工程实践与科技创新课程：
超并行机器学习与海量数据挖掘

吕宝粮
计算机科学与工程系
电信楼群3号楼431
Tel: 021-3420-5422

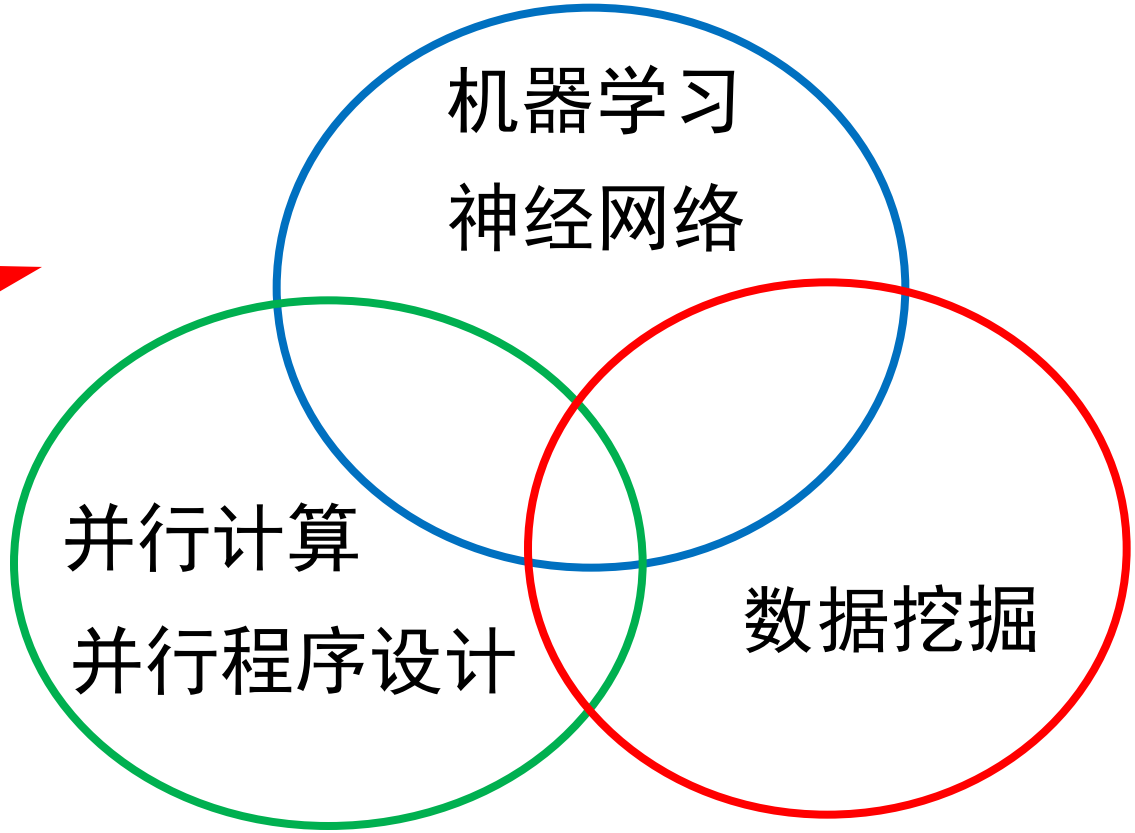
bllu@sjtu.edu.cn
http://bcmi.sjtu.edu.cn/~blu

课程安排

- 讲课：16学时；2月22日至3月15日（共四周）：
  每周三下午7-10节
- 助教：刘伟：liuwei@liujr.com
- 实验：16学时（2人一组；指定）；3月16日-4月28日
- 大作业：4月28日提交报告
- 5月中旬大作业讲演会：6-7组；每组2分钟介绍；全体参加
- 学分：2
- 评分形式：平时作业：30%；
  小测验及课堂表现：20%；
  大作业：50%

为什么要开设此课程？

大数据
云计算



AlphaGo掀起了新一轮AI浪潮！

- 2016年1月，Google在Nature上发文，表示他们的AlphaGo系统在正式比赛中打败了欧洲围棋冠军。
- 2016年3月，AlphaGo成功挑战韩国围棋九段李世石，比分为4:1
- 2017年年初Master横扫人类棋手！
- AlphaGO:强化学习+深度学习+搜索技术
- AI将对人类社会的各个方面产生深远影响！

大数据及其特点

- 大数据(Big Data)是指由于规模和复杂度两方面因素，无法在一定的时间内用常规软件工具对其内容进行抓取、管理和处理的数据集合。
- Volume：数据体量巨大。从TB级别，跃升到PB级别；
- Variety：数据类型繁多。网络日志、视频、图片、地理位置信息等。
- Value：价值密度低。以视频为例，连续不间断监控过程中，可能有用的数据仅仅有一两秒。
- Velocity：数据的产生和更新速度快，需要我们能快速实时地处理。

(1TB=1024GB; 1PB=1024TB)

课程简介

- **机器学习**（Machine Learning)：

  如果一个计算机程序针对某类任务T，在P的衡量标准下，根据经验E来自我完善，那么我们称这个计算机程序在从经验E中学习。

  机器学习致力于研究建立能够根据经验自我提高处理性能的计算机程序。

- **超并行机器学习** (Massively Parallel Machine Learning)

  为了解决大规模问题，在超并行计算环境下，实现机器学习算法

课程简介（续）

- **数据挖掘** (Data Mining)

  从大量数据中提取或“挖掘”知识

- **海量数据挖掘** (Massive Data Mining)

  如何利用超并行计算环境，从海量数据中，发现知识。

- **课程的目的：**
  - 掌握机器学习和数据挖掘的基本原理
  - 实践并行程序设计
  - 了解目前的方法和尚未解决的问题
  - 为将来应用和研究新的理论、方法和技术作准备

讲课的顺序和学时比例

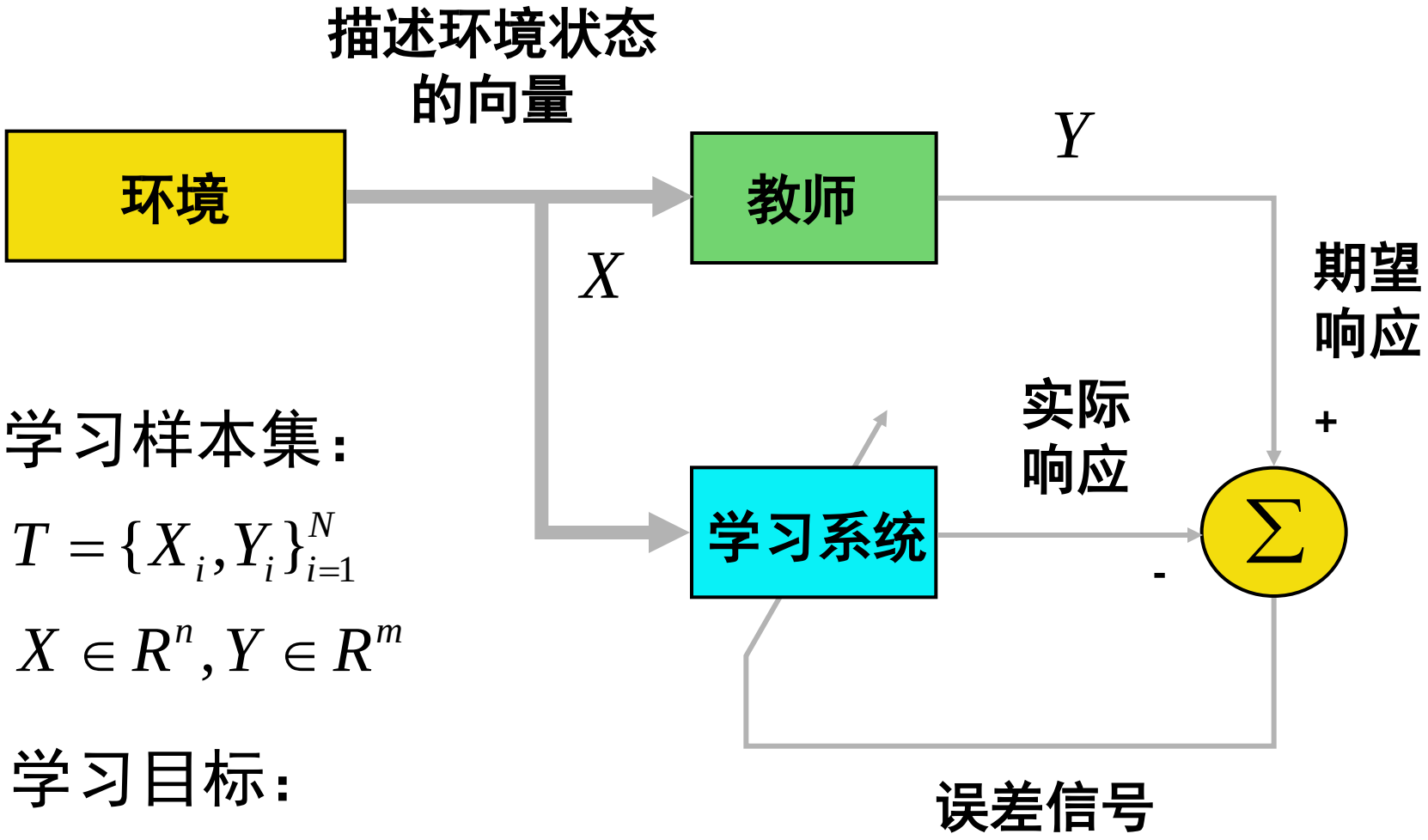
- 机器学习（14学时）
- 数据挖掘，并行程序设计（2学时）

机器学习概述

机器学习的三种范式

- 有监督学习（Supervised Learning）
- 无监督学习（Unsupervised Learning）
- 强化学习（Reinforcement Learning）

监督学习



学习样本集：

$$T = \{X_i, Y_i\}_{i=1}^{N}$$

$$X \in R^n, Y \in R^m$$

学习目标：

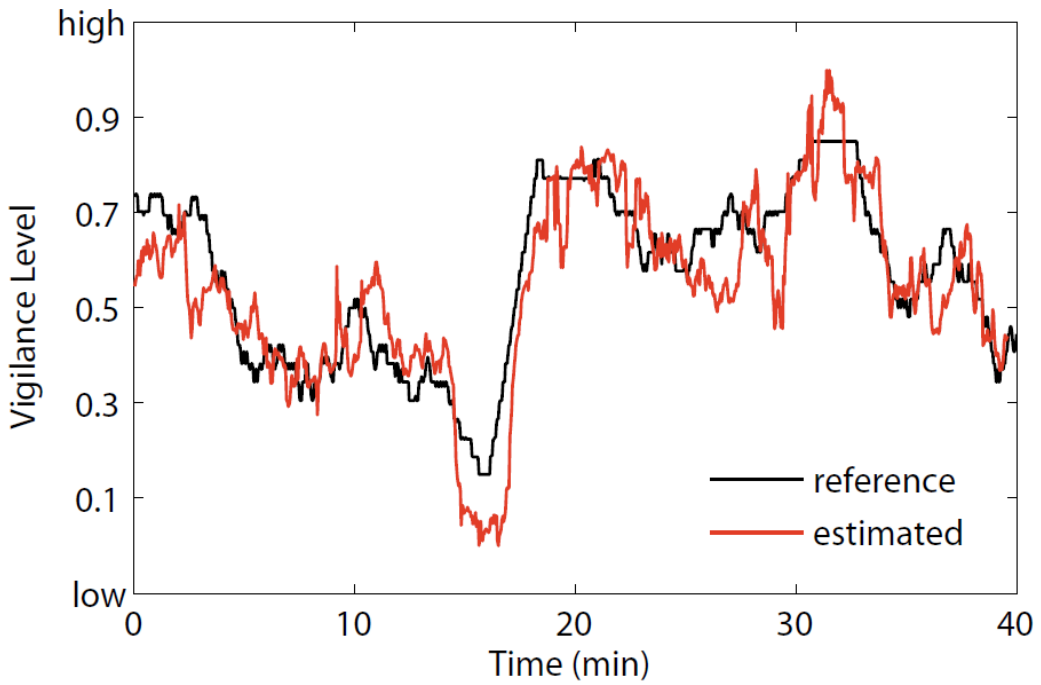
$$Y = \hat{F}(X)$$

监督学习的两类学习任务

- 分类(Classification): 教师信号是离散值
- 预测 (Prediction): 教师信号是连续值

  也叫回归 (Regression)，或

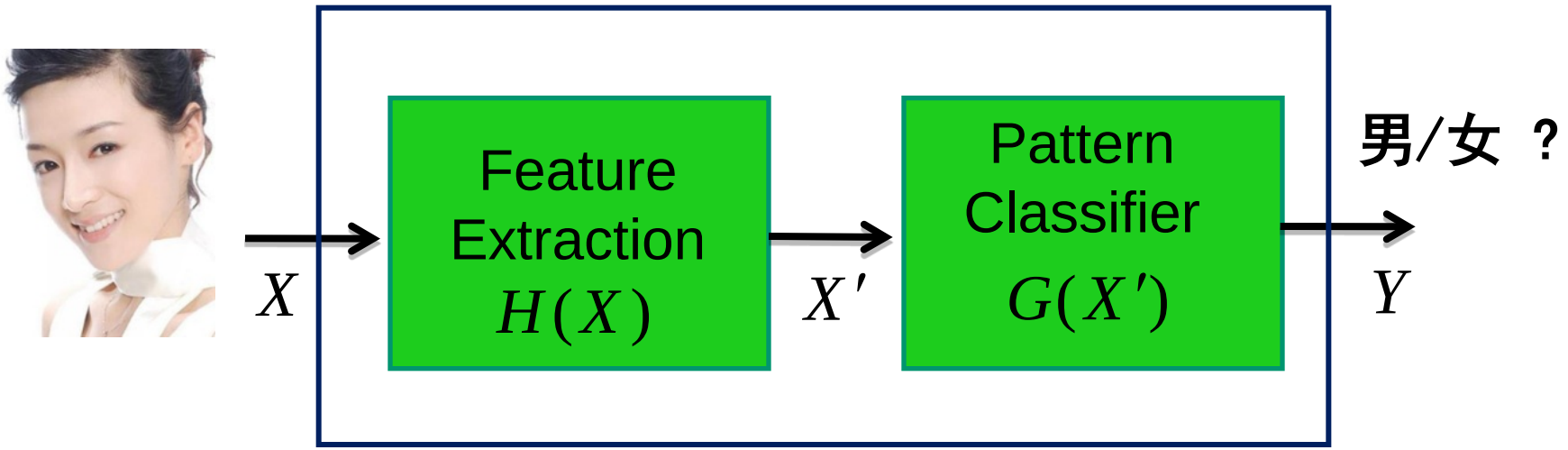
  函数逼近 (Function Approximation)



实际的分类问题：性别识别

基于机器学习的性别识别系统

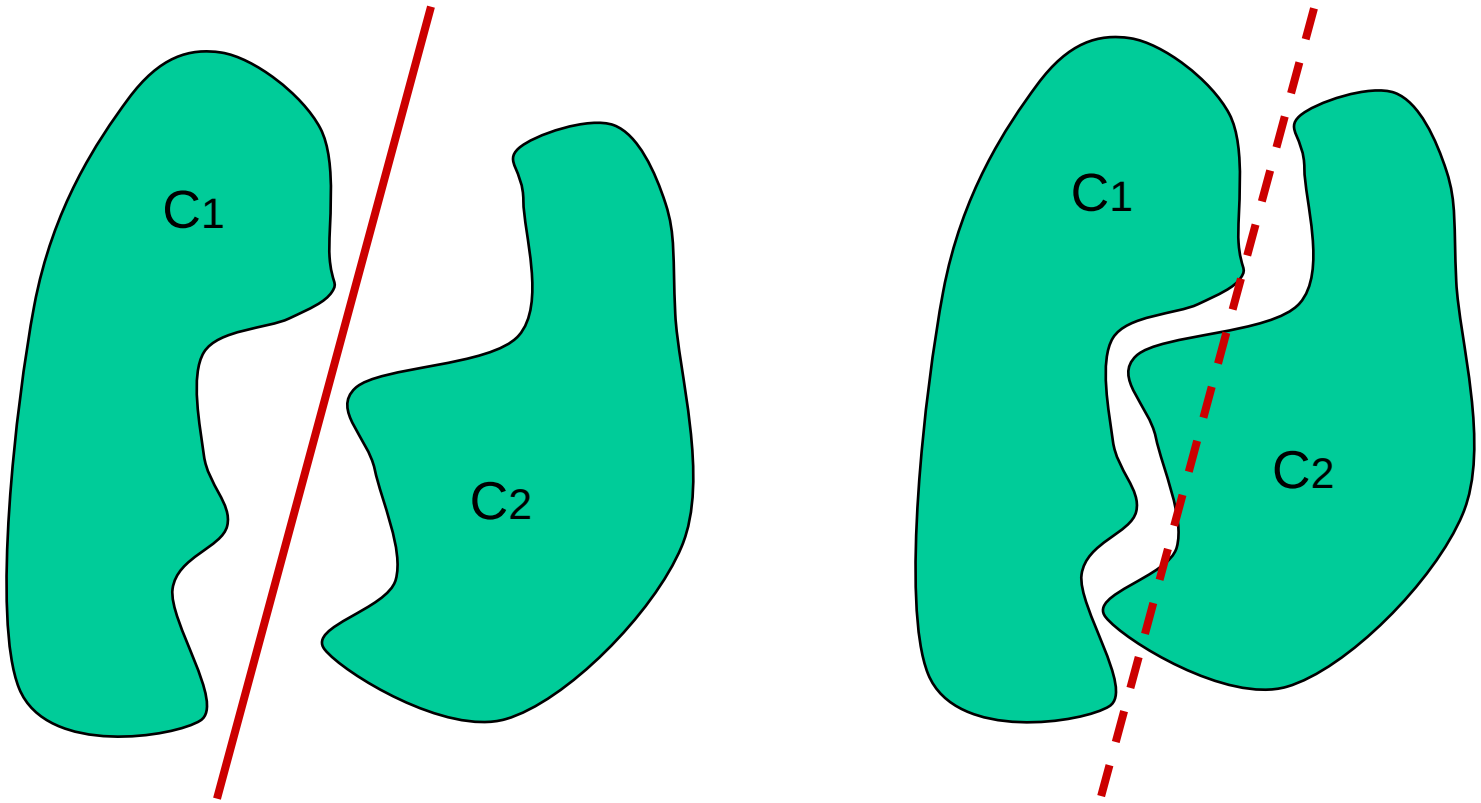
- 系统输入：人脸图像（X）
- 系统输出：男/女（Z）
- 问题：如何用机器学习方法构造系统？



性别识别系统

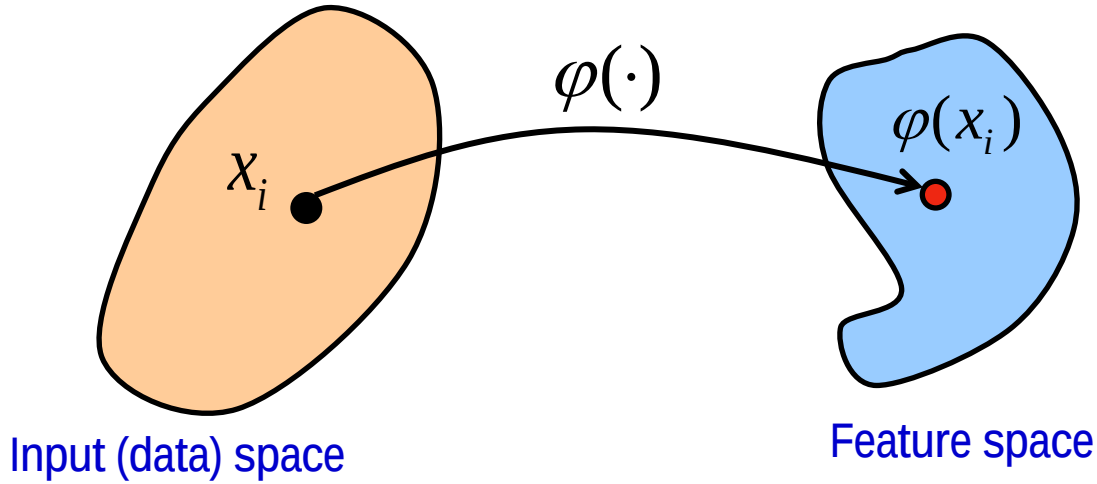
线性可分与非线性可分

- 线性可分(Linear Separable)
- 线性不可分（Linear Non-separable）

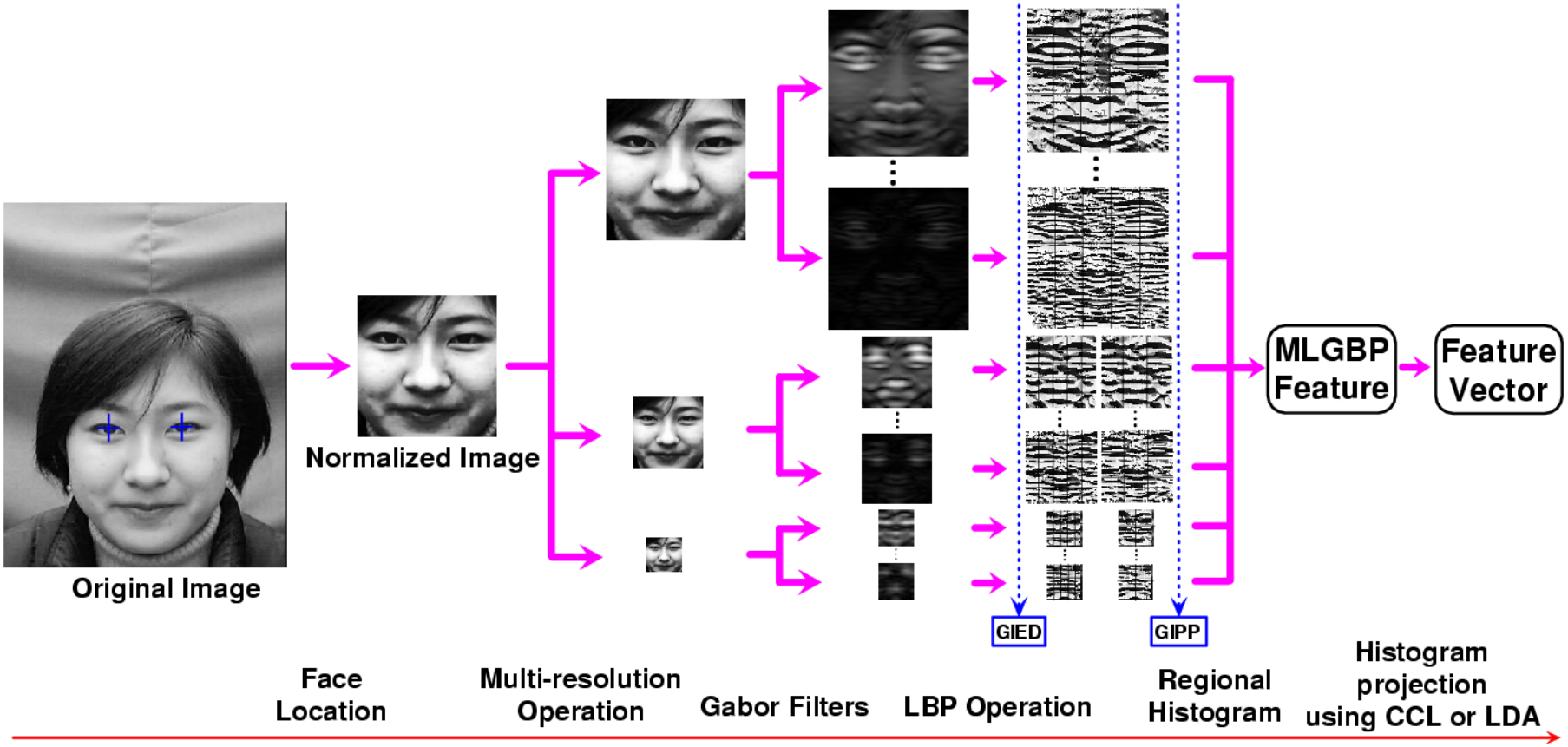


特征提取与特征选择

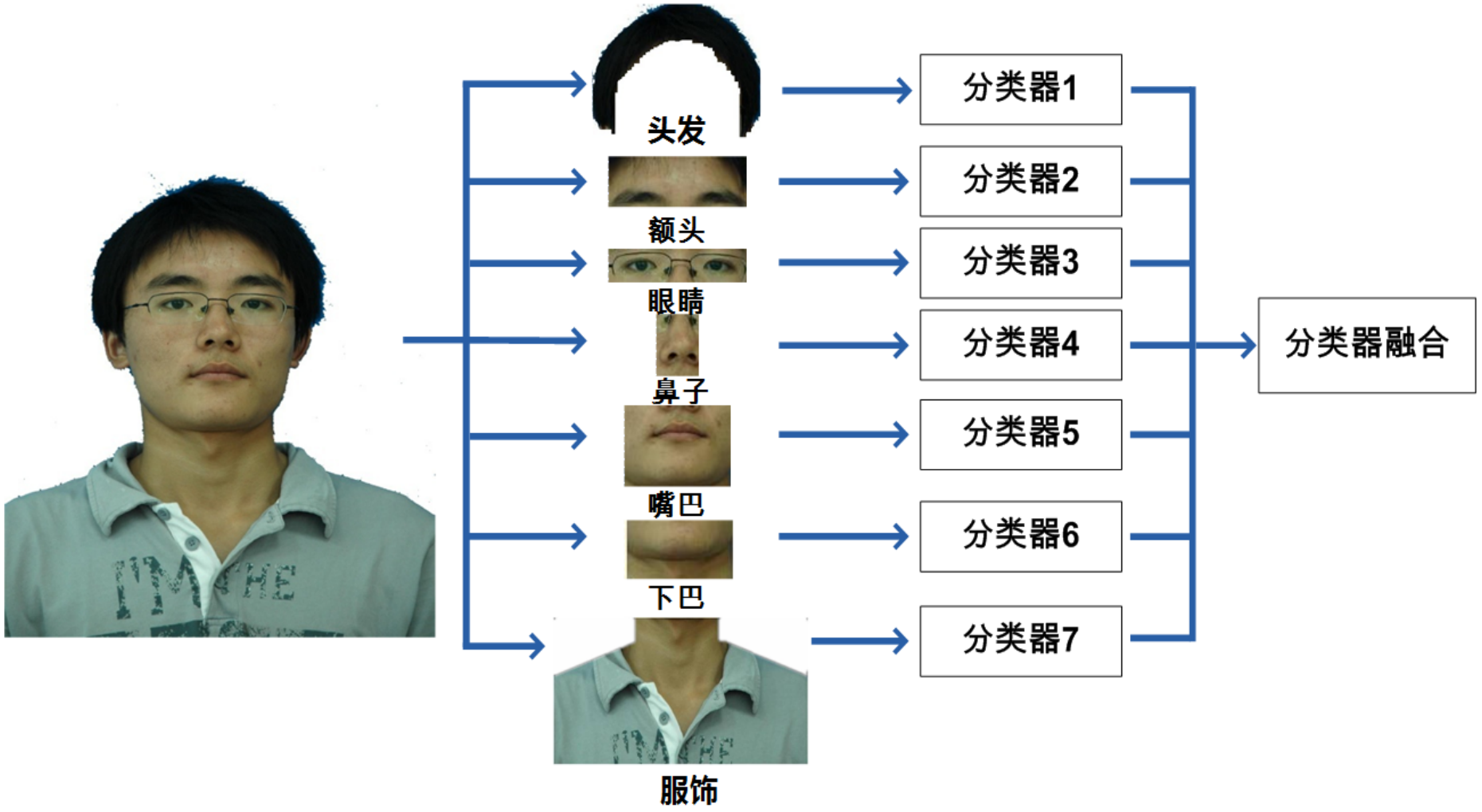
- 特征提取或抽取（Feature Extraction）
  - 对某一模式的一组测量值进行变换或映射以突出该模式具有代表性特征的方法
- 特征选择（Feature Selection）



