty} \rho_t = 0; \quad \sum_{t=0}^{\infty} \rho_t = \infty; \quad \sum_{t=0}^{\infty} \rho_t^2 < \infty$$

which we can speed up using the momentum trick as before

Maximum likelihood estimation of $\mathbf{w}$

The momentum trick can also be applied in the on line version

$$\boldsymbol{\theta}_i(t+1) = \boldsymbol{\theta}_i(t) + \Delta\boldsymbol{\theta}_i(t)$$

$$\Delta\boldsymbol{\theta}_i(t) = \rho_t\left(y_n^i - \mu_n^i\right)x_n\Big|_{\boldsymbol{\theta}_i=\boldsymbol{\theta}_i(t)} + \alpha\Delta\boldsymbol{\theta}_i(t-1) \text{ where } 0 < \alpha < 1$$

$$= \sum_{\tau=0}^{t} \alpha^{t-\tau}\rho_t\left(y_n - \mu_n\right)x_n\Big|_{\boldsymbol{\theta}_i=\boldsymbol{\theta}_i(\tau)}$$

Summary

- For a large class of generative models, the probability distribution of class conditioned on the input can be modeled by the exponential family
- Generative models can perform poorly when the assumed parametric form for the distribution is incorrect
- Discriminative models can perform poorly when the assumed form of $G(\mathbf{x})$ is inappropriate – but it is often easier to choose the form of $G(\mathbf{x})$ than it is to specify the precise form of the generative model
- Discriminative models focus on the classification problem without solving (potentially more difficult) problem of learning the generative model for data
- Estimating the parameters in the discriminative setting requires solving an optimization problem although their generative counterparts have closed form solutions (via sufficient statistics)

Summary

- We can learn classifiers in a discriminative setting using maximum likelihood or maximum a posteriori or bayesian estimation of parameters
- Discriminative models may overfit the data – use of priors or regularization recommended
- Initializing the discriminative model parameters with estimates based on generative model helps