人脸特征（ MLGBP）

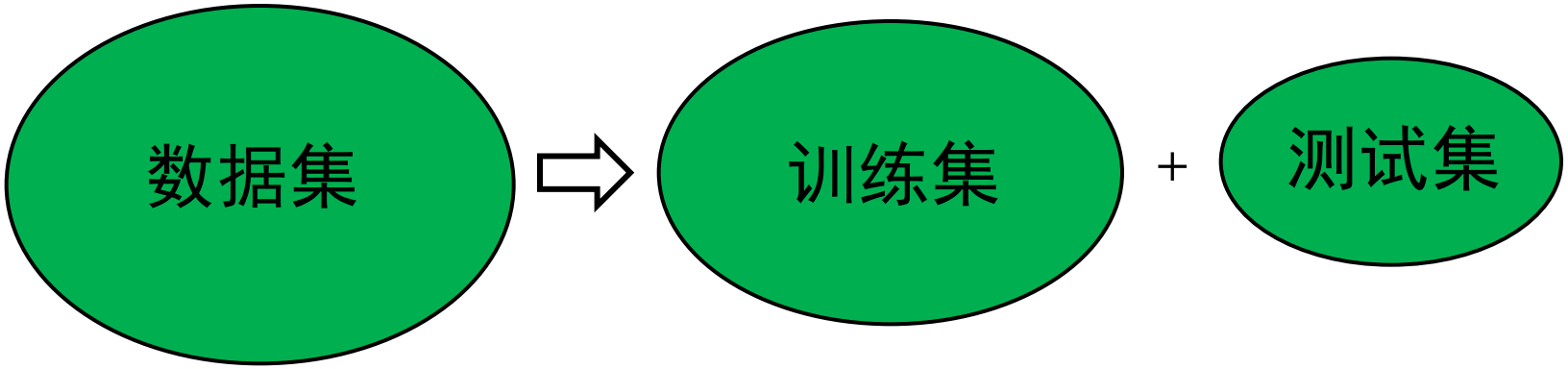

Original Image
Normalized Image
GIED
GIPP
MLGBP Feature
Feature Vector
Face Location
Multi-resolution Operation
Gabor Filters
LBP Operation
Regional Histogram
Histogram projection using CCL or LDA

多特征性别分类框架



监督学习的两个不同阶段

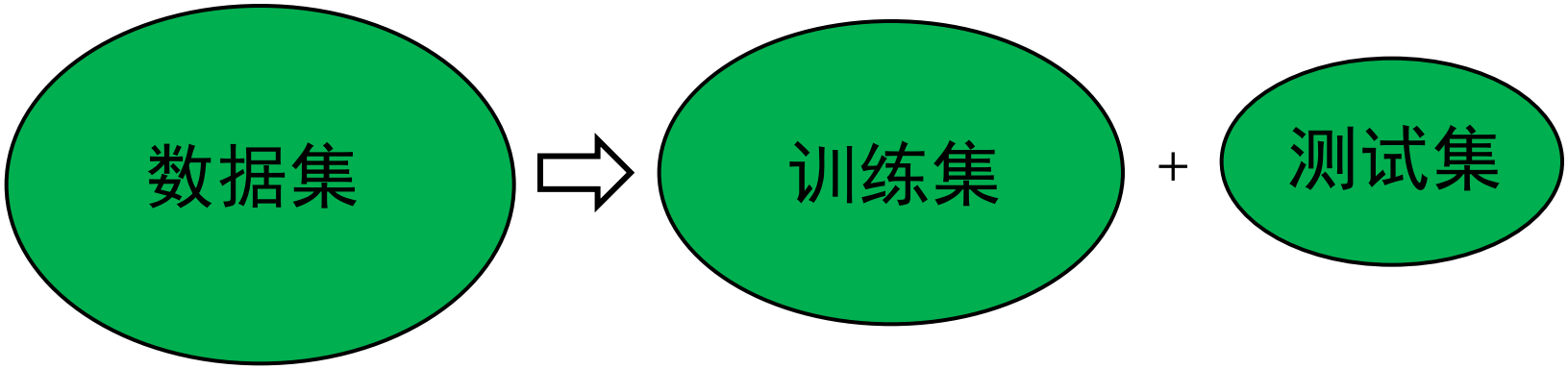
- 学习阶段或称训练阶段
  - 训练数据集
- 预测阶段或称识别阶段
  - 测试数据集
- 学习系统的一般化能力或泛化能力



$$T_{All} = T_{Train} \bigcup T_{Test} \; ; \quad T_{Train} \bigcap T_{Test} = \Phi$$

交叉验证（Cross-validation）

- 数据集的分割



$$T_{All} = T_{Train} \bigcup T_{Test} \; ; \quad T_{Train} \bigcap T_{Test} = \Phi$$

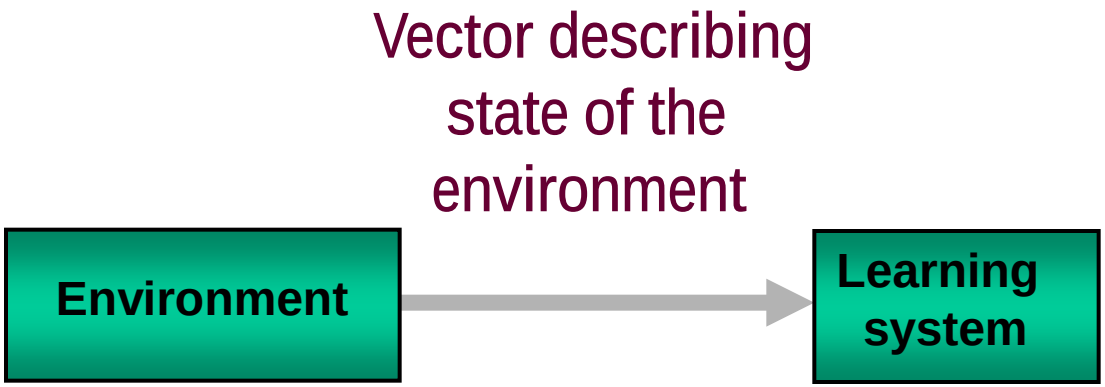
- 用交叉验证的目的是为了得到可靠稳定的模型
- 10折交叉验证(10-fold cross validation)，将数据集分成十份，轮流将其中9份做训练1份做测试，10次的结果的均值作为对算法精度的估计。

训练集和测试集同分布假设

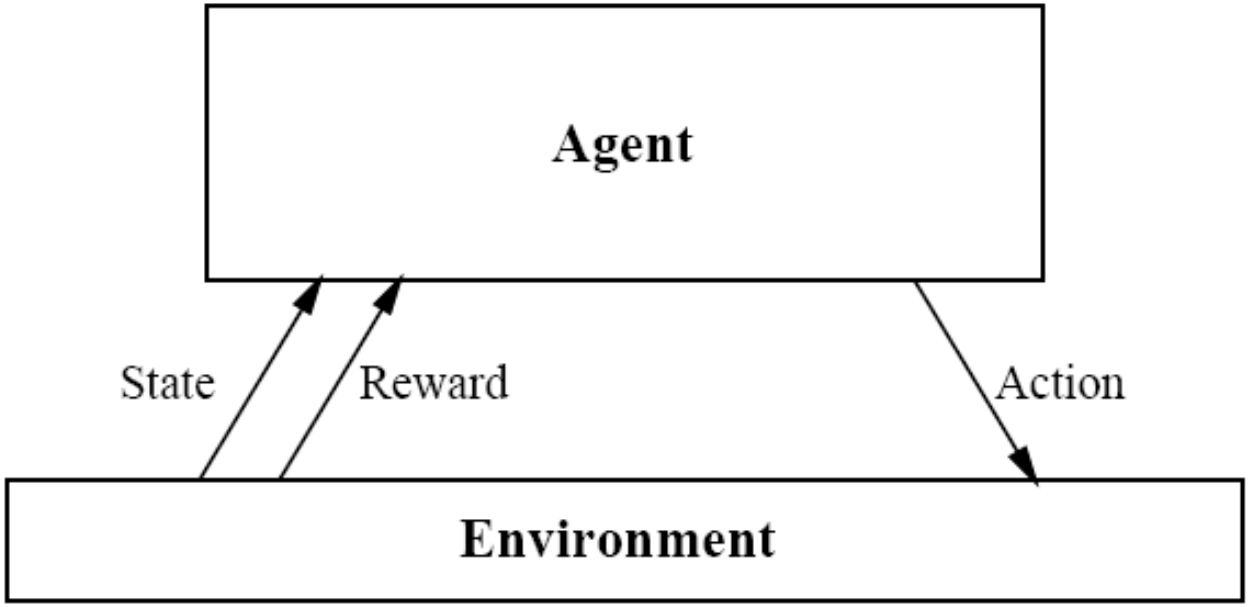
- 机器学习理论基于训练数据集与测试数据集具有相同分布的假设
- 在解决实际问题时，上述假设经常是不成立的。例如，脑电信号。

无监督学习

Unsupervised Learning:
No labeled training examples are now available



强化学习



$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \ldots$$

Goal: Learn to choose actions that maximize

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots \text{ , where } 0 \leq \gamma < 1$$

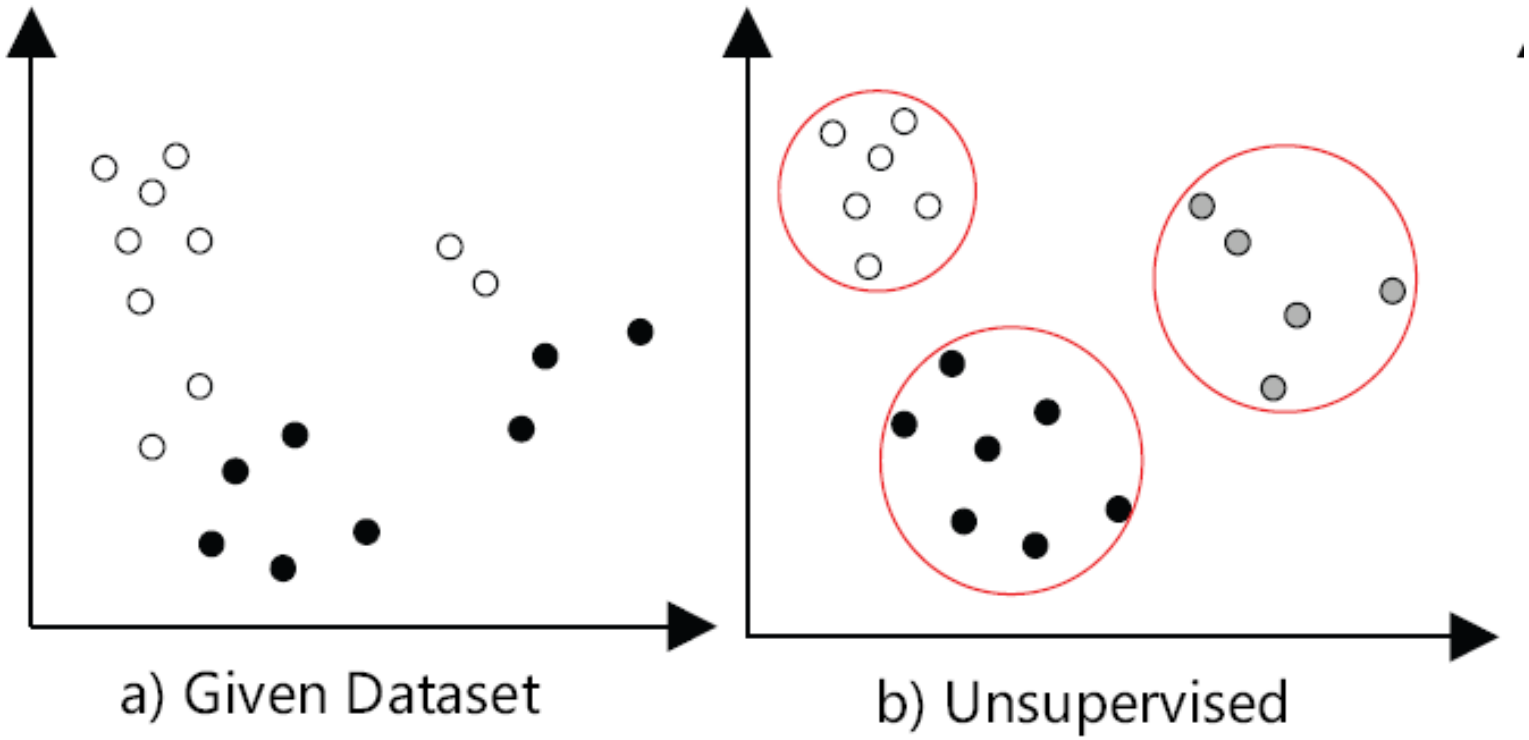
强化学习

- 所谓强化学习就是智能系统从环境到行为映射的学习，以使奖励信号(强化信号)函数值最大
- 强化学习不同于监督学习，主要表现在教师信号上，强化学习中由环境提供的强化信号是对产生动作的好坏作一种评价(通常为标量信号)，而不是告诉强化学习系统 （RLS）如何去产生正确的动作。
- 由于外部环境提供的信息很少，RLS必须靠自身的经历进行学习。
- 通过这种方式，RLS在行动-评价的环境中获得知识，改进行动方案以适应环境。

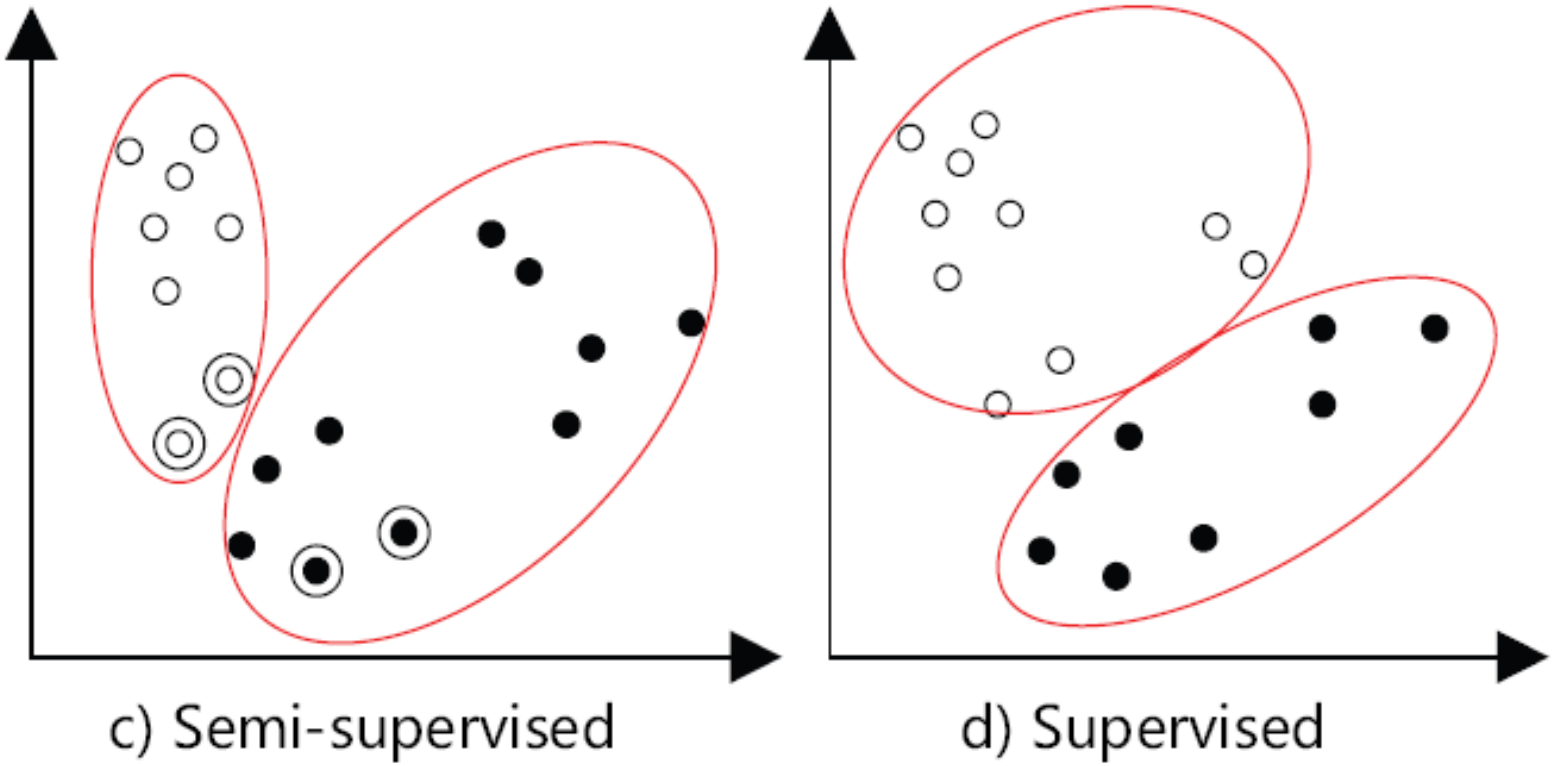
其他学习范式

- Semi-supervised Learning
- Semi-supervised clustering
- Supervised clustering
- Transfer learning

Unsupervised Clustering



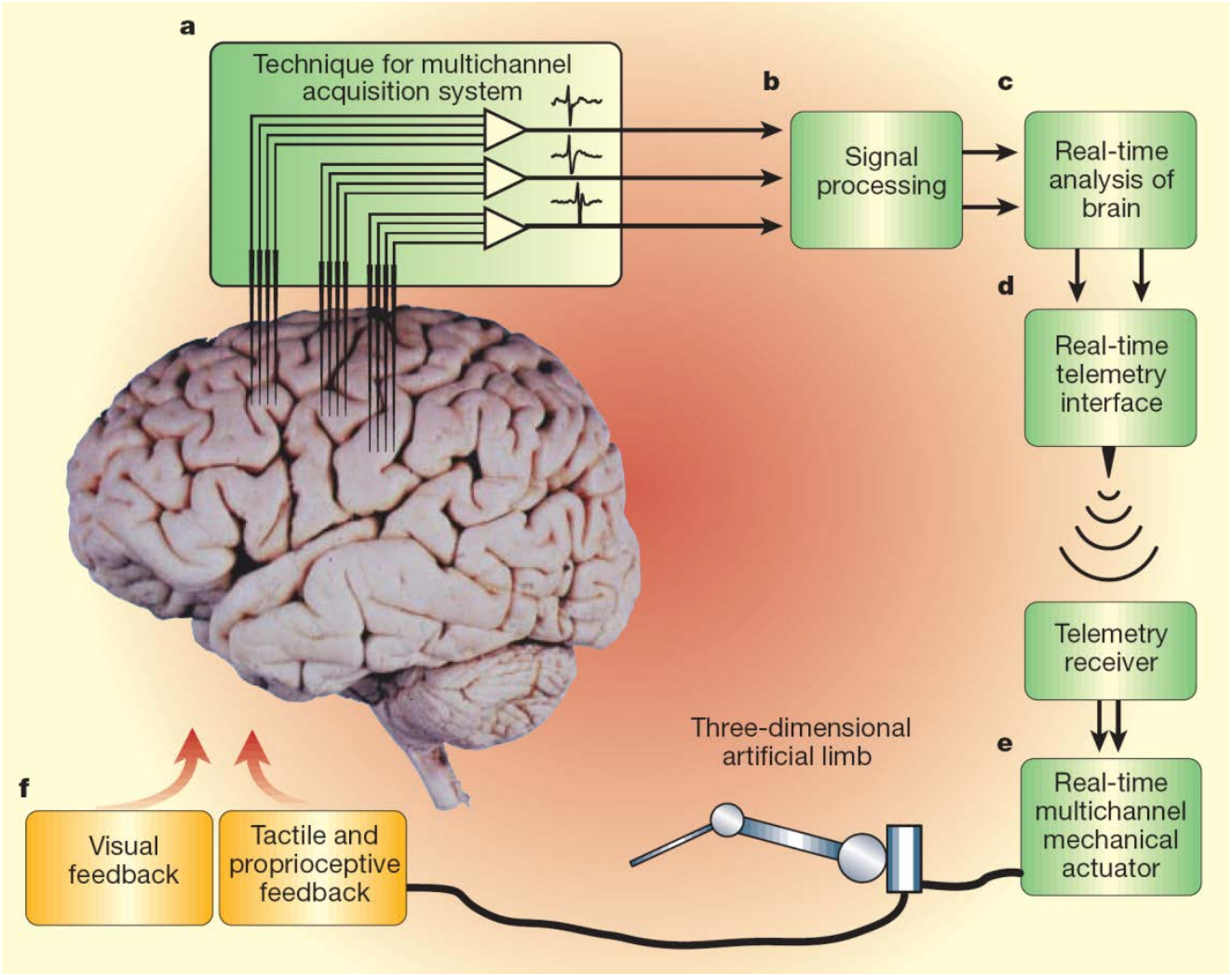
Semi-supervised vs Supervised



Applications in my Lab

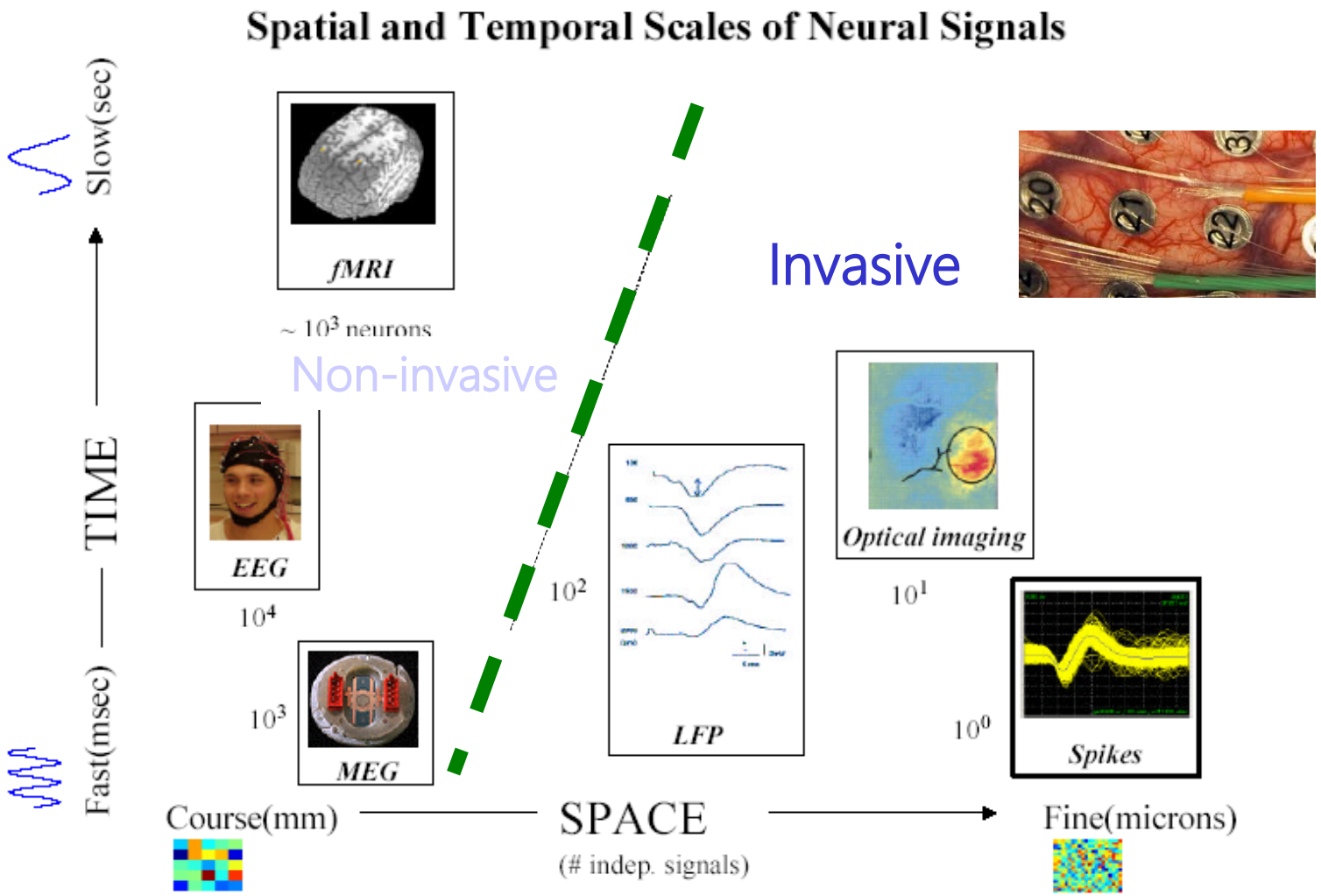
- Driving fatigue detection
- Affective Brain-Computer Interaction (aBCI）

Brain-Computer/Machine Interface, BCI, BMI



脑-机接口是是在人或动物脑与外部设备间创建的直接连接通路

BCIs: Invasive vs. Non-invasive



《科学》2012年度十大科学突破

BRAIN-MACHINE INTERFACES START TO GET A GRIP

This week researchers in Pennsylvania reported that a 53-year-old woman paralyzed from the neck down by a genetic neurodegenerative condition had learned to manipulate a robotic arm with her thoughts. Surgeons had implanted two 4×4-millimeter grids of hair-thin electrodes in her brain to capture signals from an area involved in planning hand movements. A computer translated those signals into commands to move the robotic arm, which was engineered to have nearly all the same movement capabilities as the real thing. In videos, the woman uses the arm to grasp and move various objects, removing plastic cones stacked on a base and restacking them one by one on another base, for example. The demonstrations represent the most complex movements yet performed by a paralyzed human patient using a brain-machine interface (BMI), as such sophisticated prosthetics are often called.

RUNNER-UP

By demonstrating more fluid and natural movements, this case study improves on another impressive report earlier this year. In that study—the first published demonstration that paralyzed human patients can use a BMI to execute complex movements in three dimensions—a 58-year-old woman who had been unable to speak or move her limbs for 15 years manipulated a robotic arm with her thoughts, reaching out to grasp a bottle and take a sip of coffee. A tetraplegic man, 66, also learned to touch and grasp objects.

All of this work builds on more than a decade of research with monkeys and other animals. And that work continues to advance. In 2011, researchers described a prosthetic system that provides tactile feedback by stimulating the somatosensory cortex, the brain region responsible for the perception of touch. And in April of this year, a team used signals from electrodes implanted in the motor cortex of the brain to stimulate muscles in the temporarily paralyzed arms of two monkeys, enabling the animals to pick up rubber balls and place them in a chute. Such

CREDIT: KAROL NASS, CFEL

1530 21 DECEMBER 2012 VOL 338 SCIENCE www.sciencemag.org
Published by AAAS

9. 脑–机接口：曾经用大脑神经记录移动电脑荧幕上光标的同一个研究团队在2012年向人们展示，瘫痪的病人能够用他们的思想来移动一个机械臂并从事复杂的三维运动。该技术虽然仍处于试验阶段且极端昂贵，但科学家希望更先进的计算程序可改善这种神经性假体以帮助因中风、脊髓损伤及其他疾病导致瘫痪的病人。


PVAMC
MGH

Applications in my Lab

- Driving fatigue detection
- Affective Brain-Computer Interaction (aBCI）

High demand for detecting driving fatigue

- In China, the number of traffic accident deaths is more than 75 thousands each year during 2006-2010.
- We have good technologies for detecting
  - **drunk driving**
  - **Over speed**
  - **Over load**
  - **Over persons**

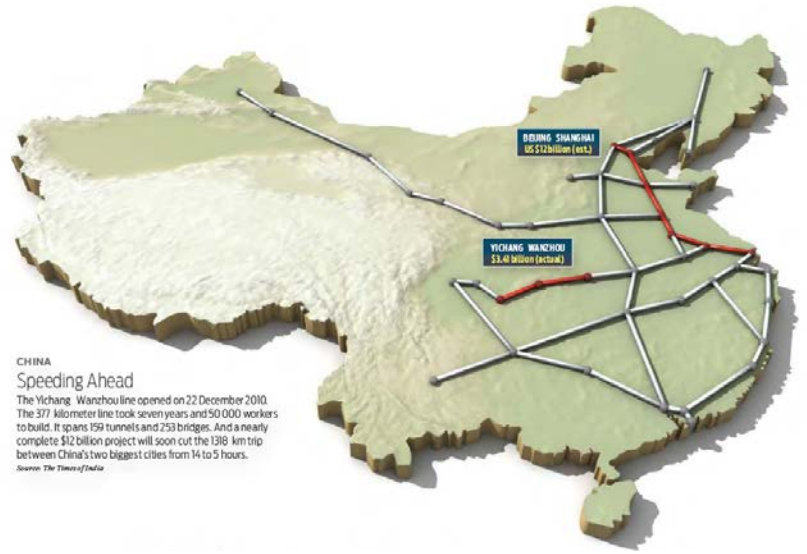
- Unfortunately, we have no reliable methods for monitoring drowsy/fatigue driving
- Difficulties of detecting drowsy/fatigue driving
  - **Drowsy/fatigue is a process, instead of a time point as checking drunk driving**
  - **There is no good measure to judge drowsy/fatigue level**

Asleep at the wheel of a high-speed train

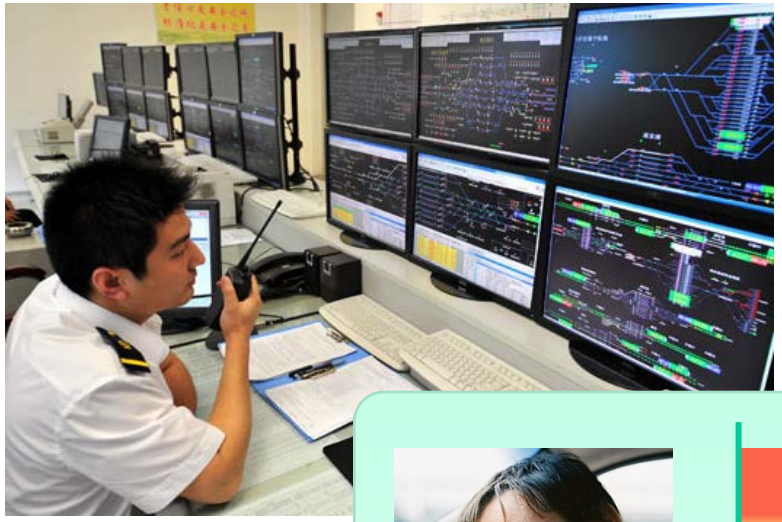
- The driver of a high-speed train in Taiwan fell asleep while going 190 miles per hour with hundreds of passengers on board on 10 May, 2010.
- Using the train‘s automatic systems, the train controllers managed to take control of the train for about 13 minutes until it glided safely into Taichung station.
- There was no casualty in this accident.

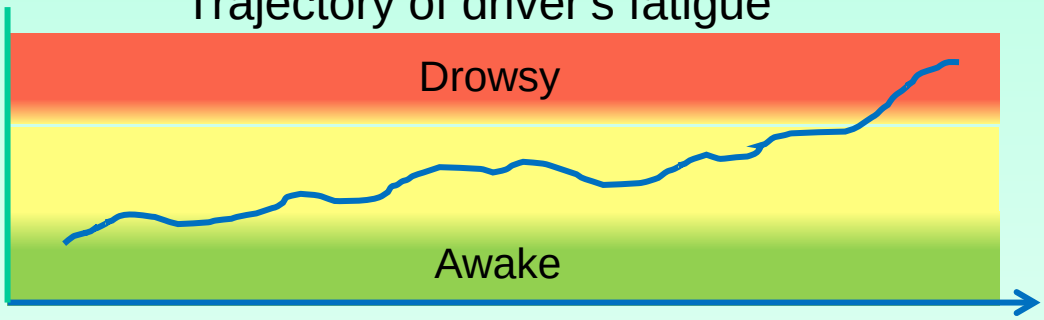
High-speed railways in China

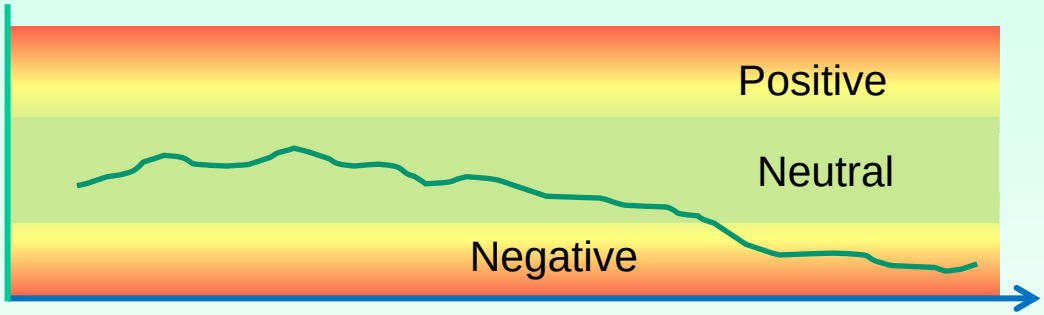
- 20380 kms in use. It is about half the world's supply
- 16775 kms are developing
- 30000 kms will be finished by 2020
- Active safety and mental state monitoring technologies become very important
  - Check sleep quality before work
  - Predict drowsy driving during work
  - Monitor mental state before work, during work, and after work



Our goal: Monitor driver's drowsy and emotion







Related work: image based approach

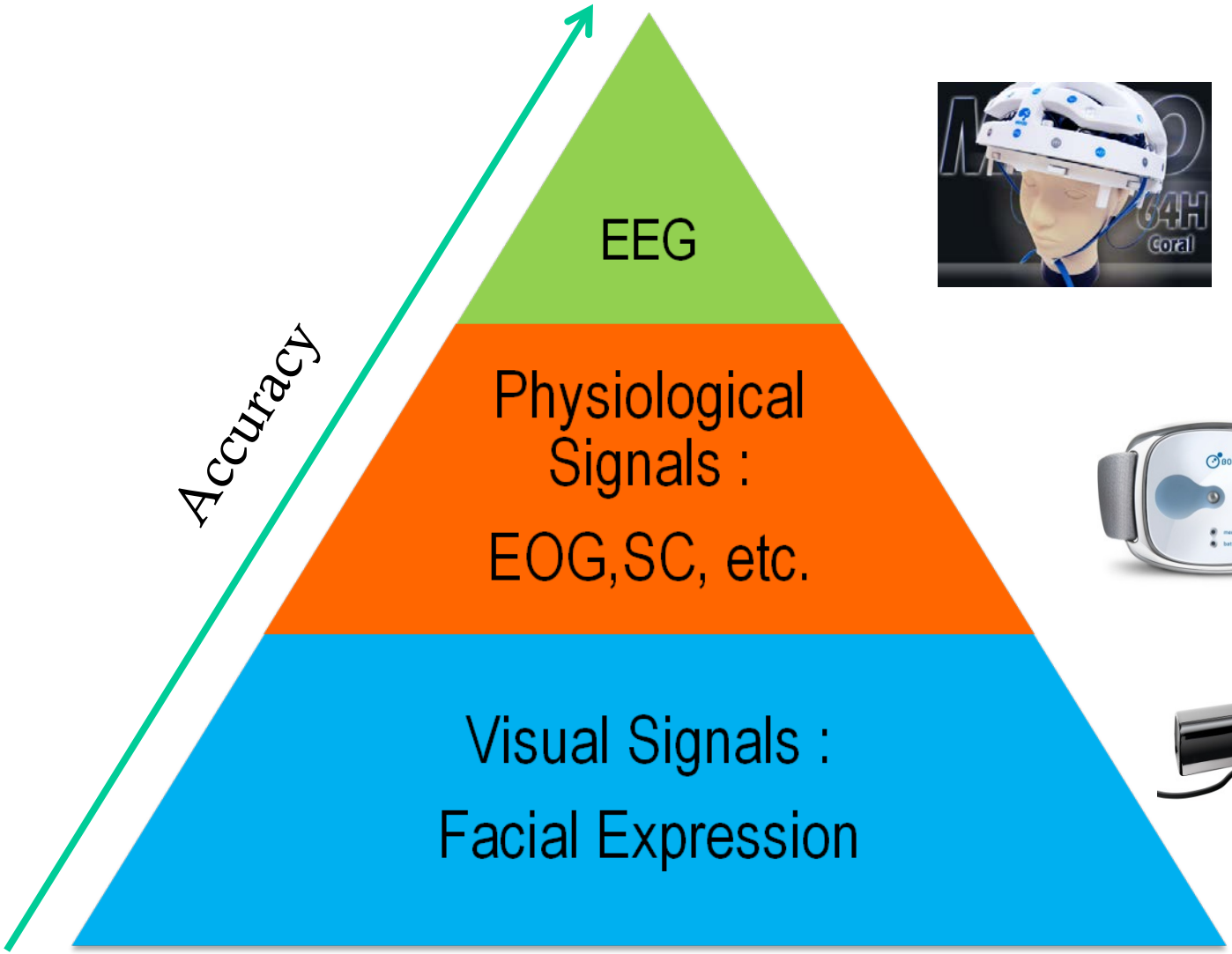
PERCLOS is used as a measure for detecting driver fatigue

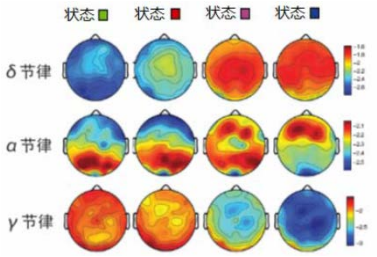
PERCLOS is the percentage of eyelid closure over the pupil over time and reflects slow eyelid closures rather than blinks.

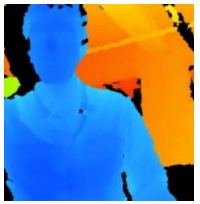
Merits and demerits of image based vigilance estimation:

- Easy to be record
- Non-contact and lower cost
- Hard to predict drowsy driving
- Can't be used to disabled persons

Multiple sensors vs. detection accuracy

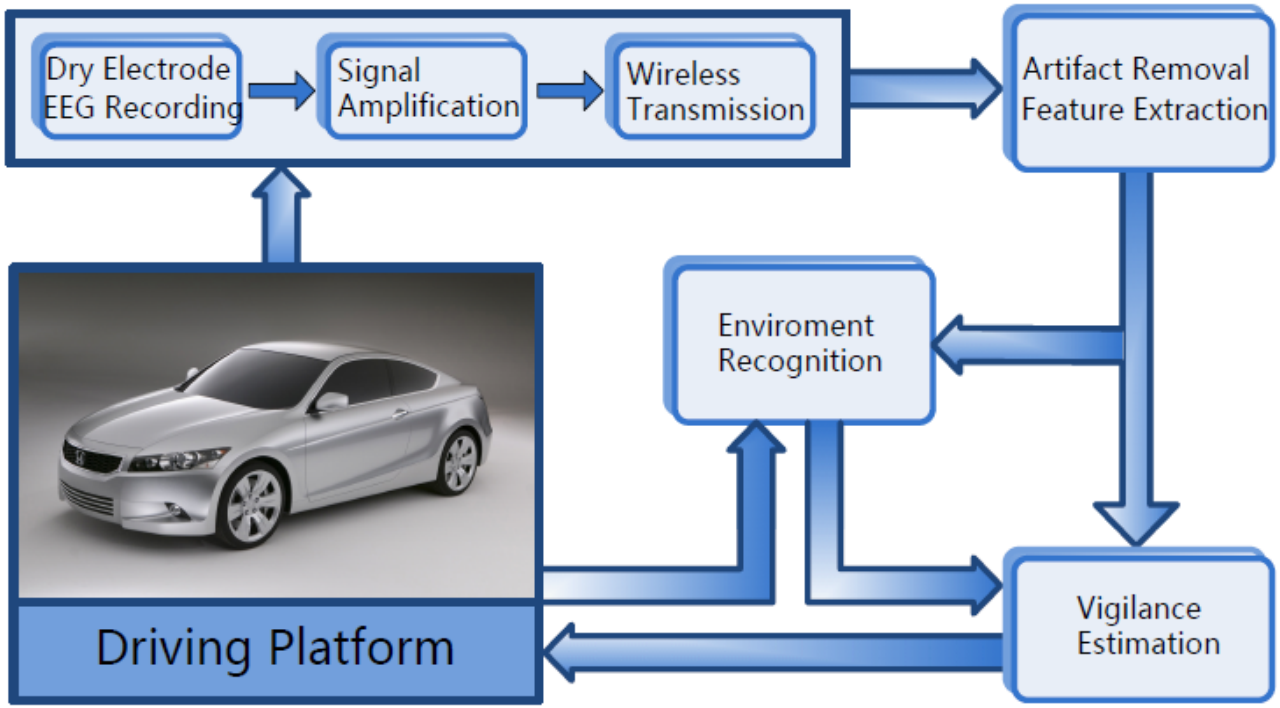






Driving fatigue detection using physiological signals

- Wearable dry-electrode EEG cap
- Low-power consumption amplifier VLSI chip
- Measure and grading standards for detecting drowsy driving
- Predicting drowsy driving with physiological signals such as EEG

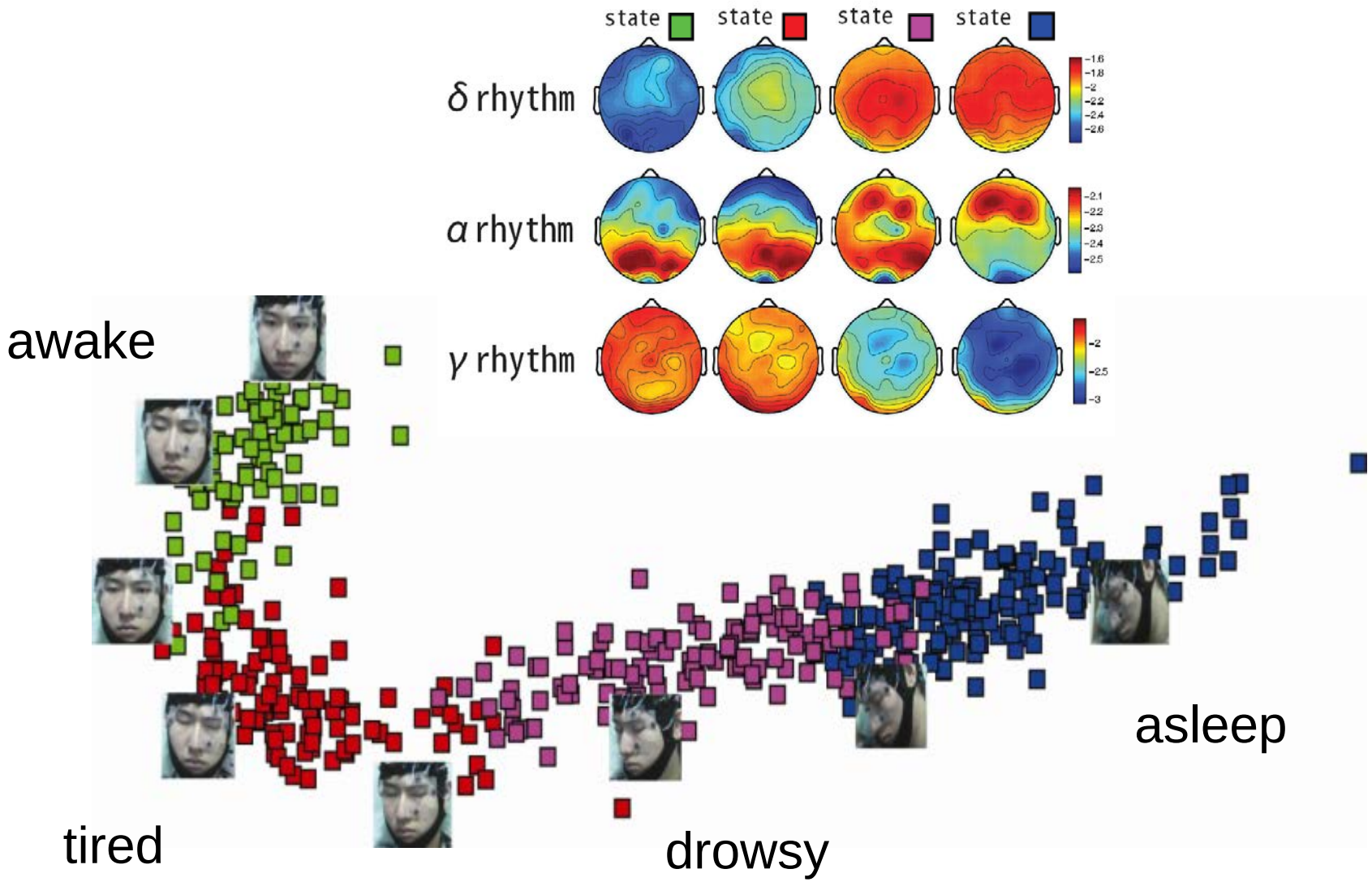


(Supported by NSFC, 973 Projects, and Science and Technology Commission of Shanghai Municipality since 2008)

Driving simulation system in our lab

Multiple sensors at driving simulation system

EEG feature and estimated vigilance states

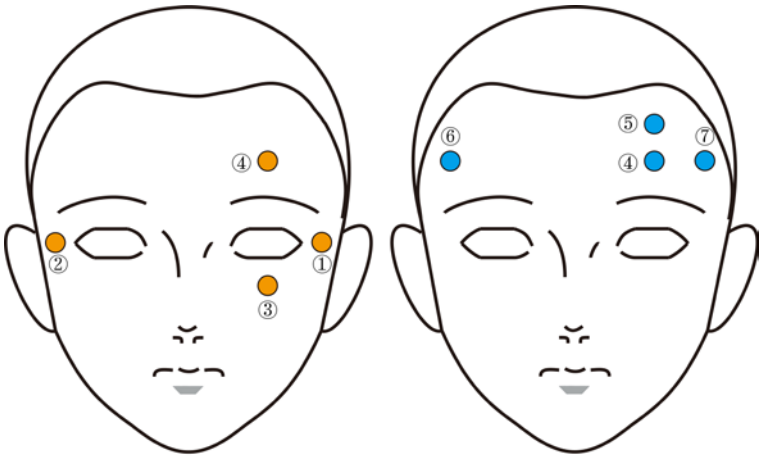


A Multimodal Approach to Estimating Vigilance Using EEG and Forehead EOG

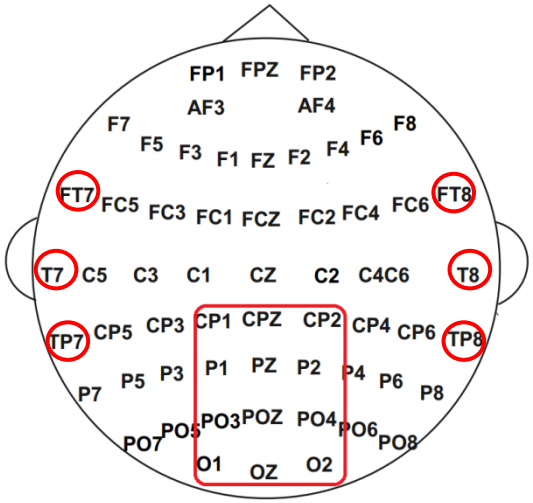
Wei-Long Zheng and Bao-Liang Lu, A Multimodal Approach to Estimating Vigilance Using EEG and Forehead EOG, Journal of Neural Engineering, 2017

Experimental Setup and Vigilance Annotations

- Electrode placements of EEG and EOG



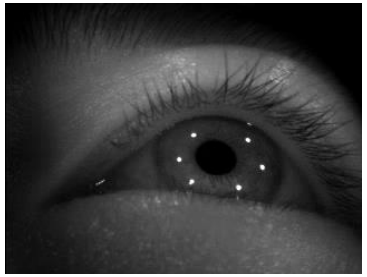
(a) Traditional EOG (b) Forehead EOG



- Vigilance Annotations

$$PERCLOS_{ETG} = \frac{blink + CLOS}{interval} \quad (1)$$

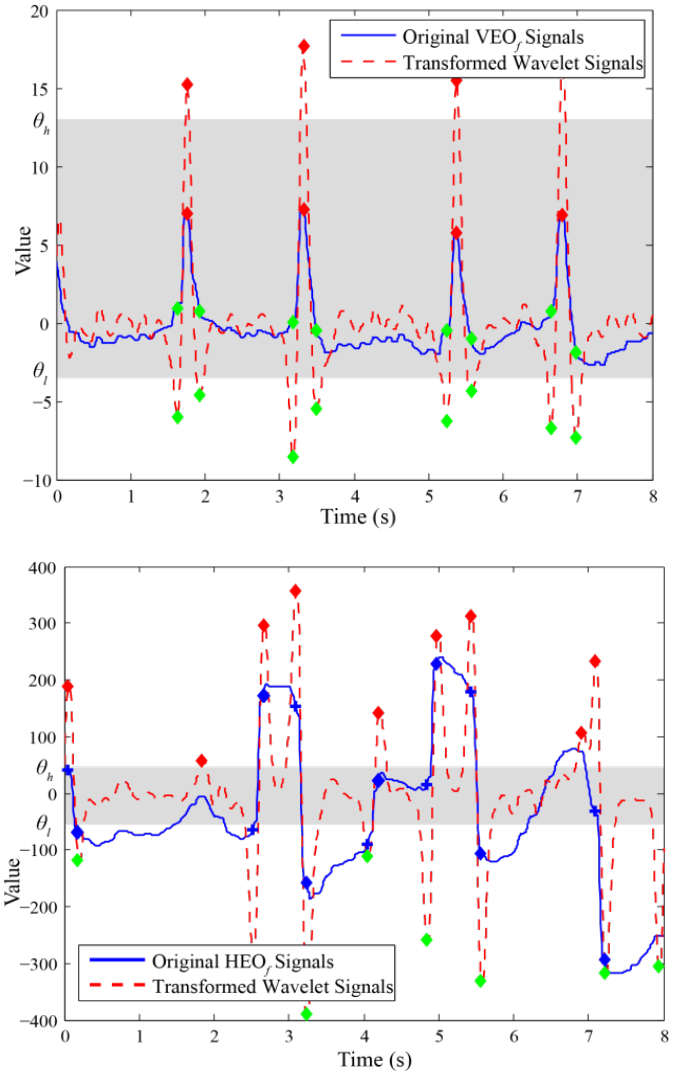
$$interval = blink + fixation + saccade + CLOS \quad (2)$$



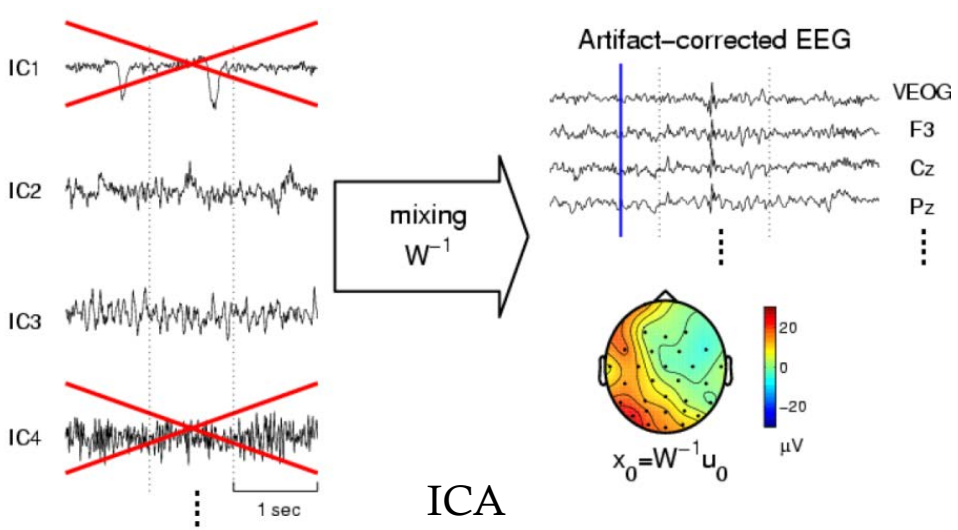
Feature Extraction for EOG

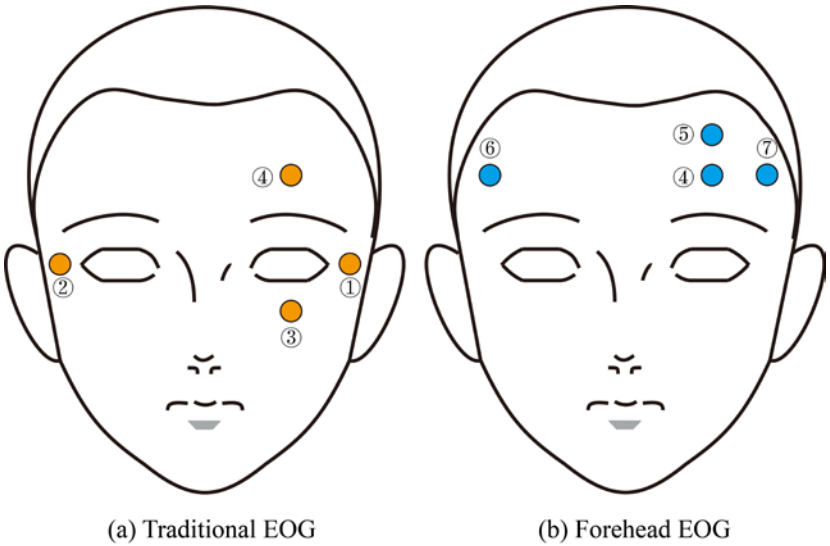
- Blink and saccade detection using continuous wavelet transform

Group	Extracted Features
Blink	maximum/mean/sum of blink rate maximum/minimum/mean of blink amplitude, mean/maximum of blink rate variance and amplitude variance power/mean power of blink amplitude blink numbers
Saccade	maximum/minimum/mean of saccade rate and saccade amplitude, maximum/mean of saccade rate variance and amplitude variance, power/mean power of saccade amplitude, saccade numbers
Fixation	mean/maximum of blink duration variance and saccade duration variance maximum/minimum/mean of blink duration and saccade duration.

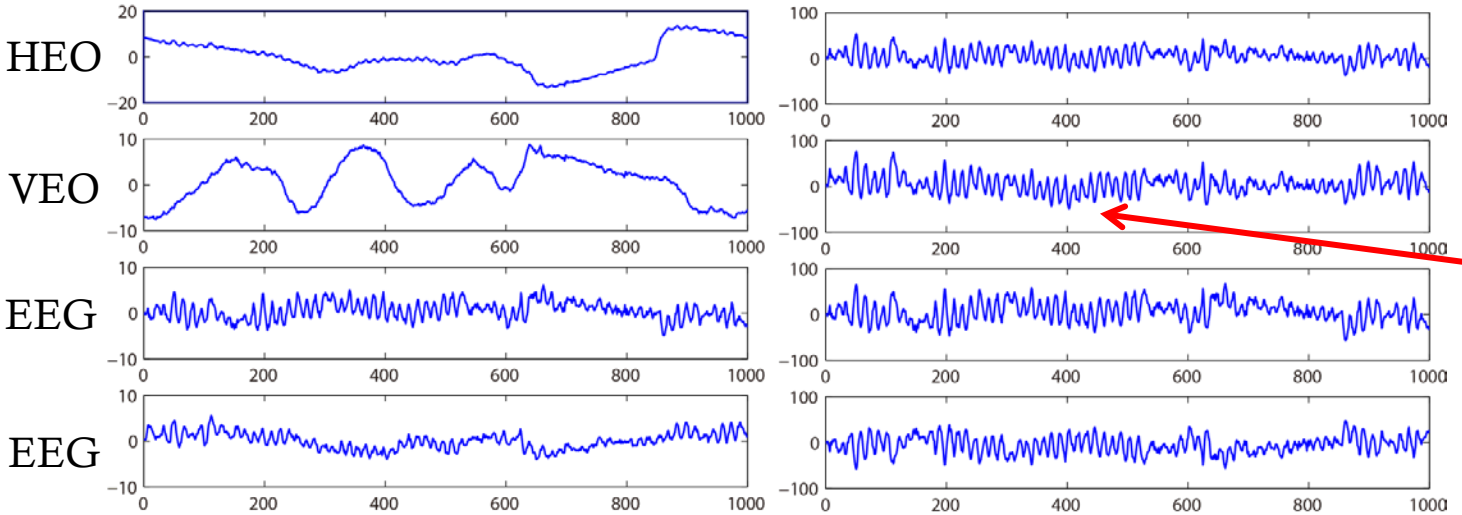


Extracting EEG from Forehead Electrodes





(a) Traditional EOG (b) Forehead EOG



(a) Decomposed ICA Components (b) Reconstructed Forehead EEG

Strong Alpha activities are verified under eye closure conditions.

Incorporating Temporal Dependency

- Vigilance is a dynamic changing process because the intrinsic mental states of users involve temporal evolution.
- Continuous conditional random field (CCRF)
- Continuous conditional neural field (CCNF)

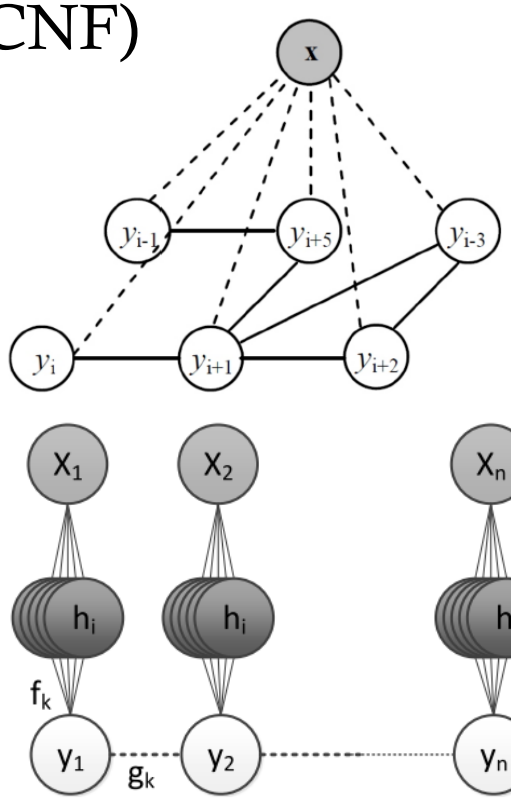
$$P(\mathbf{y}|\mathbf{x}) = \frac{exp(\Psi)}{\int_{-\infty}^{\infty} exp(\Psi) d\mathbf{y}},$$

$$\Psi = \sum_{i} \sum_{k=1}^{K_1} \alpha_k f_k(y_i, \mathbf{x}_i, \boldsymbol{\theta}_k) + \sum_{i,j} \sum_{k=1}^{K_2} \beta_k g_k(y_i, y_j)$$

$$f_k(y_i, \boldsymbol{x}_i, \boldsymbol{\theta}_k) = -(y_i - h(\boldsymbol{\theta}_k, \boldsymbol{x}_i))^2, and$$

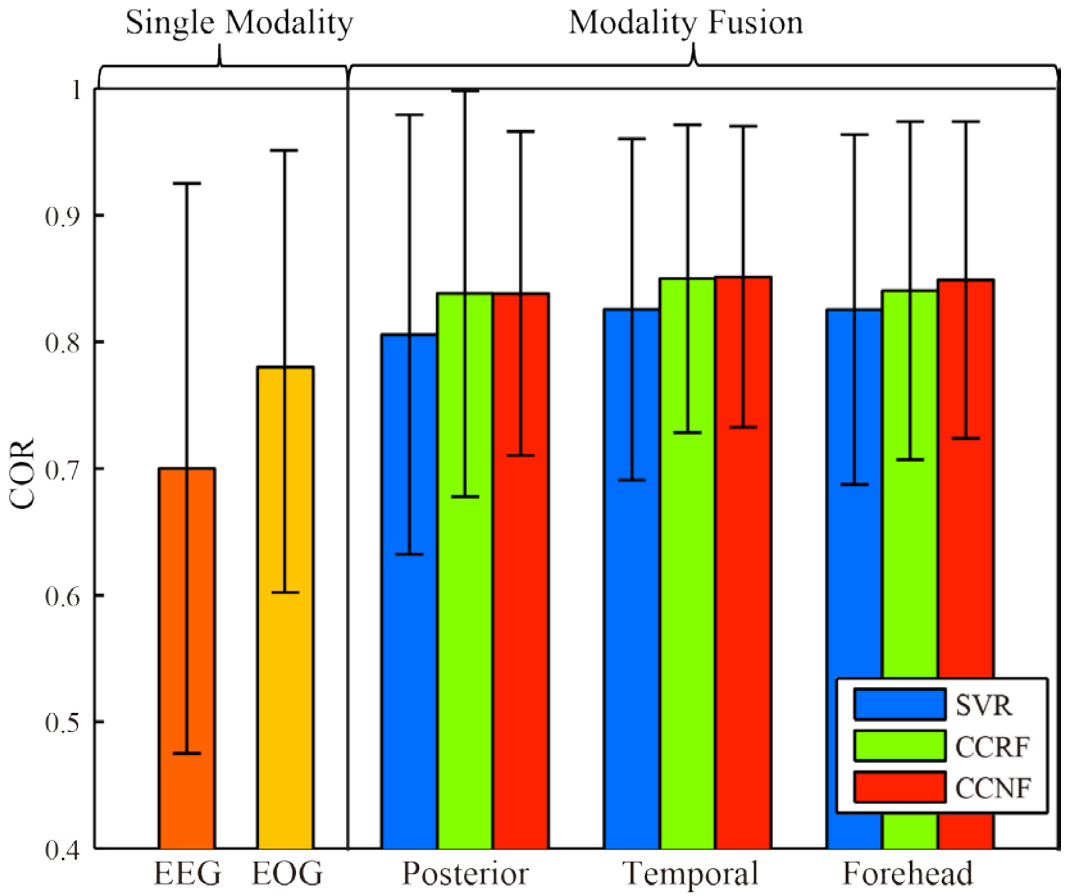
$$h(\boldsymbol{\theta}, \boldsymbol{x}_i) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \boldsymbol{x}_i}},$$

$$g_k(y_i, y_j) = -\frac{1}{2} S_{i,j}^{(k)} (y_i - y_j)^2$$



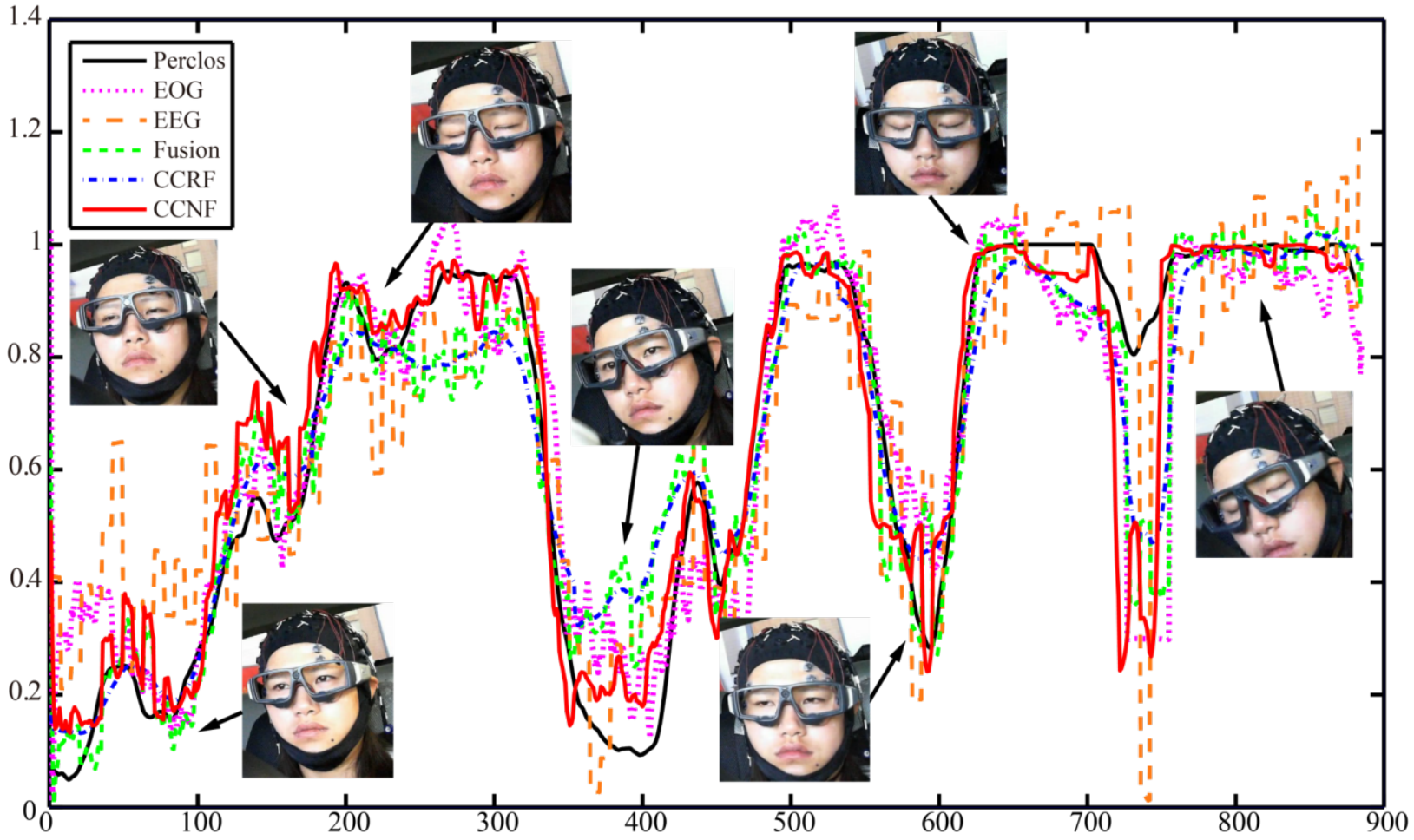
Experimental Results

- For the forehead setup, the mean COR/RMSE of SVR, CCRF, and CCNF are 0.83/0.10, 0.84/0.10, and **0.85/0.09,** respectively. The CCNF achieves the best performance with higher accuracies and lower standard deviations.



Experimental Results (2)

- The predictions from our proposed approaches are almost consistent with the true subjects' behaviors and cognitive states.

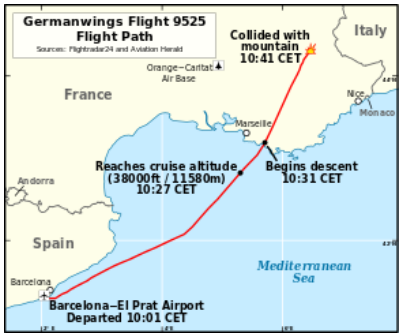


Applications in my Lab

- Driving fatigue detection
- Affective Brain-Computer Interaction (aBCI）

Germanwings Flight 9525

- On 24 March 2015, the aircraft, an Airbus A320211, crashed 100 kilometres northwest of Nice in the French Alps. All 144 passengers and six crew members were killed!
- The crash was deliberately caused by the copilot, Andreas Lubitz, who had previously been treated for suicidal tendencies and been declared "unfit to work" by a doctor.
- The European Federation of Psychologists' Associations (EFPA) issued a statement supporting psychological testing in the selection of pilots.



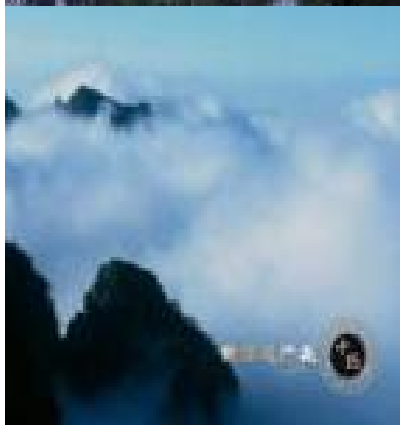
Emotion Recognition

- Traditional methods for emotion recognition
  - facial expression, voice, gesture
  - advantage: easy to obtain signals
  - disadvantage: not reliable to detect emotions; cannot be used for people with disabilities
- EEG-based method for emotion recognition
  - advantage: reliable to detect emotions; wide applications
  - disadvantage: need special equipment to get EEG signals

The selected films as emotion stimuli

Happy

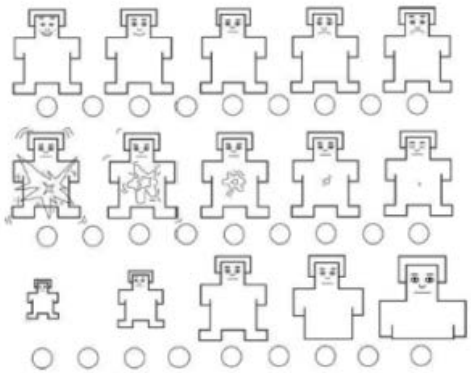
Neutral



Sad

Fear

Process of the experiment



positive or negative movie clip

Hint of Start
• 5sec

Movie clip
• 4min

Self-assessment
• 45sec

Rest
• 15sec

Session 1 → Session 2 → Session 3 → … → Session N

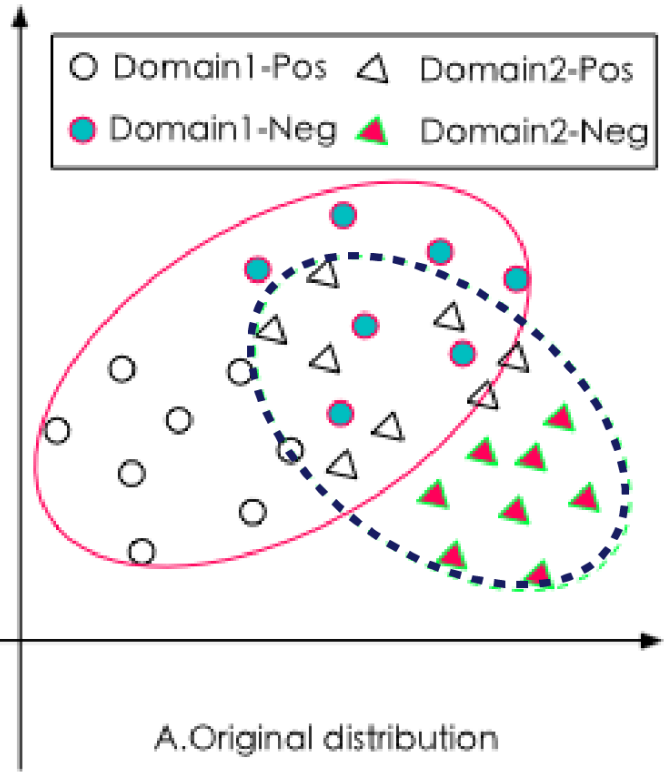
Personalizing EEG-based Affective Models with Transfer Learning

Wei-Long Zheng and Bao-Liang Lu, Personalizing EEG-based Affective Models with Transfer Learning, Proc. of the 25th International Joint Conference on Artificial Intelligence (IJCAI-16).

Challeges of Subject Transfer

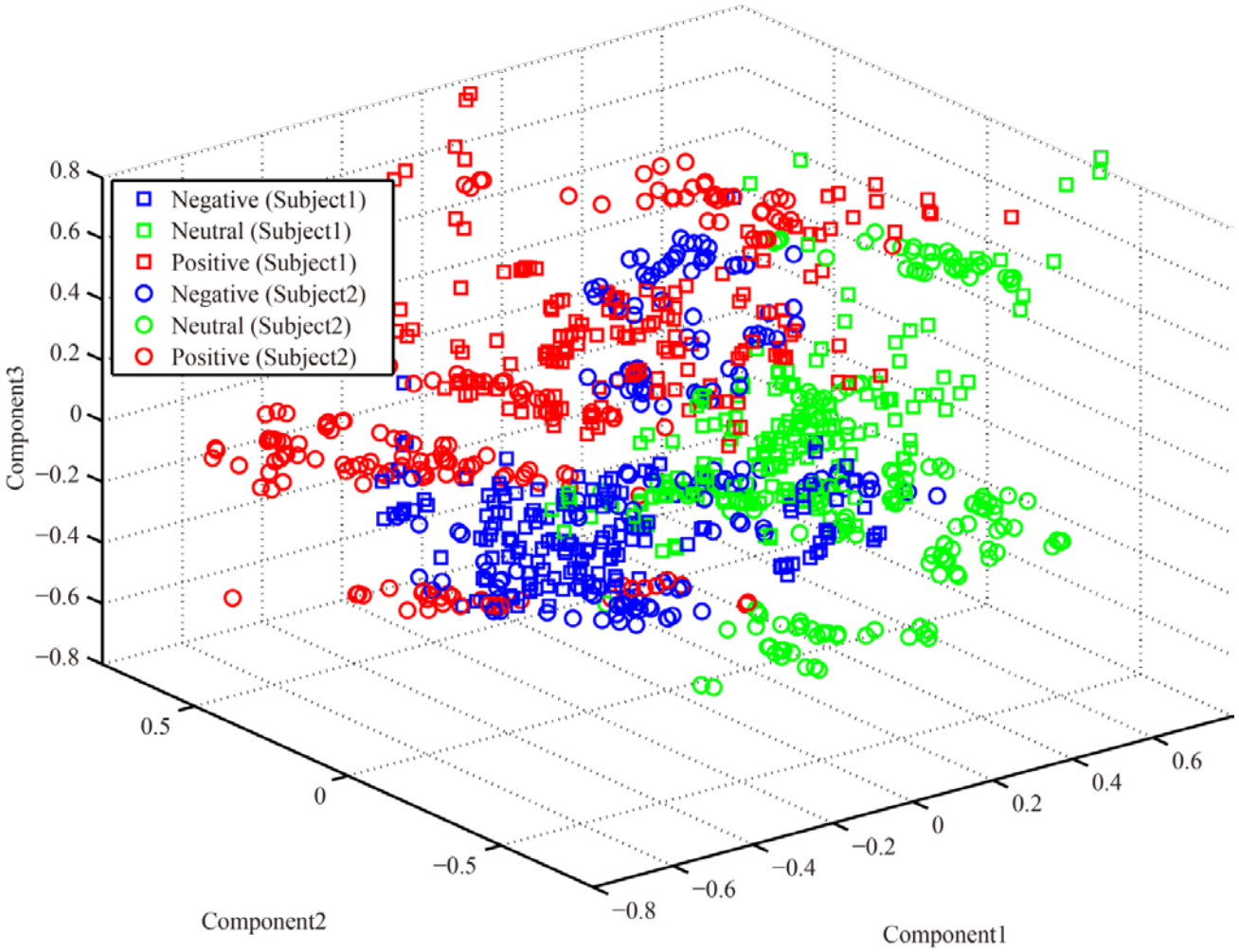
The feature distributions of EEG data of source subjects and target subjects are not independently and identically distributed (i.i.d.) due to:

- The structural and functional variabili between subjects
- The non-stationary nature of EEG signals
- The inherent changes of environmenta variables

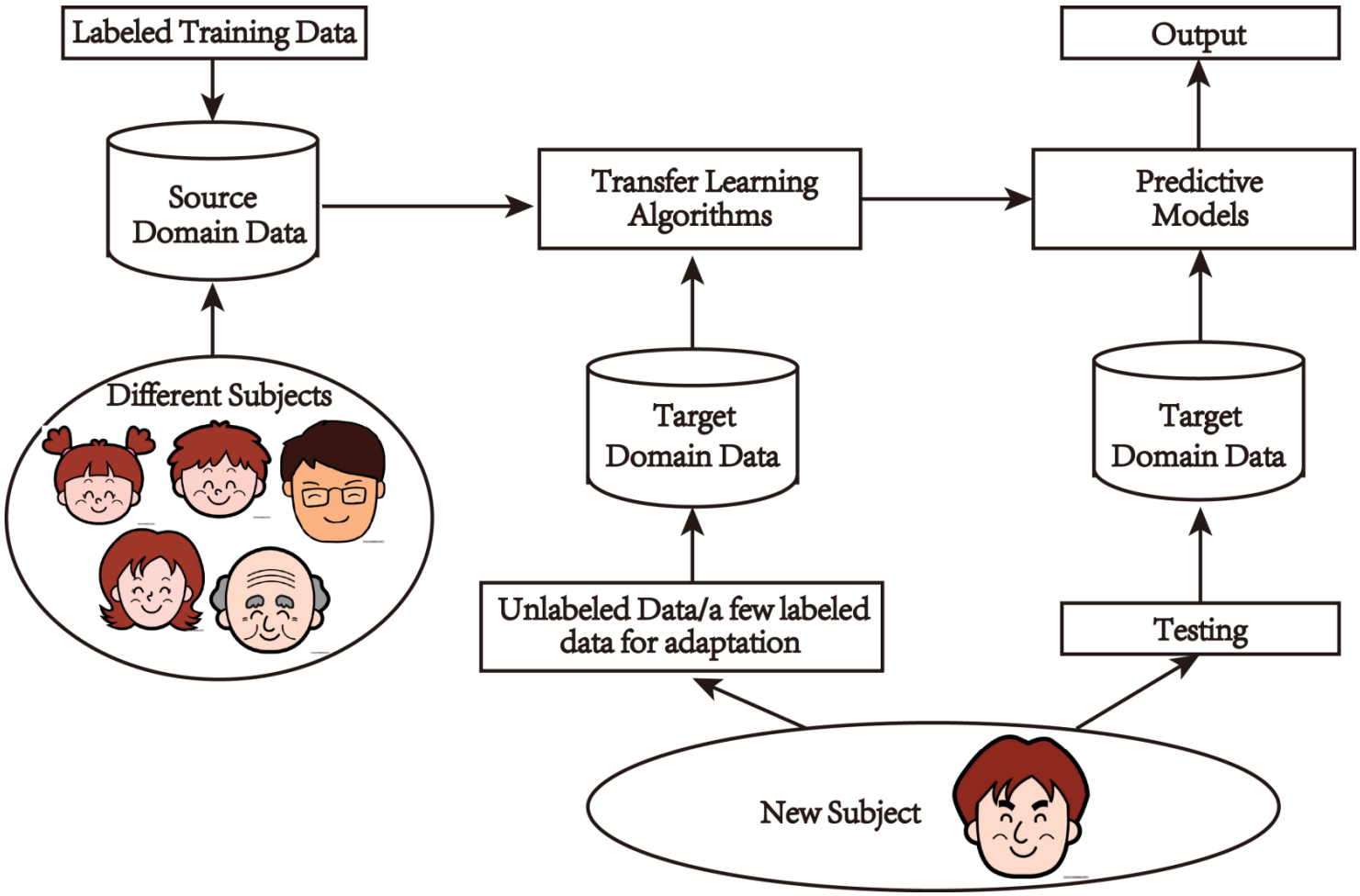


Covariate Shift Challenges

- Individual Differences Across Subjects and Sessions
- Non-stationary Characteristics of EEG



Transfer learning framework



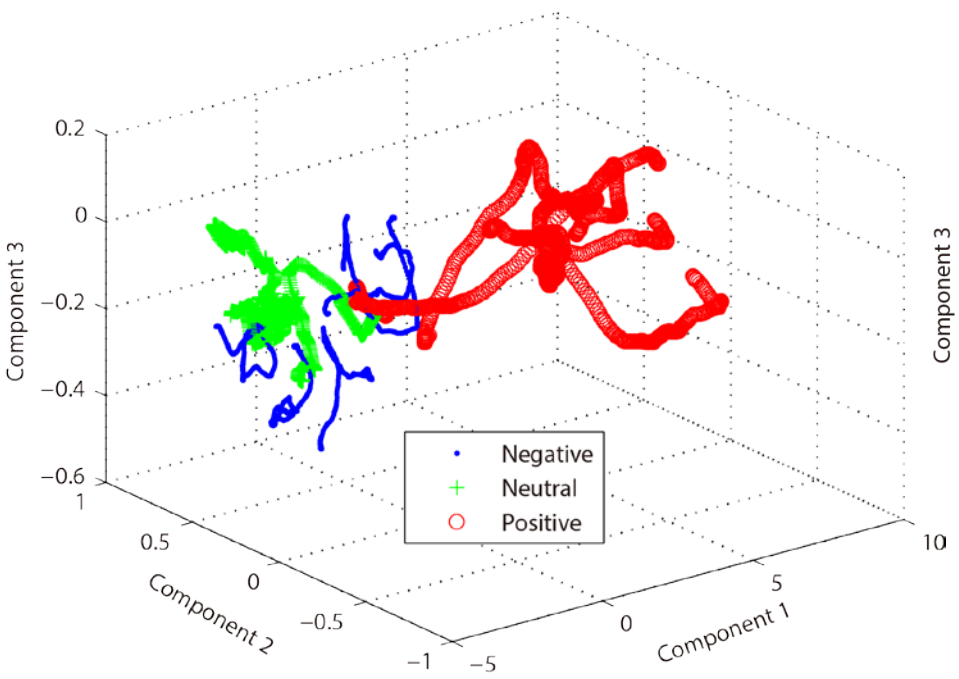
Transfer Component Analysis

Although the distributions of source domain and target domain in high dimensional space are different, we may find a low dimensional manifold space where the distributions of both domains are similar.

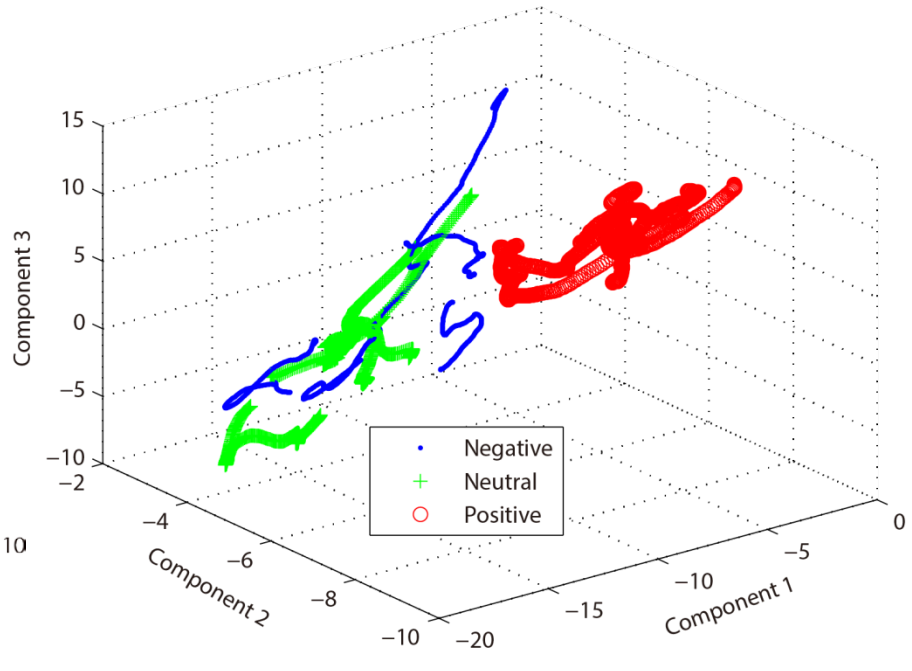
TCA and KPCA try to learn a set of common transfer components underlying both the source domain and the target domain. When projected to this subspace, the difference of feature distributions of both domains can be reduced.

$$P(\phi(X_S)) \approx P(\phi(X_T))$$
$$P(Y_S|\phi(X_S)) \approx P(Y_T|\phi(X_T))$$

Feature visualization in 3D latent space



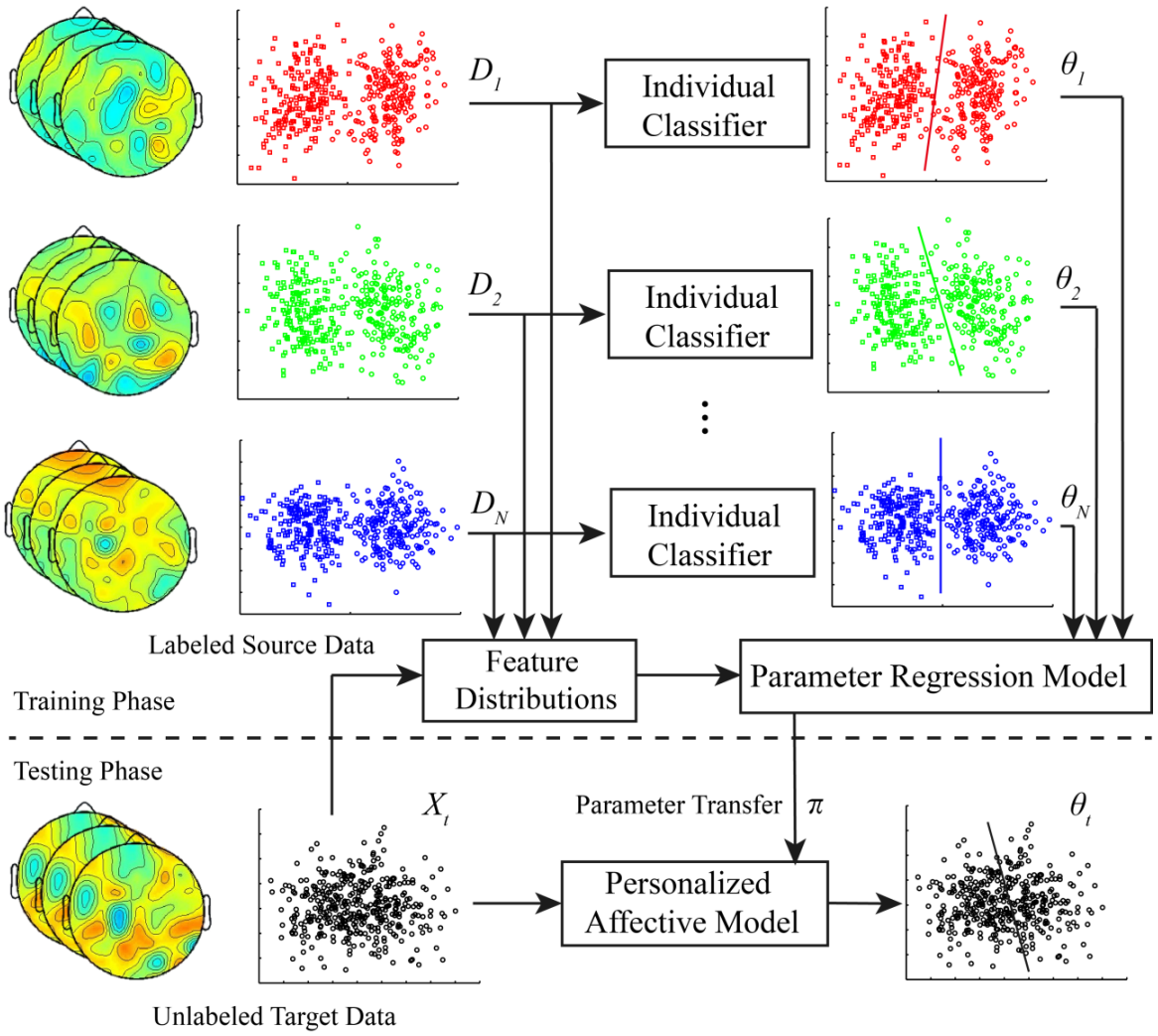
TCA



KPCA

The neural patterns of neutral and negative emotions are more similar to each other and it is easier to recognize positive emotions among these three emotions.

Transductive Parameter Transfer



Enver Sangineto, Gloria Zen, Elisa Ricci, and Nicu Sebe. We are not all equal: Personalizing models for facial expression analysis with transductive parameter transfer. In ACM International Conference on Multimedia, pages 357–366. ACM, 2014.

Transductive Parameter Transfer

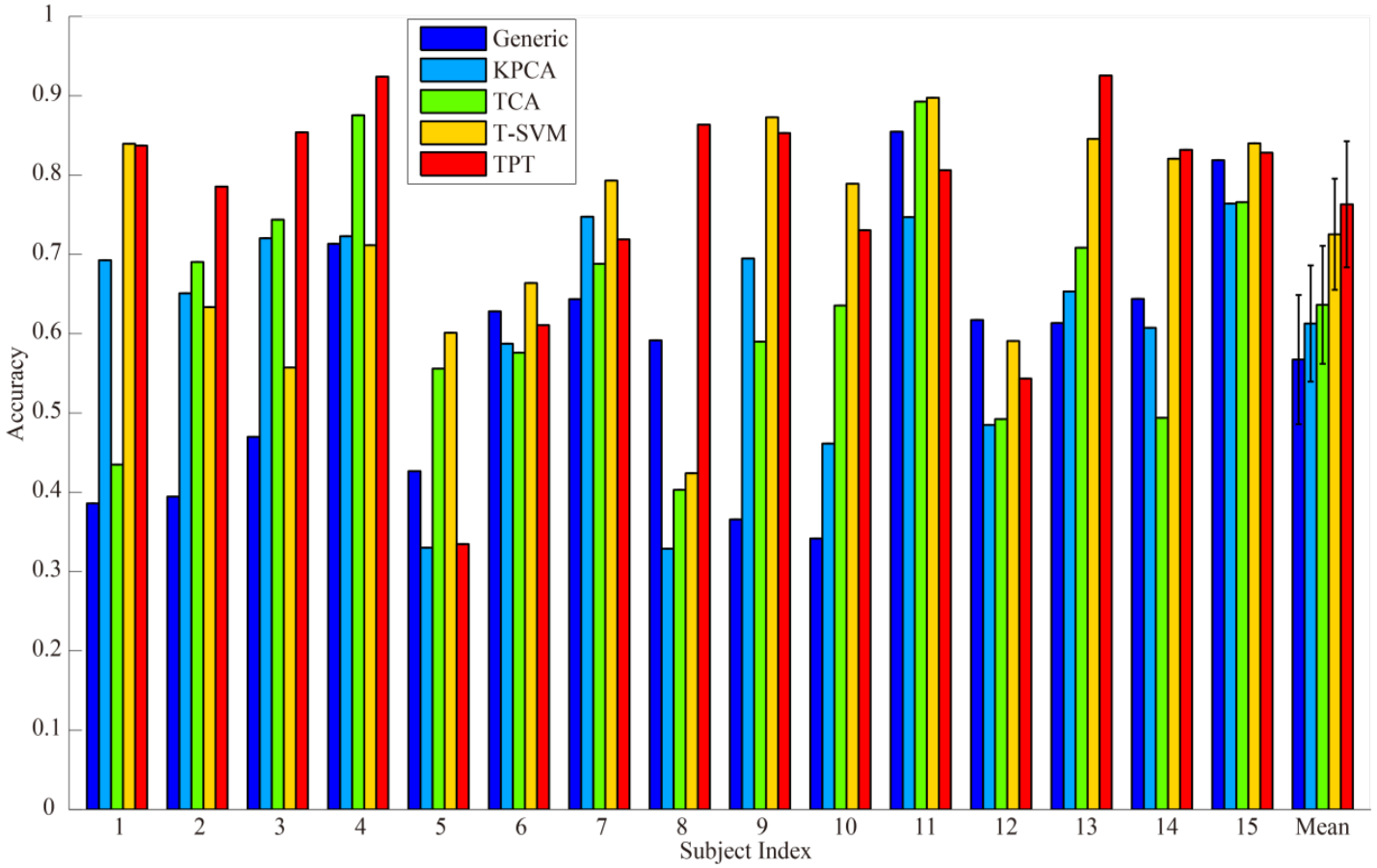
- Multiple individual classifiers are learned on each training dataset
- A regression function is trained to learn the relation between the data distributions and classifiers' parameter vectors.
- Finally, the target classifier is obtained using the target feature distribution and the distribution-to-classifier mapping.

Algorithm 3 TPT-based Subject Transfer

input : Source domain data sets $\mathcal{D}_1^s, ..., \mathcal{D}_N^s$, target domain data set X^t, and some regularization parameters for SVMs.
output : The parameter vector of target classifier: $\boldsymbol{w}_t, b_t$.

1: Construct individual classifiers: $\{\boldsymbol{\theta_i} = (\boldsymbol{w}_i, b_i)\}_{i=1}^N$.
2: Create a training set $\mathcal{T} = \{X_i^s, \boldsymbol{\theta_i}\}_{i=1}^N$.
3: Compute the kernel matrix $\boldsymbol{K}$, $\boldsymbol{K}_{ij} = \kappa(X_i^s, X_j^s)$.
4: Given K and $\mathcal{T}$, learn $f(\cdot)$ using multioutput support vector regression.
5: Compute $(\boldsymbol{w}_t, b_t) = f(X^t)$
6: **Return** $\boldsymbol{w}_t, b_t$.

Evaluation Results



Stats.	Generic	KPCA	TCA	T-SVM	TPT
Mean	0.5673	0.6128	0.6364	0.7253	**0.7631**
Std.	0.1629	0.1462	0.1488	**0.1400**	0.1589

智能机器与人类沟通时如何识别情绪变化？中国团队给出答案

麻省理工科技评论 2016-01-27 12:12

没有人知道怎样通过观察脑电波来识别人类的情绪状态，直到机器学习算法的出现。

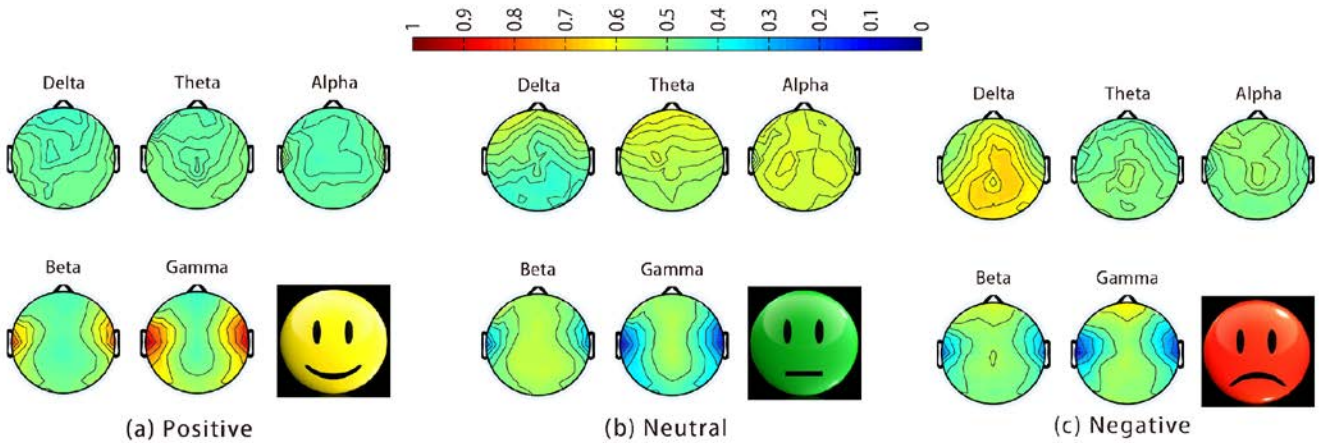
人们在进行交谈时，总是对彼此的情绪状态十分敏感。的确，大多数人希望，他们的交流的对象能够考虑到自己的情绪状态。因为在这种情况下进行的交流往往会更加有效。

所以，如果电脑要有效地与人类沟通的话，那他们将需要用一些方式来重复这个技巧，并评估对话者的情绪状态。了解一个人的精神状态是处于正面还是负面，可以使电脑的反馈质量大相径庭。

但如何做到这一点呢？评估人类精神状态的一种方法是使用脑电图仪器分析大脑中产生的电子信号。这种方法能够可靠地揭示出大脑状态的各个方面，比如其专心程度或焦点等。

然而，大脑的情绪状态非常复杂，之前就有很多研究指出，与特定情绪相关的脑电波似乎会随着时间的推移而改变。因此，还没有人找到一种能够利用脑电波清晰可靠地识别情绪状态的方法。

但在上海交通大学的郑伟龙（音译，Wei-Long Zheng）及其同事的努力下，这一切出现了转机。



机器学习的主要杂志

- IEEE Trans. Pattern Analysis and Machine Intelligence
- Machine Learning
- Journal of Machine Learning Research
- IEEE Trans. Neural Networks and Learning Systems
- Neural Computation
- Artificial Intelligence
- Neural Networks
- Journal of Artificial Intelligence Research
- Pattern Recognition
- Neurocomputing

机器学习的主要会议

- International Conference on Machine Learning
- Neural Information Processing Systems (NIPS)
- International Joint Conference on Artificial Intelligence(IJCAI)
- National Conference on Artificial Intelligence(AAAI)
- IEEE/INNS International Joint Conference on Neural Networks (IJCNN)
- International Conference on Neural Information Processing (ICONIP)
- International Conference on Artificial Neural Networks (ICANN)

数据挖掘的主要杂志

- IEEE Trans. Knowledge and Data Engineering
- ACM Trans. Information Systems
- Data Mining and Knowledge Discovery
- Journal of Management Information System

机器学习参考书

- 《机器学习》，Tom M. Michell著，机械工业出版社，2003
- 《模式分类》，Richard O. Duda, Peter E. Hart, David G. Stork著，机械工业出版社，2003
- 《神经网络与机器学习》，Simon Haykin著，机械工业出版社，2011

数据挖掘的主要会议

- ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)
- International Conference on World Wide Web (WWW)
- International Conference on Data Engineering (ICDE)
- IEEE International Conference on Data Mining

数据挖掘参考书

- 《数据挖掘导论》，Pang-Ning Tan, Michael Steinbach, Vipin Kumar著， 人民邮电出版社，2011
- 《数据挖掘：概念与技术》，Jiawei Han， Micheline Kamber著，机械工业出版社, 2011

并行程序设计参考书

- 《并行程序设计原理》，Calvin Lin, Lawrence Snyder著，机械工业出版社，2009

小结

- 机器学习的三种范式
- 监督学习的2种主要任务
- 学习的2个不同阶段
- 训练集与测试集
- 特征提取与特征选择

第二讲：人工神经网络

吕宝粮
计算机科学与工程系
电信楼群3号楼431
Tel: 021-3420-5422

bllu@sjtu.edu.cn
http://bcmi.sjtu.edu.cn/~blu

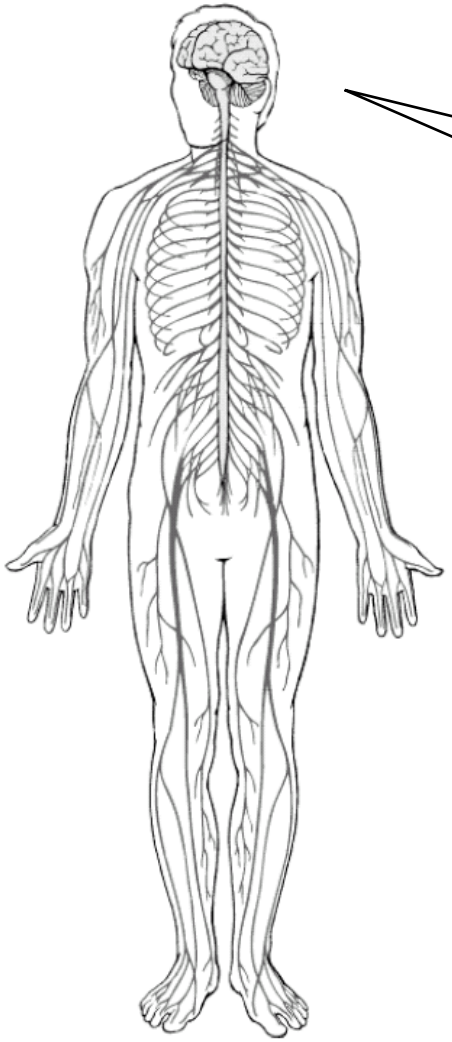
第一讲内容回顾

- 机器学习的三种范式
- 监督学习的2种主要任务
- 学习的2个不同阶段
- 训练集与测试集
- 特征提取与特征选择

第2讲内容

- 学习规则
- 基于记忆的学习
- 人工神经网络结构
- 误差修正学习
- 感知器
- 梯度下降法
- Delta学习规则

The Nervous System



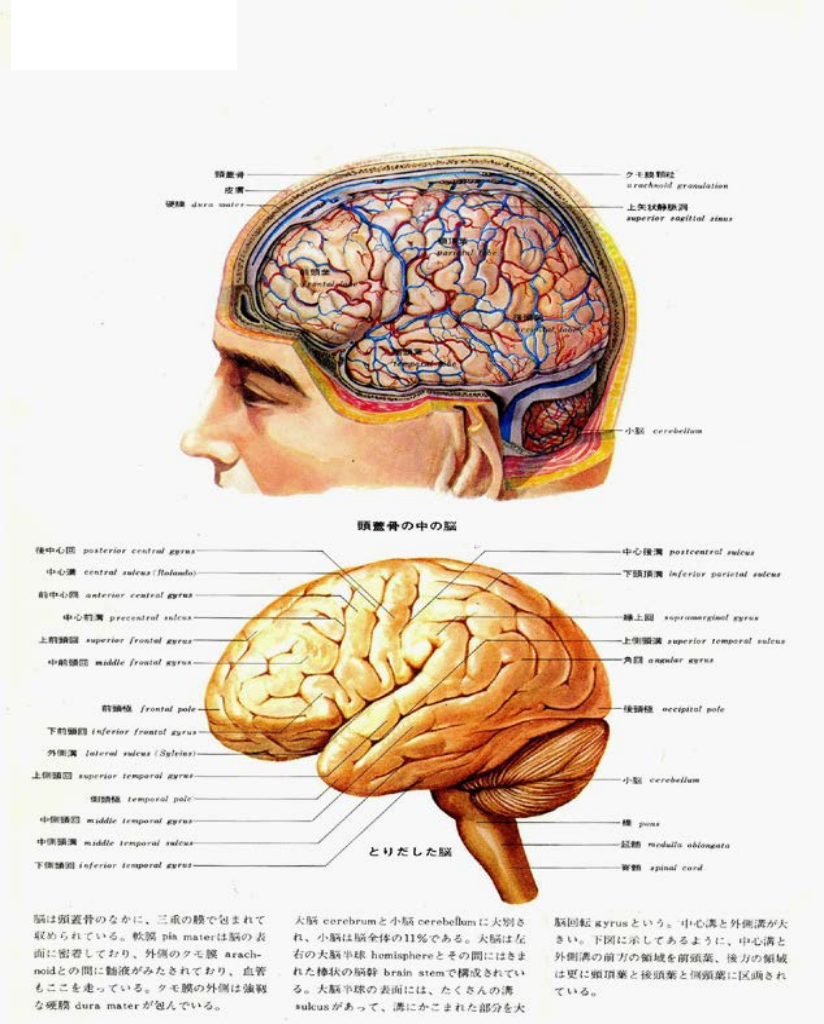
Central nervous system
(Brain & spinal cord)

中枢神经系统

Peripheral nervous system

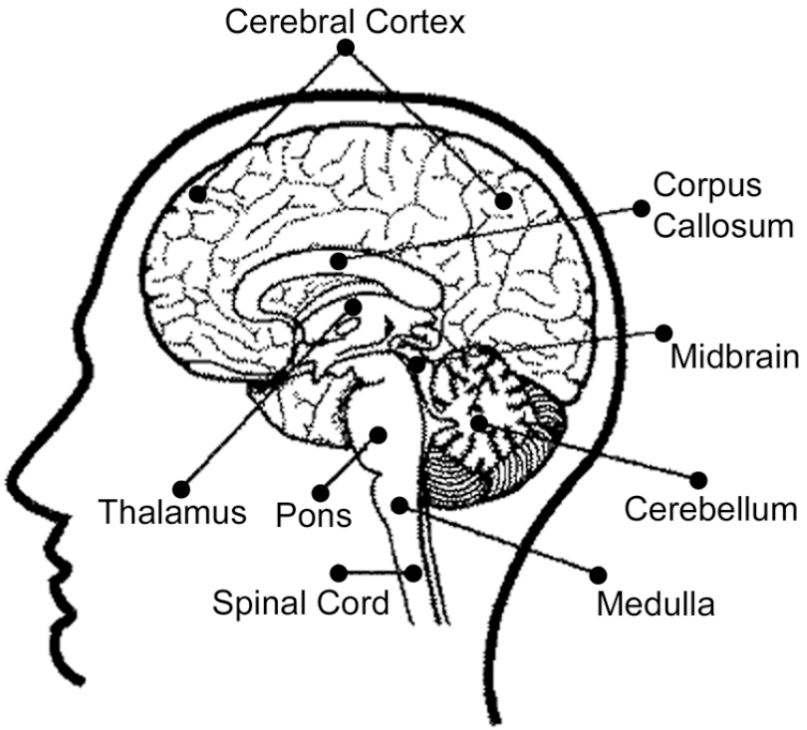
外周神经系统

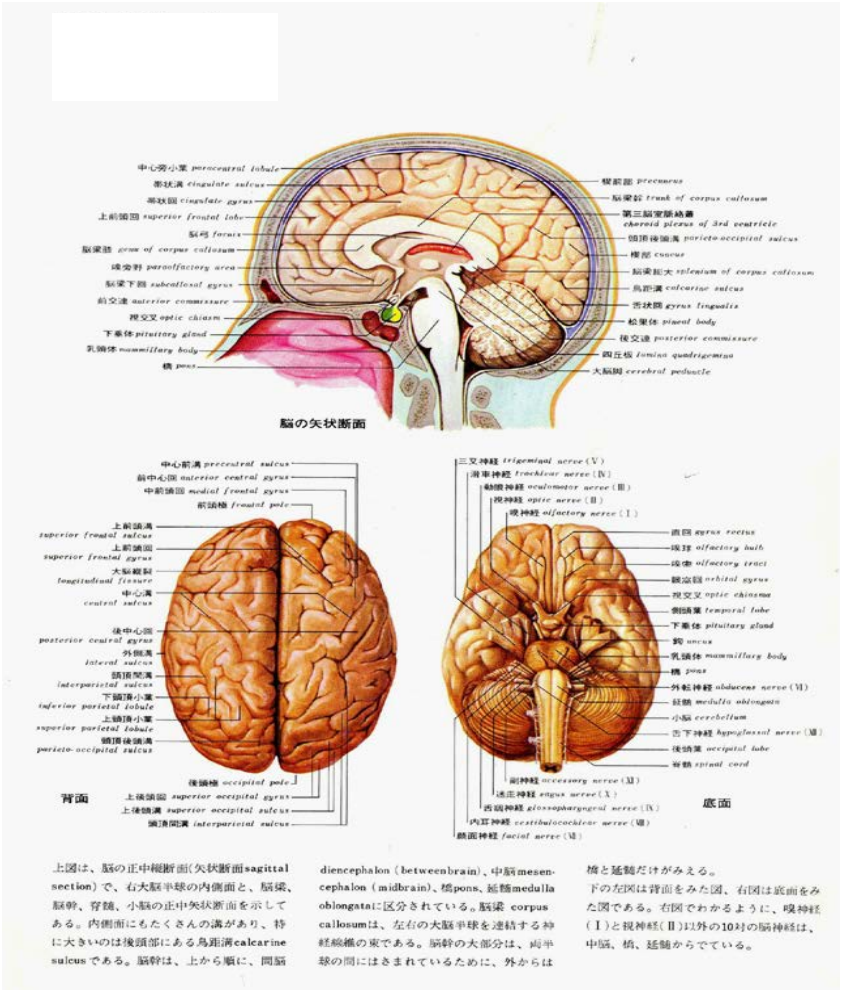
The Human Brain



- The brain contains about 100 billons neurons
- 10000 connections per neuron

Parts of the Human Brain

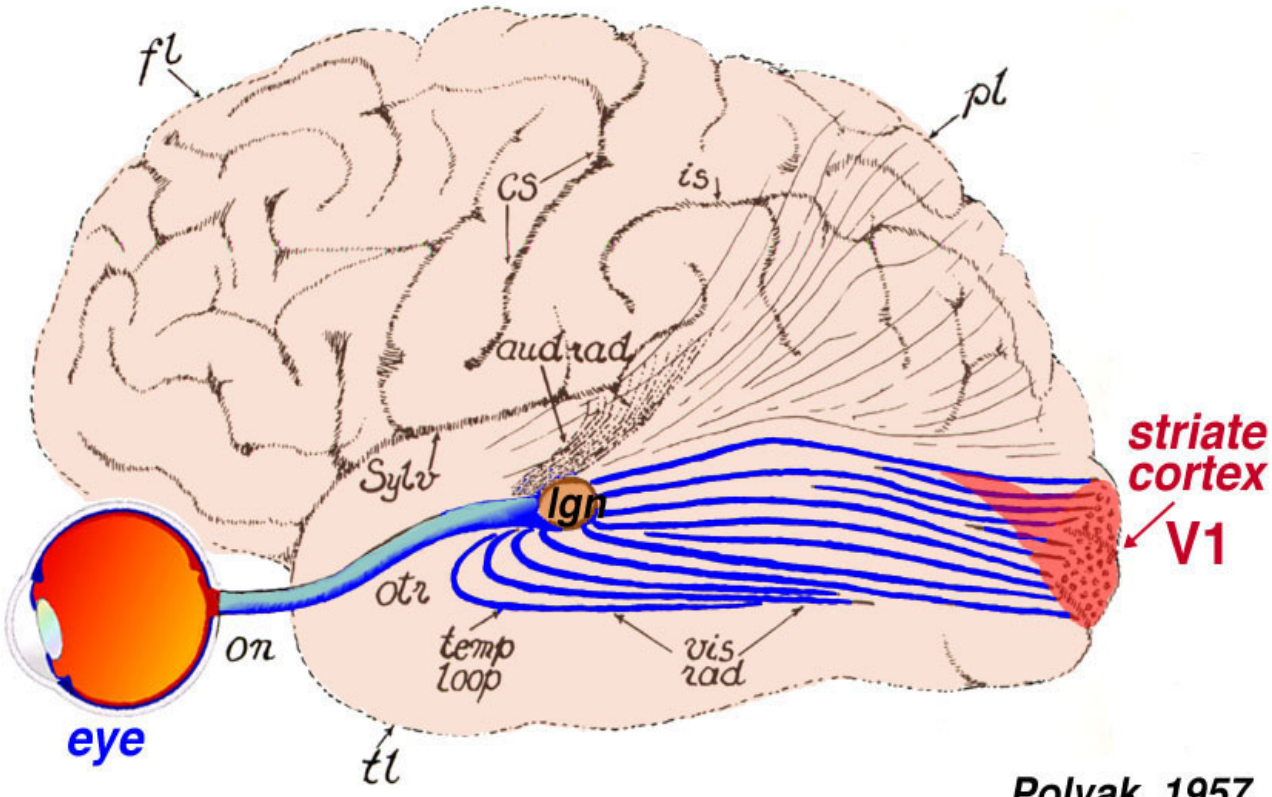


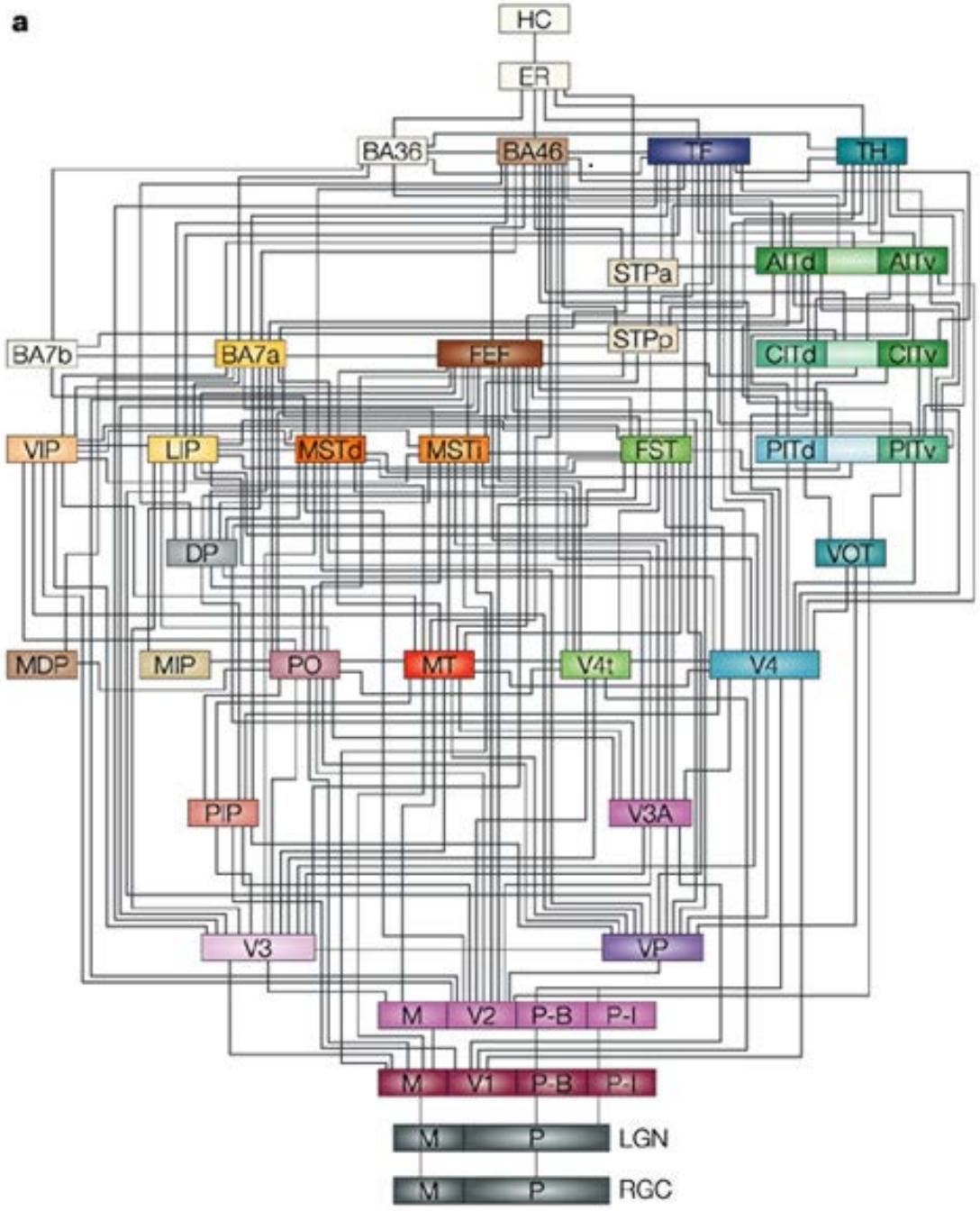


上図は、脳の正中縦断面(矢状断面sagittal section)で、右大脳半球の内側面と、脳梁、脳幹、脊髄、小脳の正中矢状断面を示してある。内側面にもたくさんの溝があり、特に大きいのは後頭部にある鳥距溝calcarine sulcusである。脳幹は、上から順に、間脳diencephalon(betweenbrain)、中脳mesencephalon(midbrain)、橋pons、延髄medulla oblongataに区分されている。脳梁 corpus callosumは、左右の大脳半球を連結する神経線維の束である。脳幹の大部分は、両半球の間にはさまれているために、外からは橋と延髄だけがみえる。
下の左図は背面をみた図、右図は底面をみた図である。右図でわかるように、嗅神経(Ⅰ)と視神経(Ⅱ)以外の10対の脳神経は、中脳、橋、延髄からでている。

视觉皮层（Visual Cortex）

- 至少有70～80%以上的外界信息是由视觉系统接受、处理和感知的；
- 人类大脑皮层约一半的区域参与视觉信息的分析。



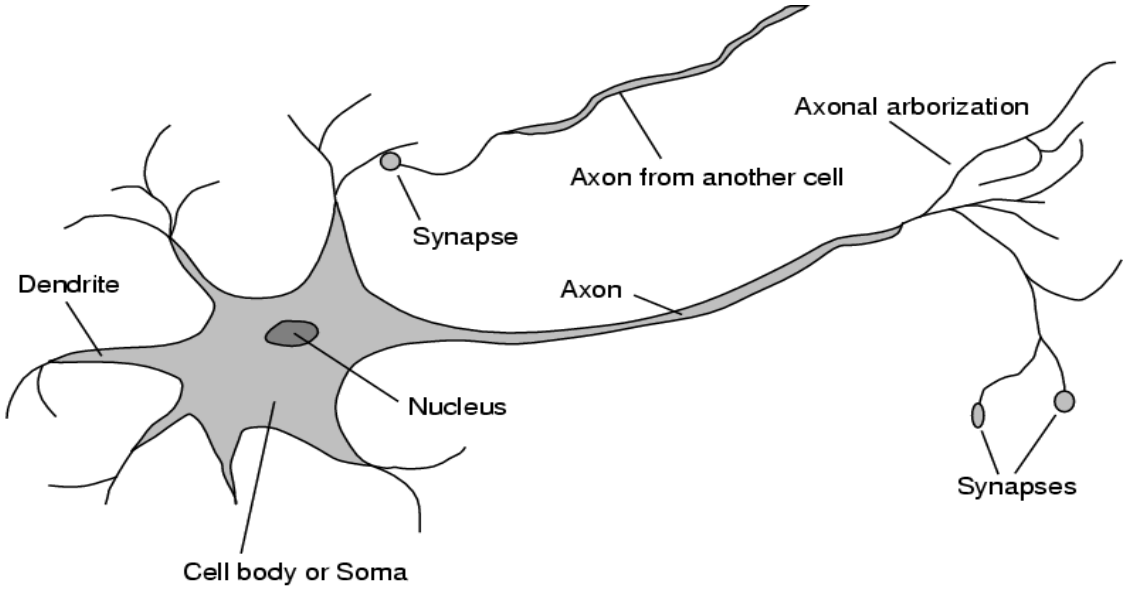


猴视觉皮层既平行又分级的30多个视觉皮层区域间连接的示意图

- 30多个视皮层区
- 10多个等级
- 305条视觉联系

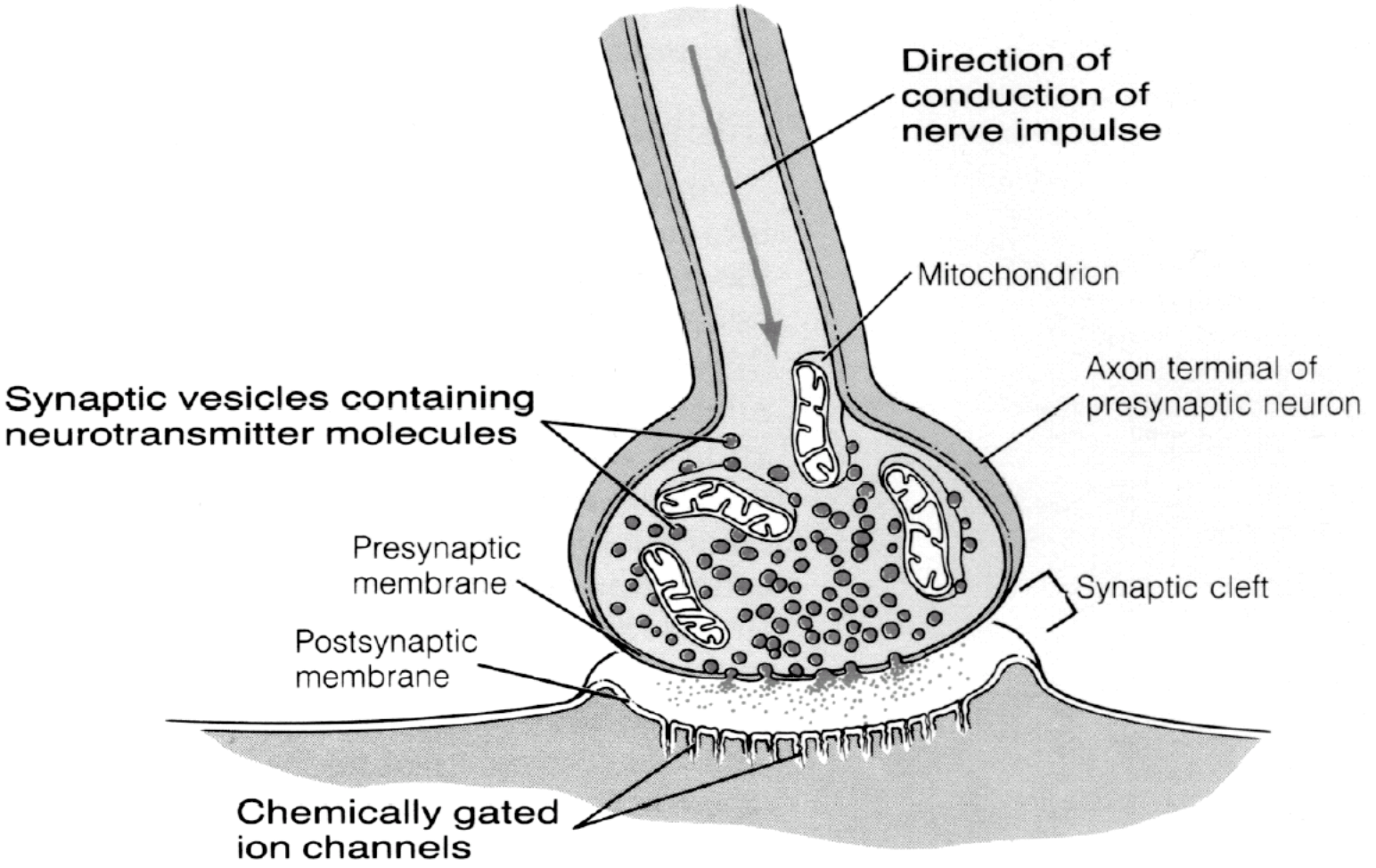
What is a Neuron ?

- Cells of the nervous system are called neurons (nerve cells)
- The human brain has about 100 billion neurons



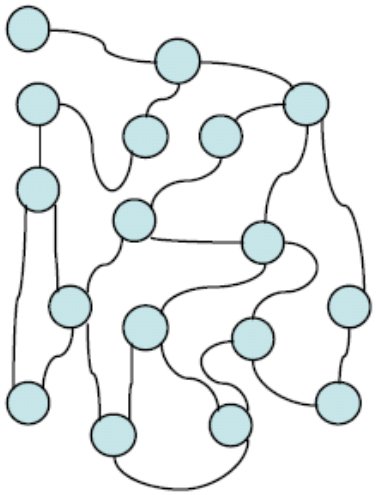
Axon : 轴突
Dendrite: 树突
Synapse: 突触

Synapse


Direction of conduction of nerve impulse
Mitochondrion
Axon terminal of presynaptic neuron
Synaptic vesicles containing neurotransmitter molecules
Presynaptic membrane
Synaptic cleft
Postsynaptic membrane
Chemically gated ion channels

What is a Neural Network ?

- A neural network is a network of many simple processors (neurons, units)
  - The units are connected by *connections*
  - Each connection has a number weight associated with it
  - The units operate only locally – on their weights and the inputs they receive through the connection



- Connectionism
- Parallel distributed processing
- Neural computation
- Neurocomputing

What is a Neural Network? (Cont.)

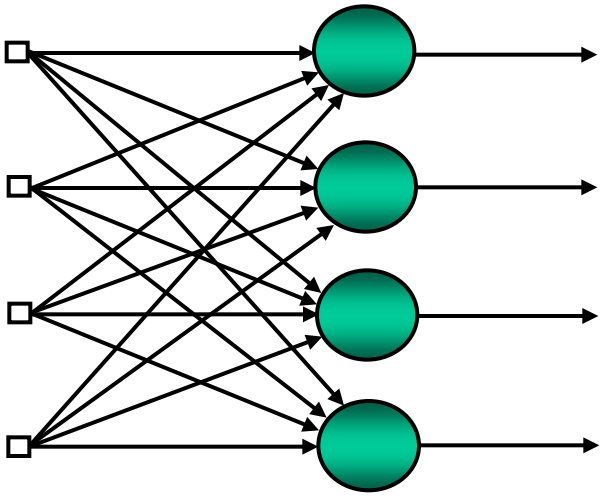
- A NN is a massively parallel distributed processor made up of simple processing unit, which has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two respects:
  - Knowledge is acquired by the network from its environment through *a learning process*
  - Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge

What is a Neural Network ? (Cont.)

- Some NNs are models of biological NNs and some are not
- The inspiration for the field of NNs came from the desire to produce artificial systems capable of sophisticated computations similar to those that the human brain routinely performs, and thereby also to enhance our understanding of the brain

Network Architectures

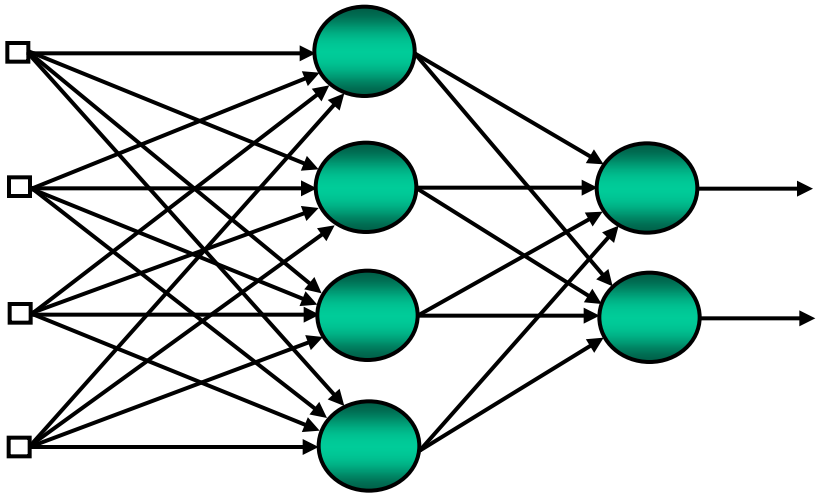
- Single-Layer Feedforward Networks
- Multilayer Feedforward Networks
- Recurrent Networks



Input layer

Output layer

Multilayer Feedforward Networks

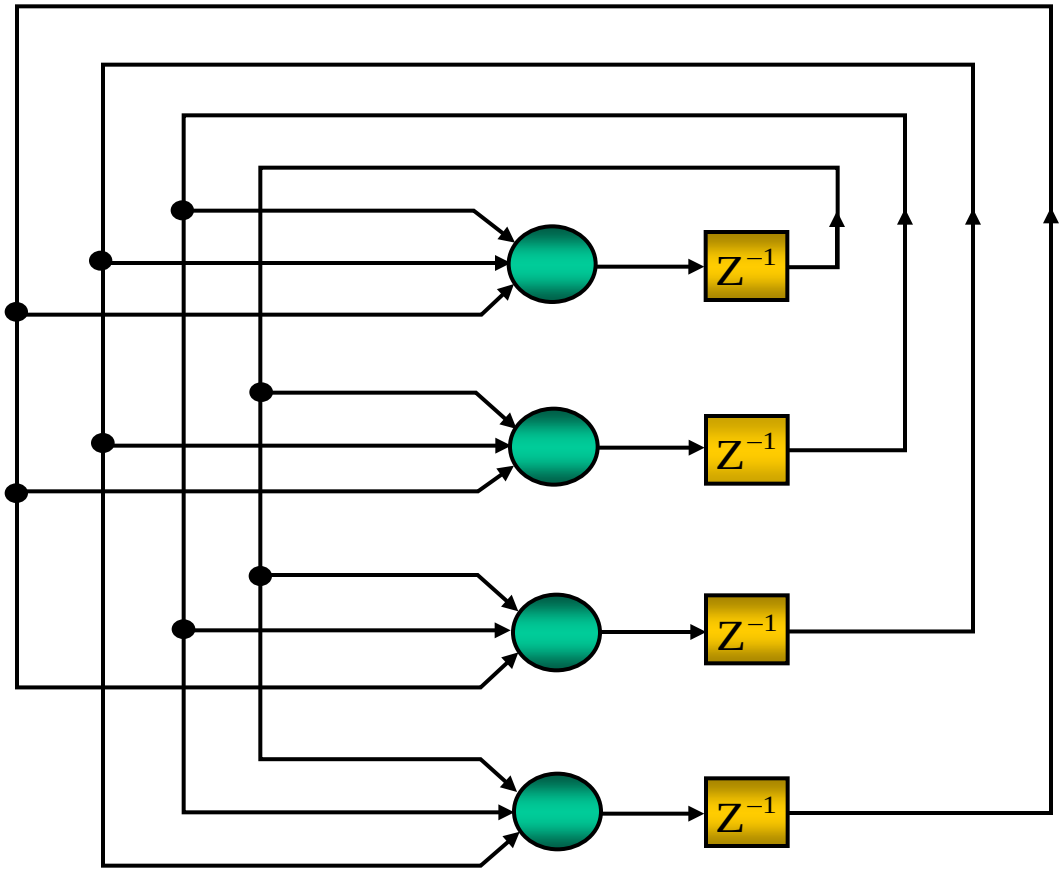


Input layer

Hidden layer

Output layer

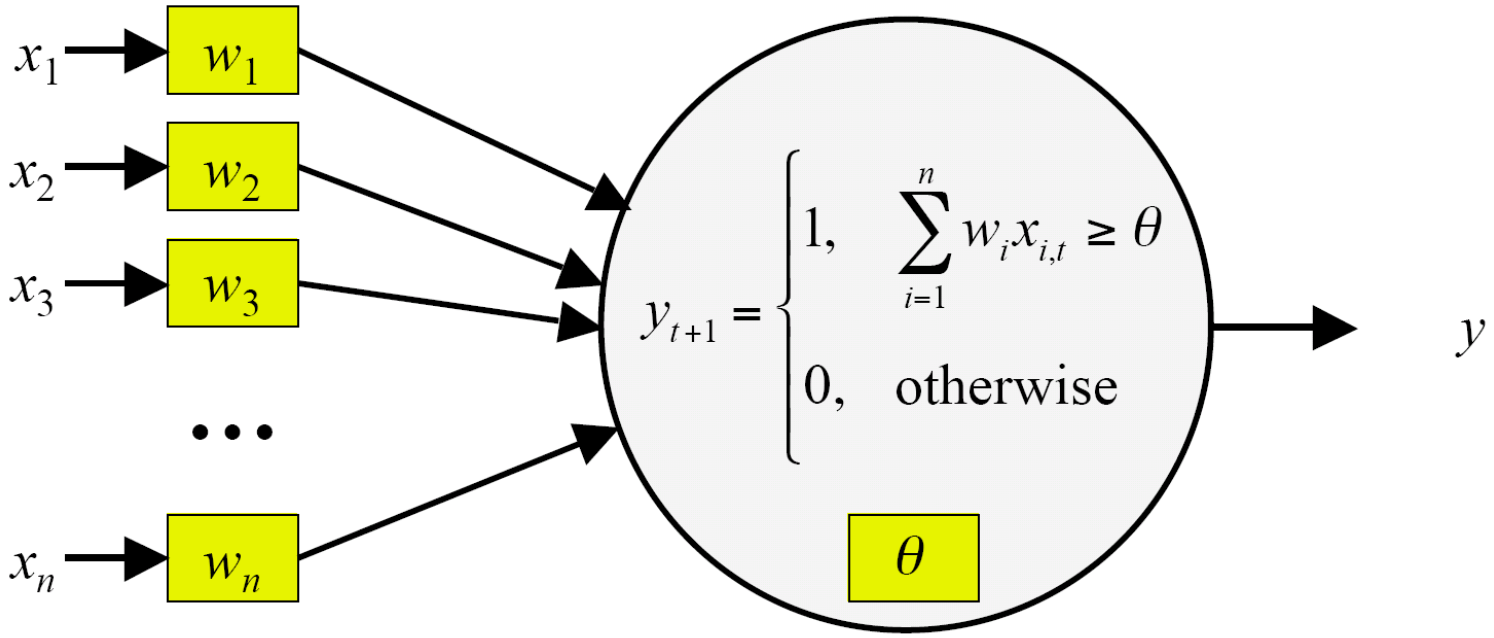
Recurrent Networks



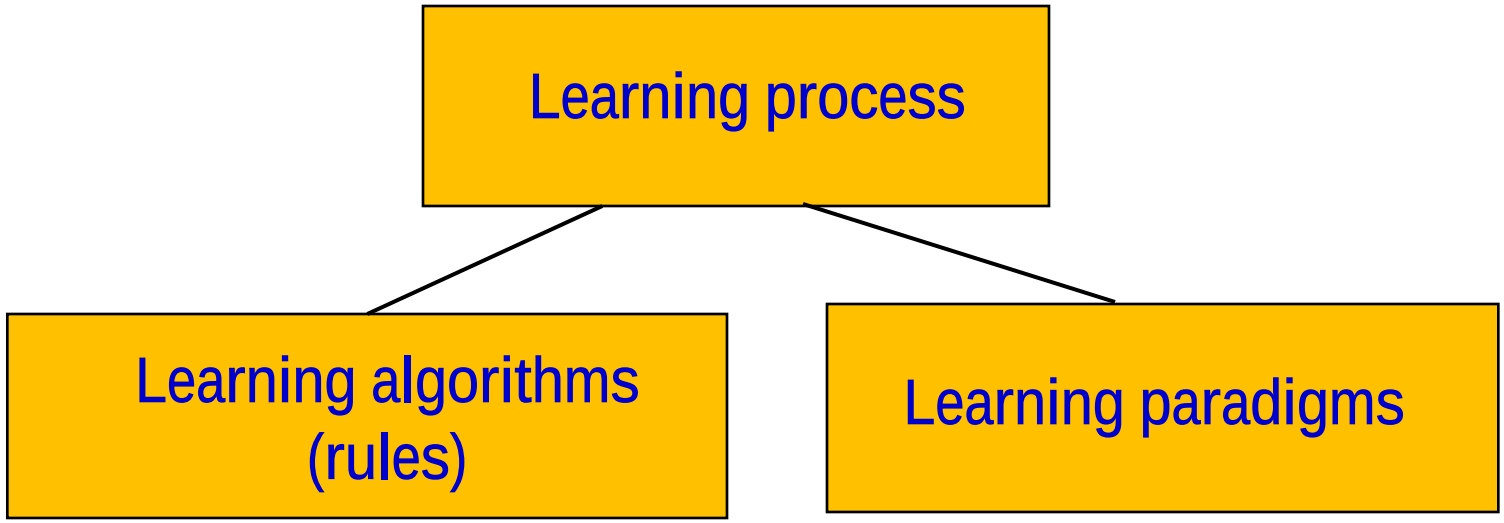
Neurons Unit-delay Operators

Basic model of neurons

- 1943: McCulloch and Pitts develop basic models of neurons
  - The first artificial neuron



A taxonomy of the learning process



- Memory-based
- Error-correct
- Hebbian
- Competitive

- Supervised
- Unsupervised
- Reinforcement

Basic Learning Rules

- Memory-based learning
- Error-corrective learning

Memory-Based Learning -1

- In *memory-based learning*, all (or most) of past experiences are explicitly stored in a large memory.
- This consists of correctly classified input-output examples $\{(\mathbf{x}_1, d_1), (\mathbf{x}_2, d_2), \ldots, (\mathbf{x}_N, d_N)\}$.
- Again, $\mathbf{x}_i$ denotes i-th input vector and d_i the corresponding desired response.
- Without loss of generality, d_i can be restricted to be a scalar.
- Typically, d_i is the number of pattern class.
- Consider now classification of a test vector $\mathbf{x}_{test}$ not seen before.
- This is done by retrieving and analyzing the training data in a local neighborhood of $\mathbf{x}_{test}$.
- All memory-based learning algorithms involve two parts:

Memory-Based Learning -2

1. Criterion used for defining the local neighborhood of the test vector $\mathbf{x}_{test}$.
2. Learning rule applied to the training examples in the local neighborhood of the test vector $\mathbf{x}_{test}$.

- A simple yet effective memory-based learning algorithm is known as the **nearest-neighbor rule**.
- The vector $\mathbf{x}'_N$ belonging to the set of training vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ is the nearest neigbor of $\mathbf{x}_{test}$ if

$$\min_i d(\mathbf{x}_i, \mathbf{x}_{test}) = d(\mathbf{x}'_N, \mathbf{x}_{test})$$

  where $d(\mathbf{x}_i, \mathbf{x}_{test})$ is the Euclidean distance between the vectors $\mathbf{x}_i$ and $\mathbf{x}_{test}$.
- $\mathbf{x}_{test}$ is classified into the same class as its nearest neigbor $\mathbf{x}'_N$.
- Nearest neigbor rule is independent of the underlying distribution.

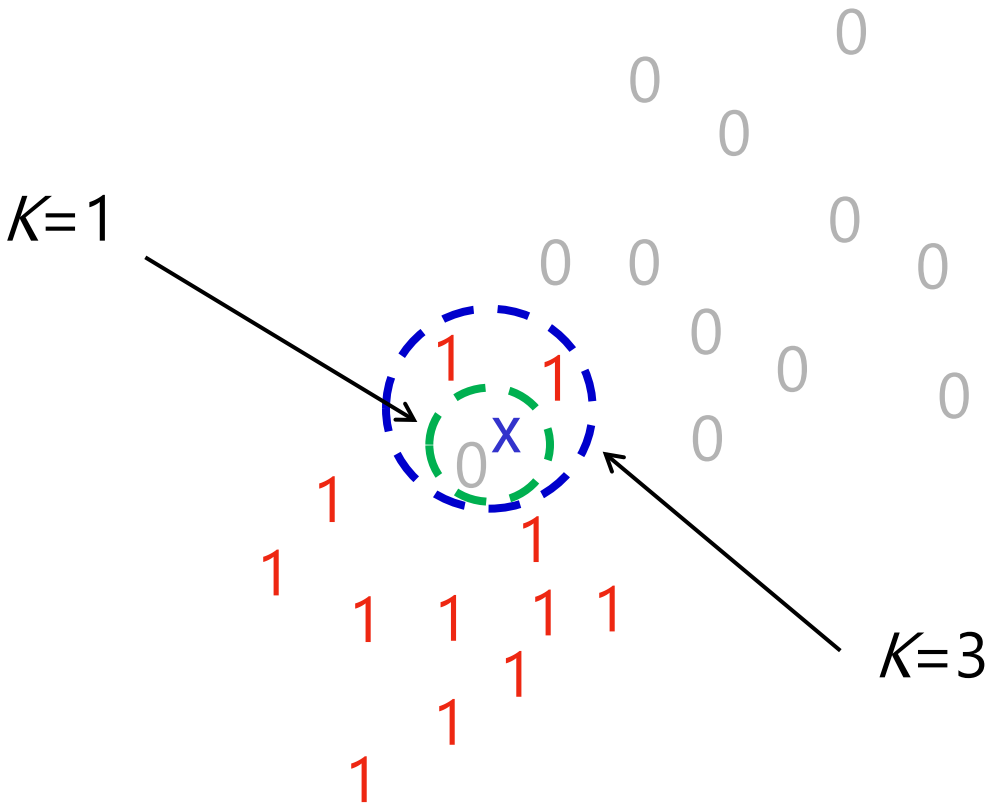
Memory-Based Learning -3

- Assume that:
  - The training samples are independently and identically distributed;
  - The sample size N is infinitely large.
- One can then show that the probability of error in nearest neighbor classification is at most twice the *Bayes probability of error* (Cover and Hart, 1967).
- The Bayes error is the smallest possible (optimal one).
- It is $P_B \leq P_{NN} \leq P_B(2 - \frac{M}{M-1} P_B) \leq 2P_B$
- Thus half the classification information in an infinitely large training set is contained in the nearest neighbor.
- **k-nearest neighbor classifier**:
  $\mathbf{x}_{test}$ is classified to the class which is most frequently represented in its k-nearest neighbors (a majority vote).

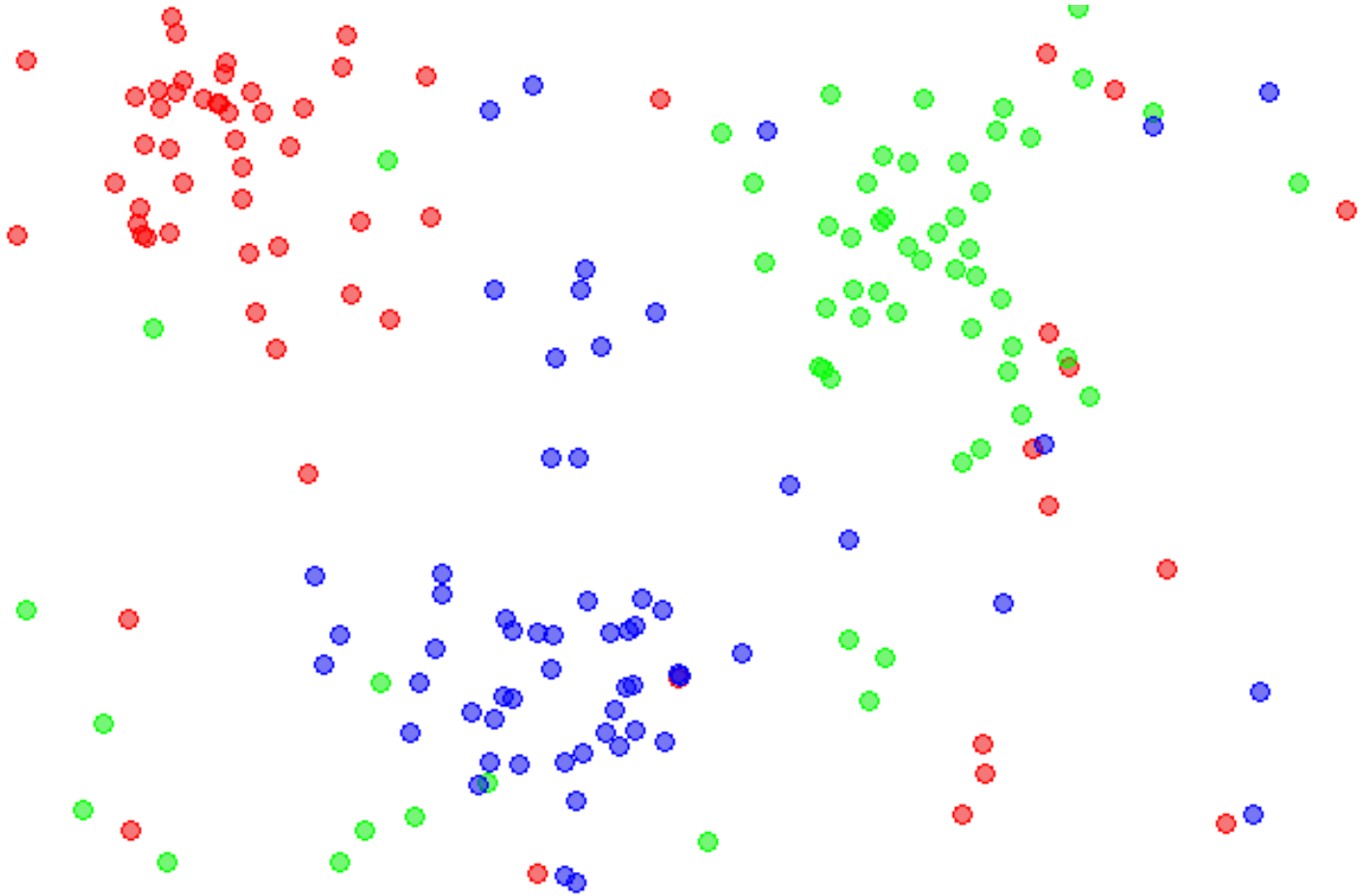
Memory-Based Learning -4

- The k-nearest neigbor classifier averages information, rejecting single outliers.
- *outlier* = exceptional, often erroneous observation.

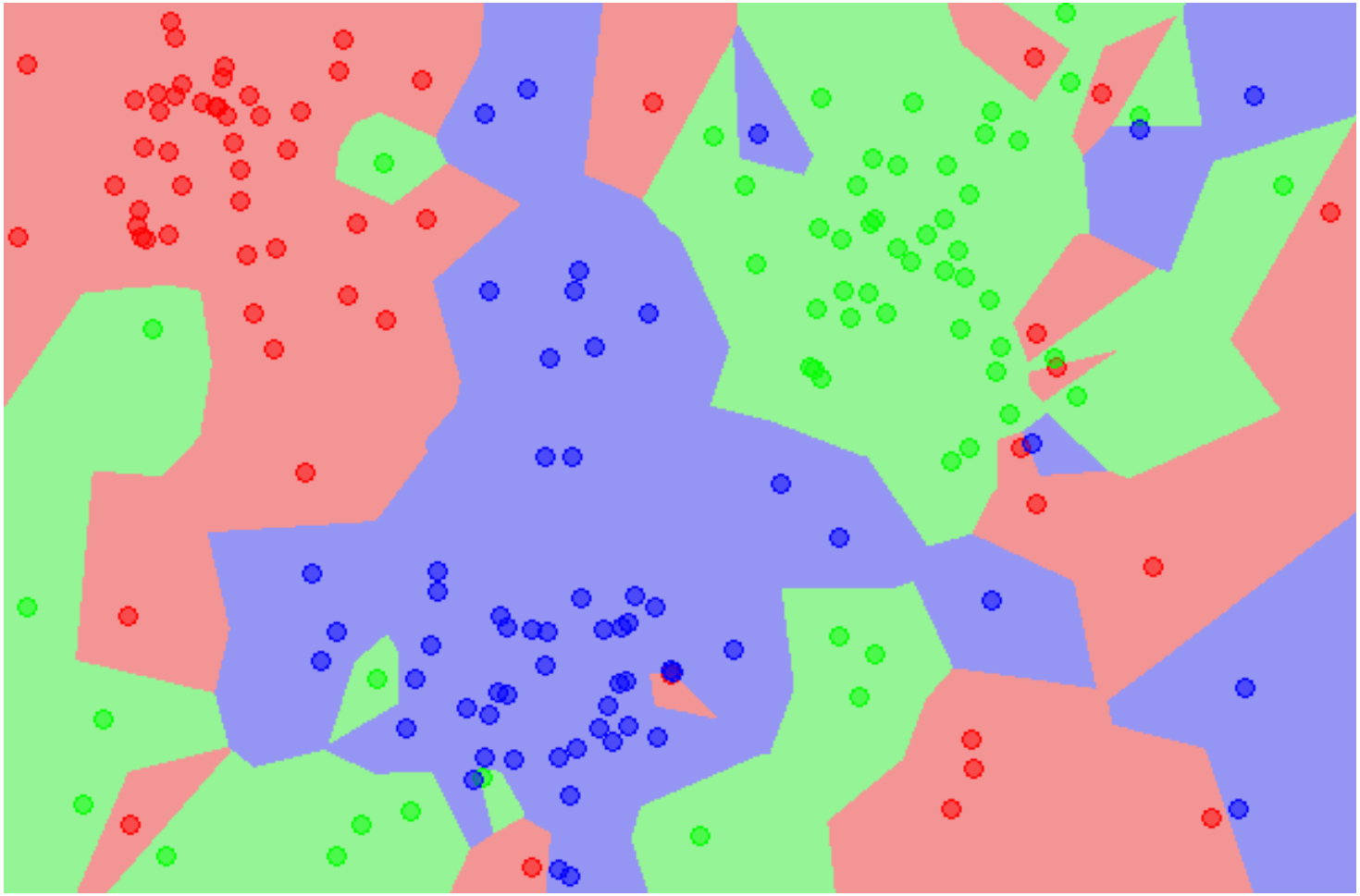
K-Nearest Neighbor Classifier



A three-class problem

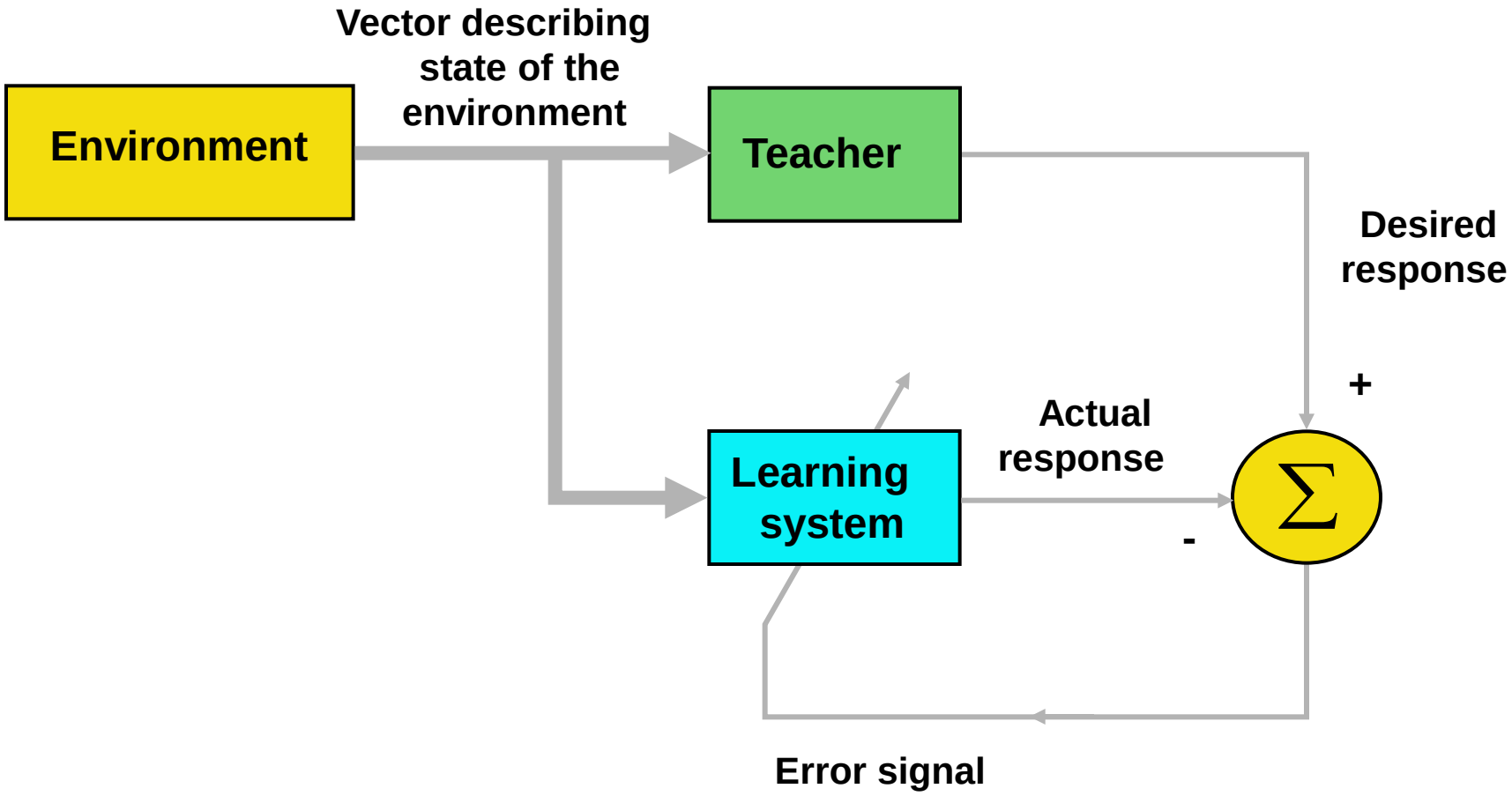


Result of NN Classifier (k=1)



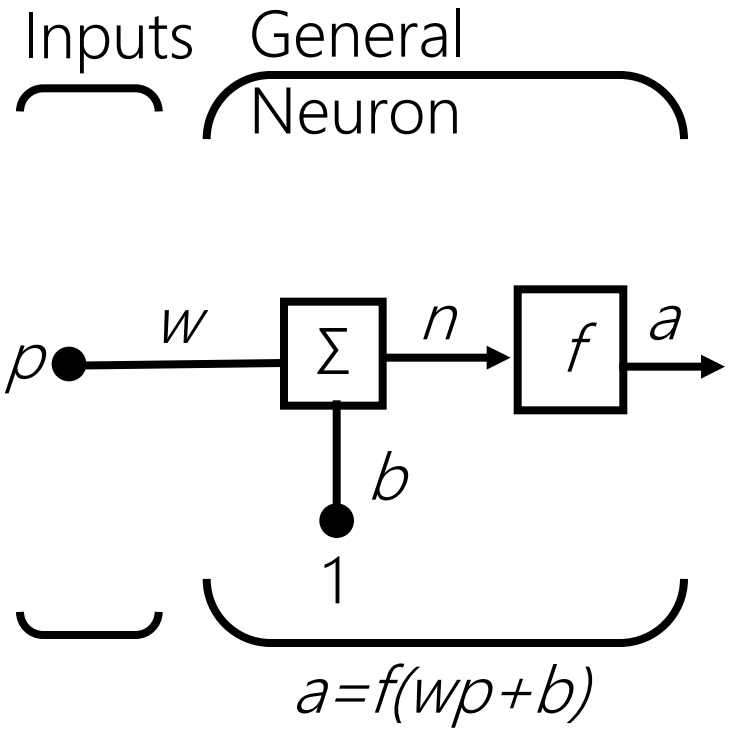
Voronoi Tessellation (棋盘形分布）

Error-Corrective Learning

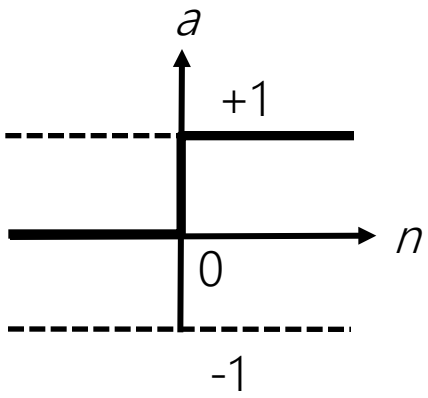


Notation and definition

Single-Input Neuron

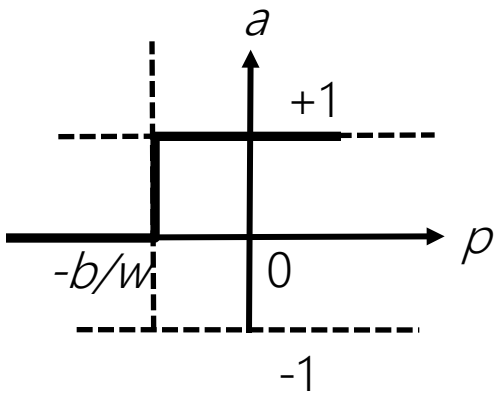


Transfer Functions



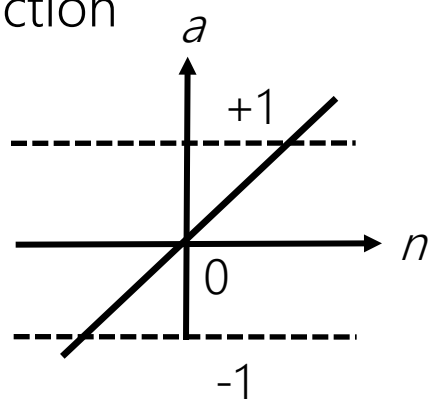
a=hardlim(n)

Hard Limit Transfer Function



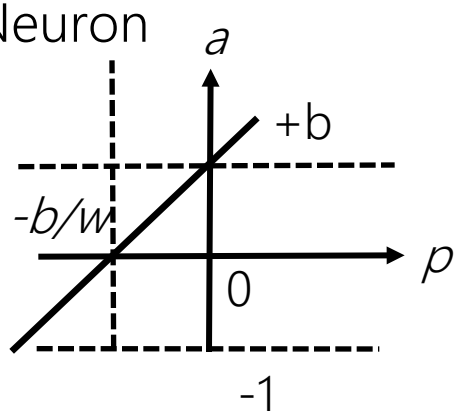
a=hardlim(wp+b)

Single-Input *hardlim* Neuron



a=purelin(n)

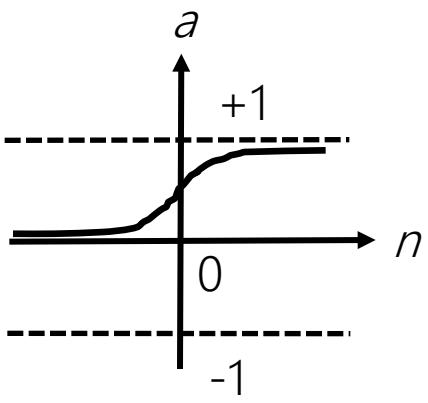
Linear Transfer Function



a=purelin(wp+b)

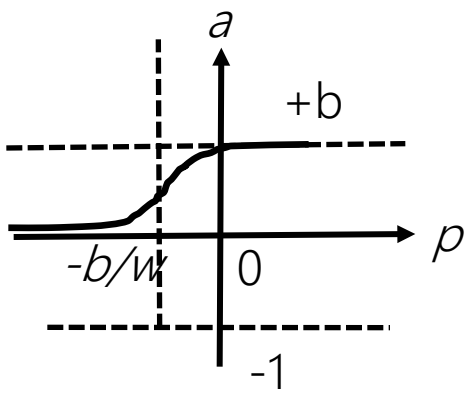
Single-Input *purelin* Neuron

Transfer Functions



a=logsig(n)

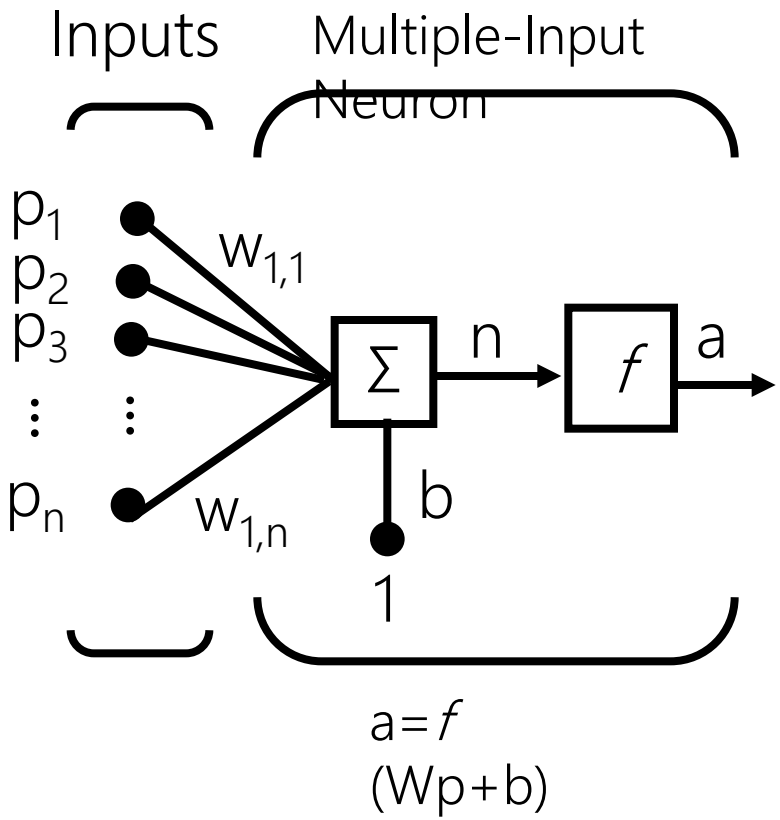
Log-Sigmoid Transfer Function



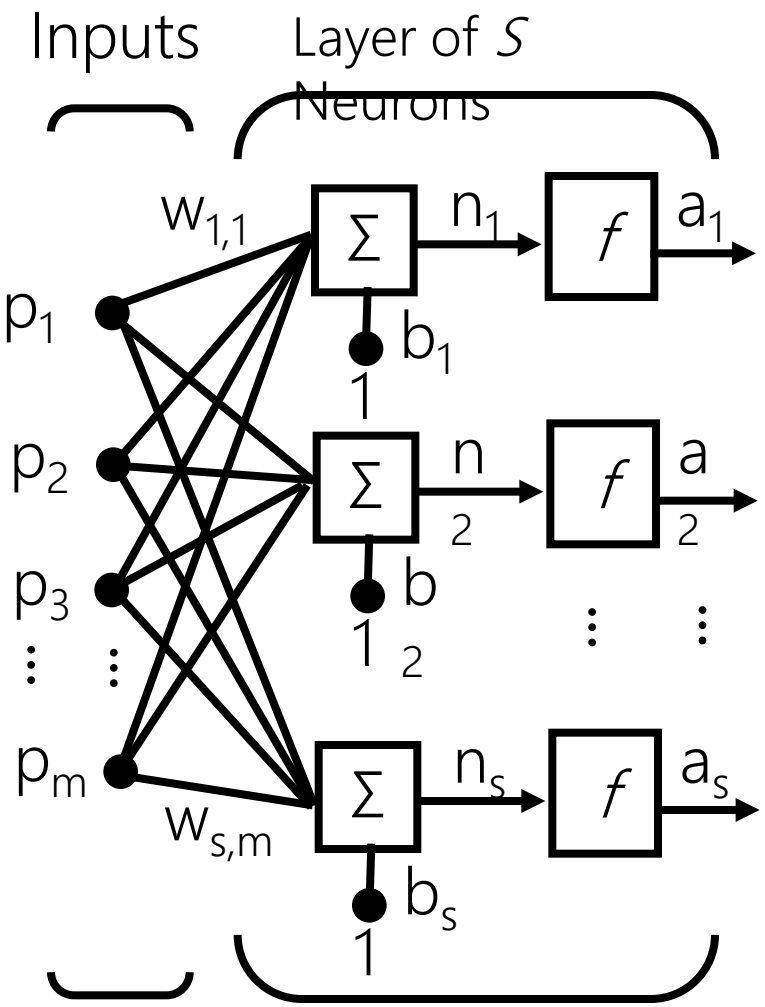
a=logsig(wp+b)

Single-Input *logsig* Neuron

Multiple–Input Neuron

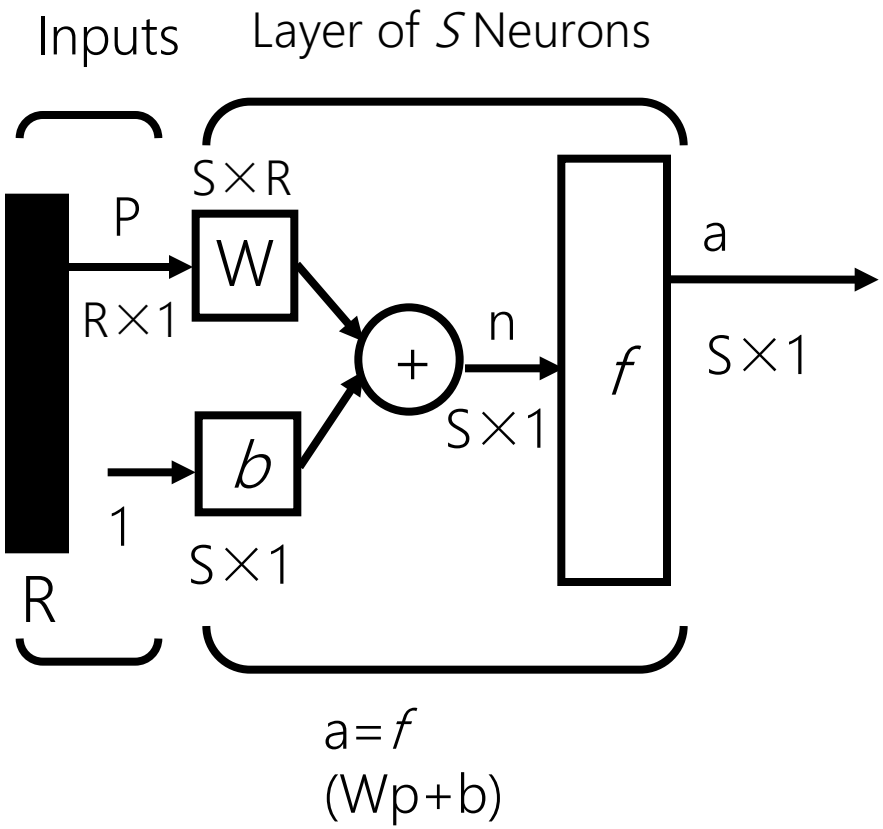


Layer of Neurons



a = *f* (Wp + b)

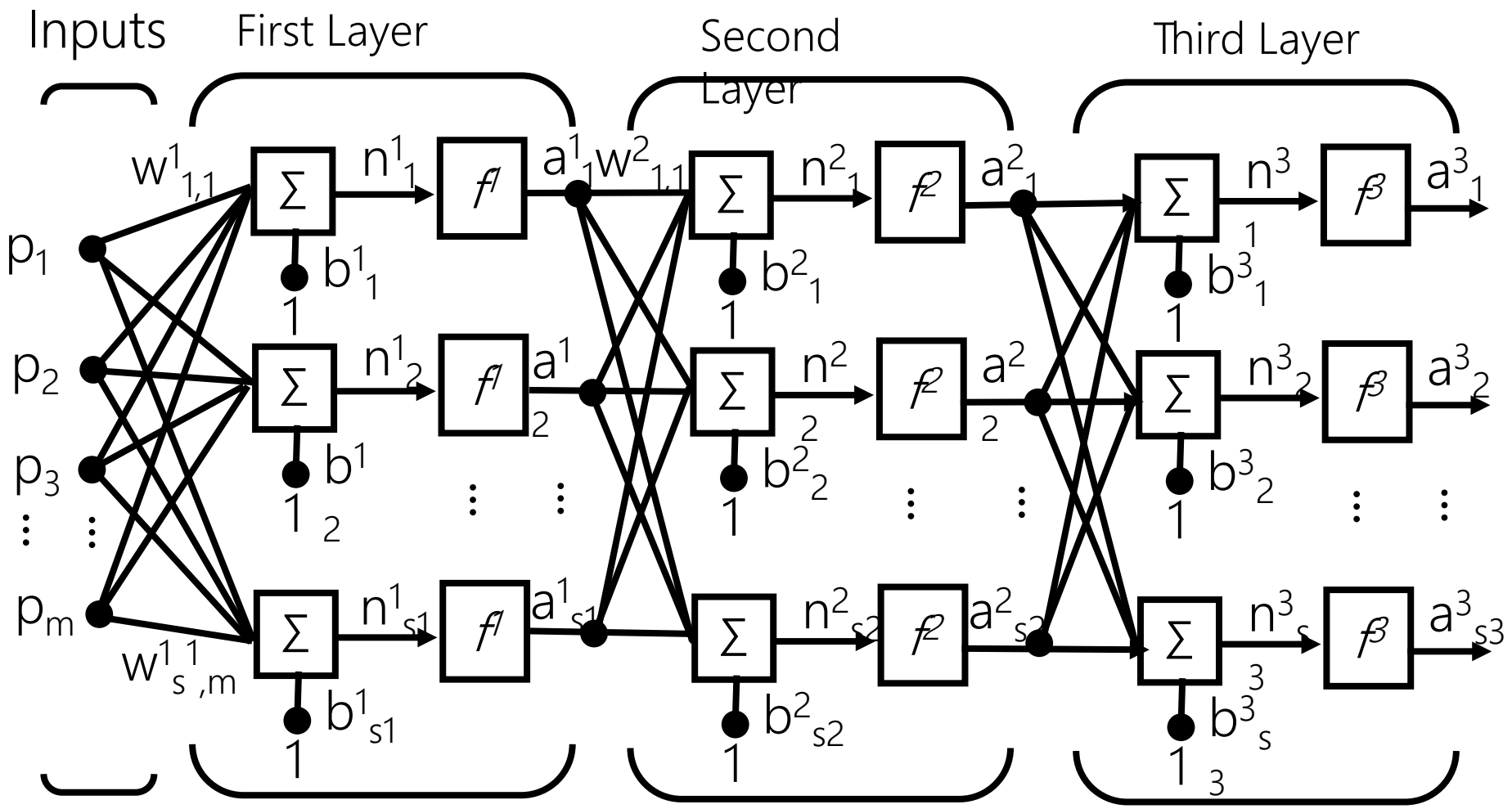
Abbreviated Notation



$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

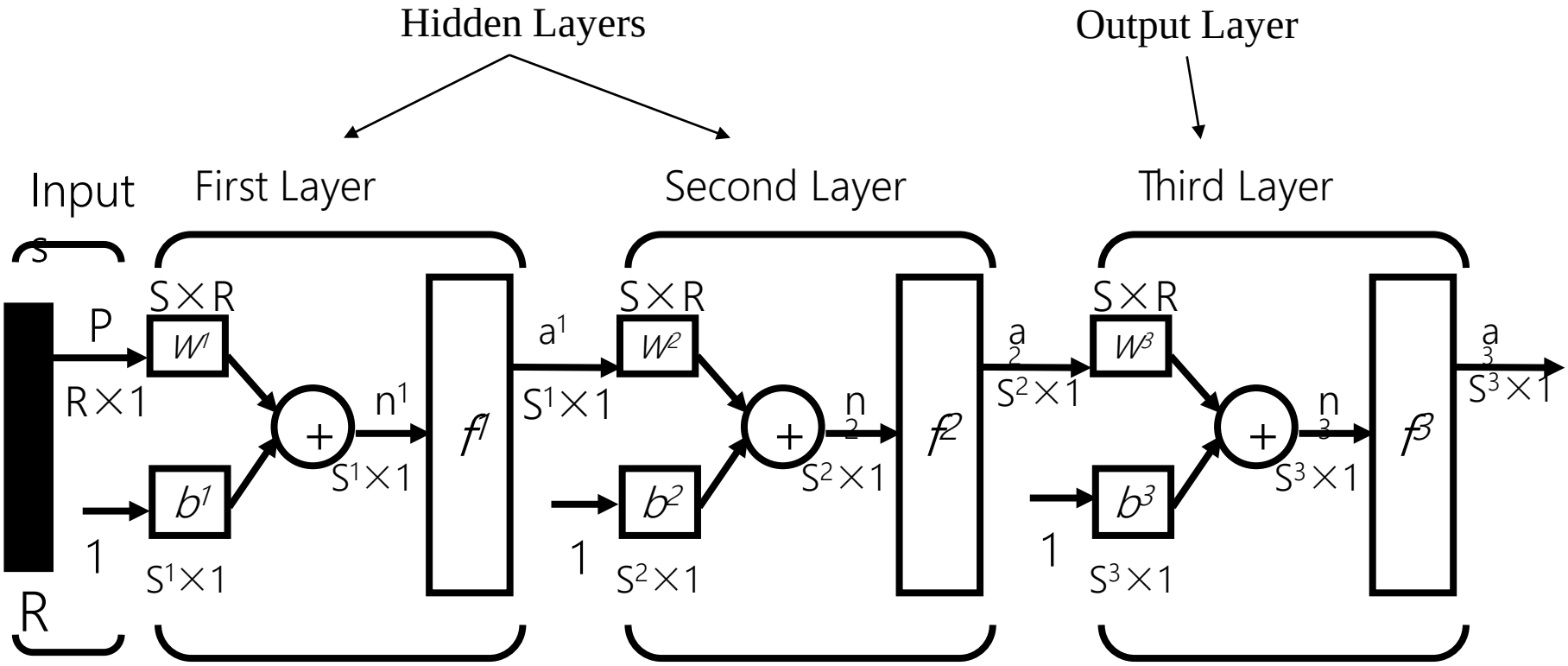
$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_R \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_S \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_S \end{bmatrix}$$

Multilayer Network

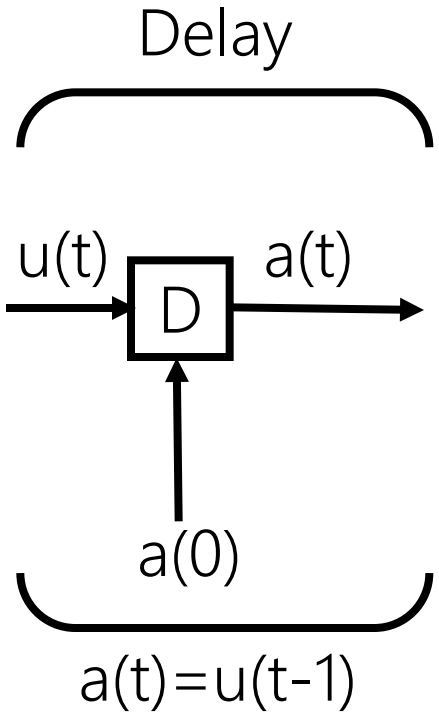


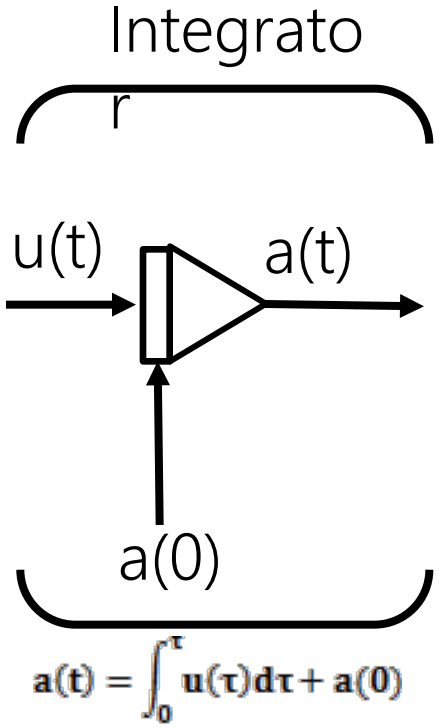
$$a^3 = f^3(W^3 f^2(W^2 f^1(W^1p+b1)+b2)+b3)$$

Abbreviated Notation

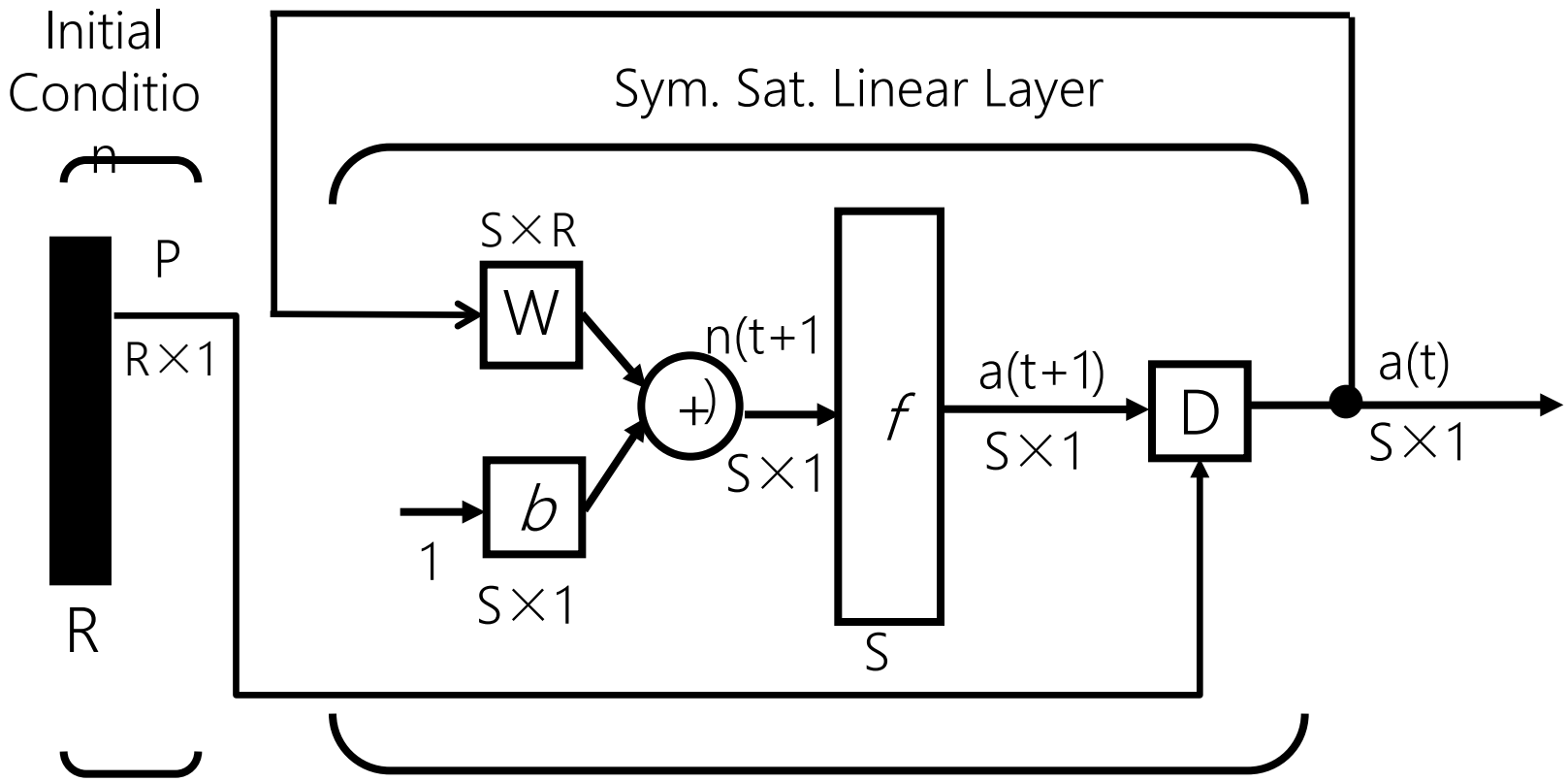


Delays and Integrators





Recurrent Network

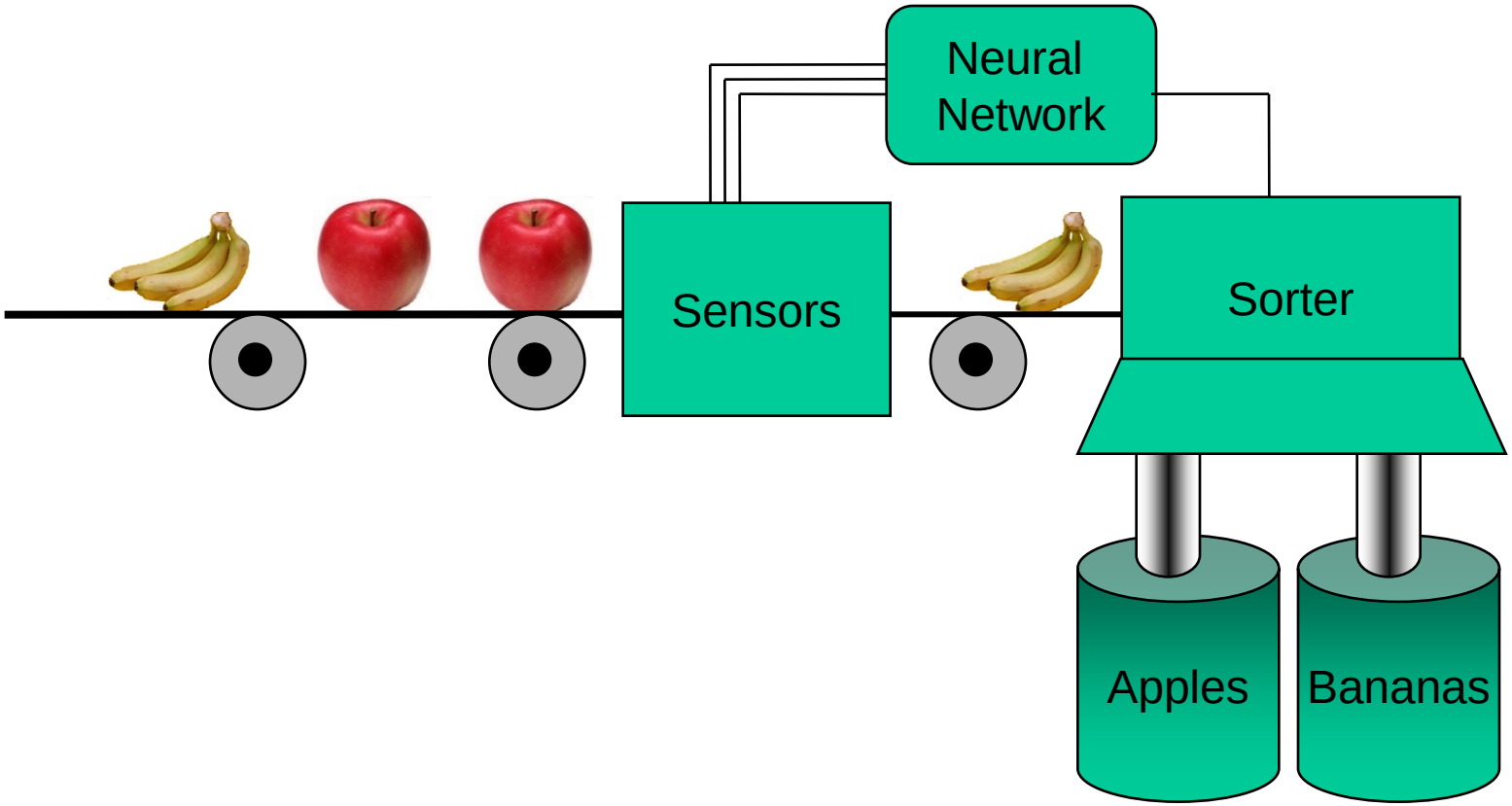


a(0)=p a(t+1)=*satlin*(Wa(t)+b)

$$\mathbf{a}(1) = \mathbf{satlins}(\mathbf{W}\mathbf{a}(0) + \mathbf{b}) = \mathbf{satlins}(\mathbf{W}\mathbf{p} + \mathbf{b})$$

$$\mathbf{a}(2) = \mathbf{satlins}(\mathbf{W}\mathbf{a}(1) + \mathbf{b})$$

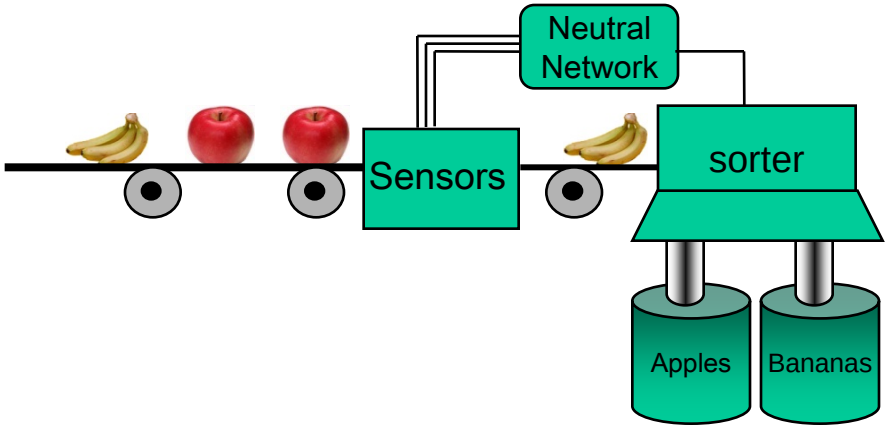
Apple/Banana Sorter



Prototype Vectors

Measurement Vector

$$\mathbf{p} = \begin{bmatrix} \text{shape} \\ \text{texture} \\ \text{weight} \end{bmatrix}$$



Shape: {1 : round ; -1 : eliptical}
Texture: {1 : smooth ; -1 : rough}
Weight: {1 : > 1 lb. ; -1 : < 1 lb.}

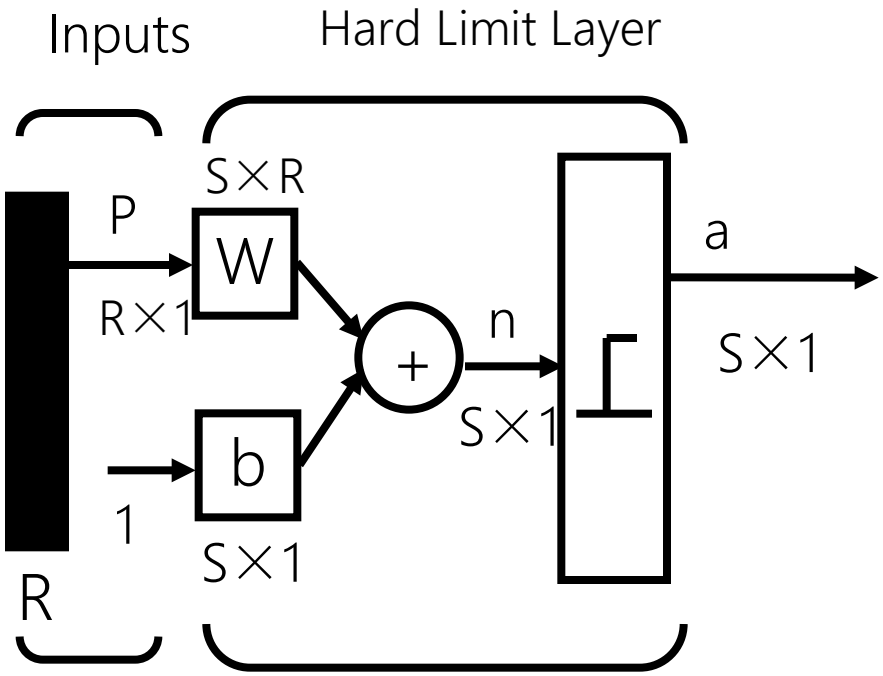
Prototype Banana

$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

Prototype Apple

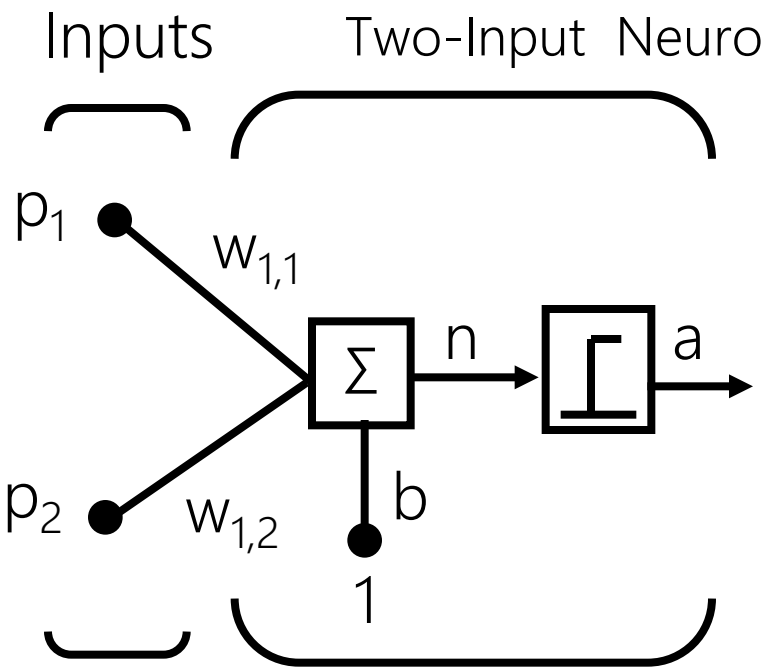
$$\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

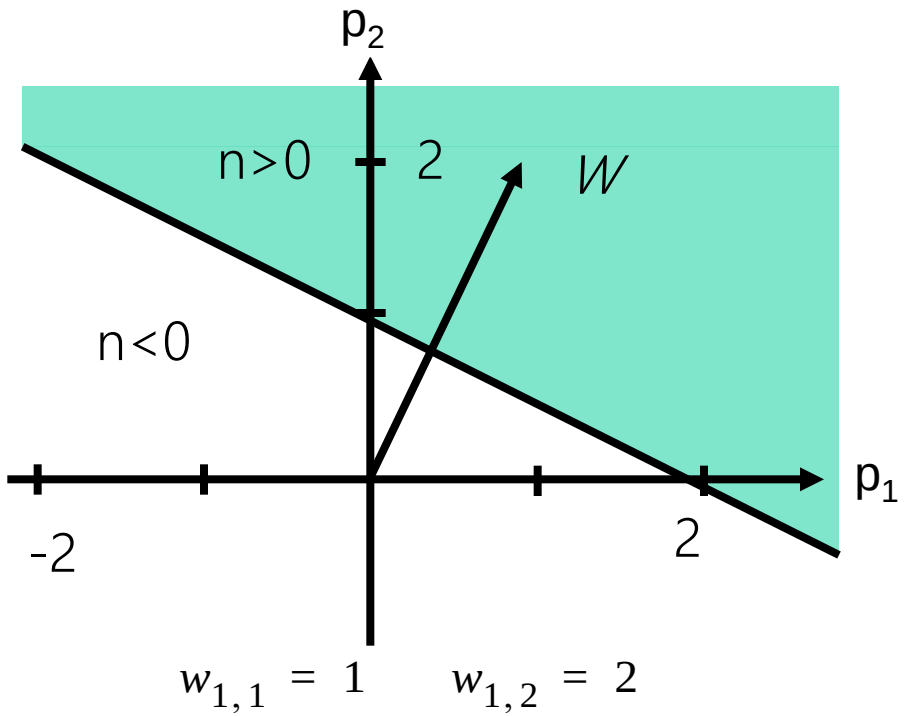
Perceptron



$a = hardlim\,(Wp+b)$

Two-Input Case





a = *hardlim* (Wp+b)

$$a = hardlims(n) = hardlims(\begin{bmatrix}1 & 2\end{bmatrix}\mathbf{p} + (-2))$$

Decision Boundary

$$\mathbf{W}\mathbf{p} + b = 0 \qquad \begin{bmatrix}1 & 2\end{bmatrix}\mathbf{p} + (-2) = 0$$

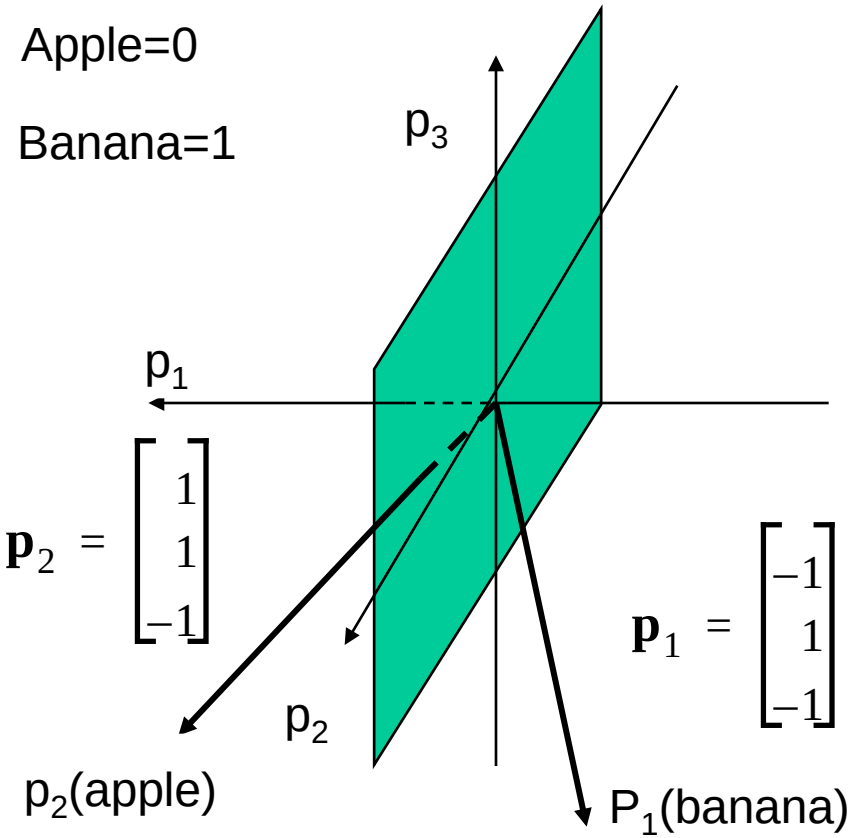
Hyperplane

Apple/Banana Example

$$a = hardlims\left(\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \end{bmatrix}\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + b\right)$$

Apple=0

Banana=1



$$\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

The decision boundary should separate the prototype vectors.

$$p_1 = 0$$

The weight vector should be orthogonal to the decision boundary, and should point in the direction of the vector which should produce an output of 1. The bias determines the position of the boundary

$$\begin{bmatrix} -1 & 0 & 0 \end{bmatrix}\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + 0 = 0$$

Testing the Network

Measurement Vector

$$\mathbf{p} = \begin{bmatrix} \text{shape} \\ \text{texture} \\ \text{weight} \end{bmatrix}$$

Shape: {1 : round ; -1 : eliptical}
Texture: {1 : smooth ; -1 : rough}
Weight: {1 : > 1 lb. ; -1 : < 1 lb.}

Banana:

$$a = hardlims\left(\begin{bmatrix} -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} + 0 \right) = 1(\text{banana})$$

Apple:

$$a = hardlims\left(\begin{bmatrix} -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + 0 \right) = -1(\text{apple})$$

"Rough" Banana:

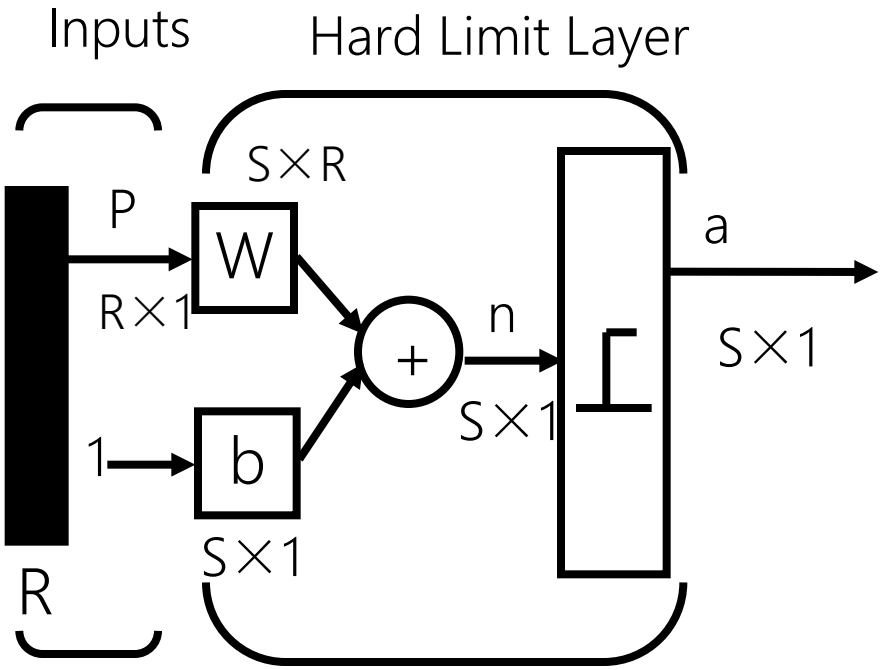
$$a = hardlims\left(\begin{bmatrix} -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + 0 \right) = 1(\text{banana})$$

Summary

- Perceptron
  - Feedforward Network
  - Linear Decision Boundary
  - One Neuron for Each Decision

Perceptron Learning Rule

Perceptron Architecture



$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

$${}_i\mathbf{w} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix}^T \qquad \mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ \vdots \\ {}_S\mathbf{w}^T \end{bmatrix}$$

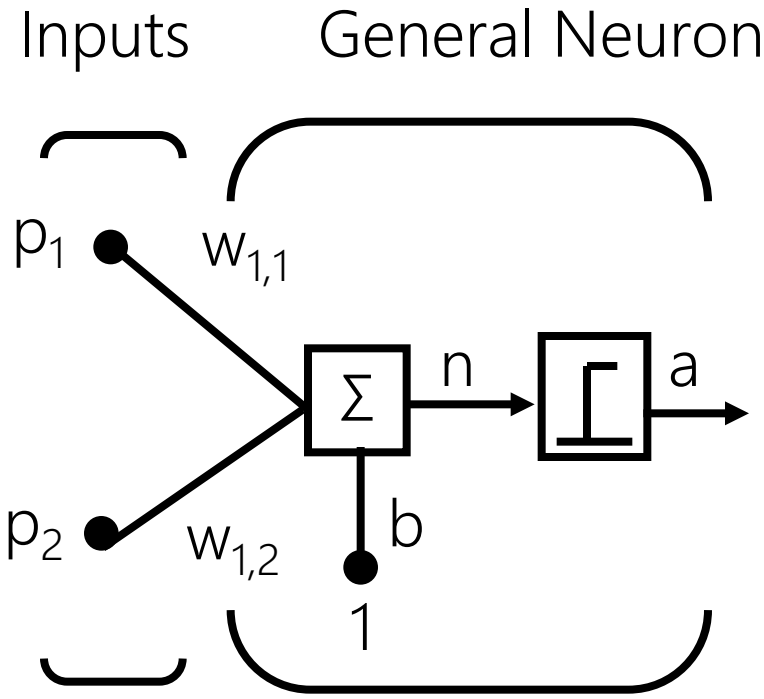
*i*th row of W

a = *hardlim* (Wp+b)

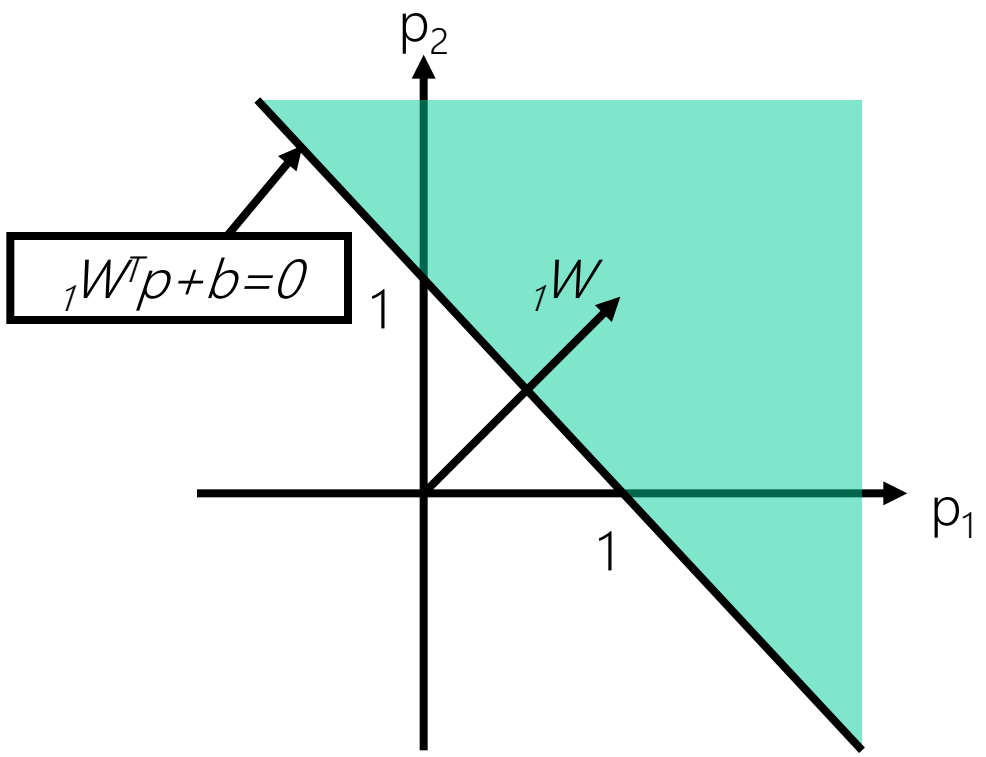
$$a_i = hardlim(n_i) = hardlim({}_i\mathbf{w}^T\mathbf{p} + b_i)$$

Single-Neuron Perceptron

$w_{1,1} = 1 \qquad w_{1,2} = 1 \qquad b = -1$



a = *hardlim* (Wp+b)

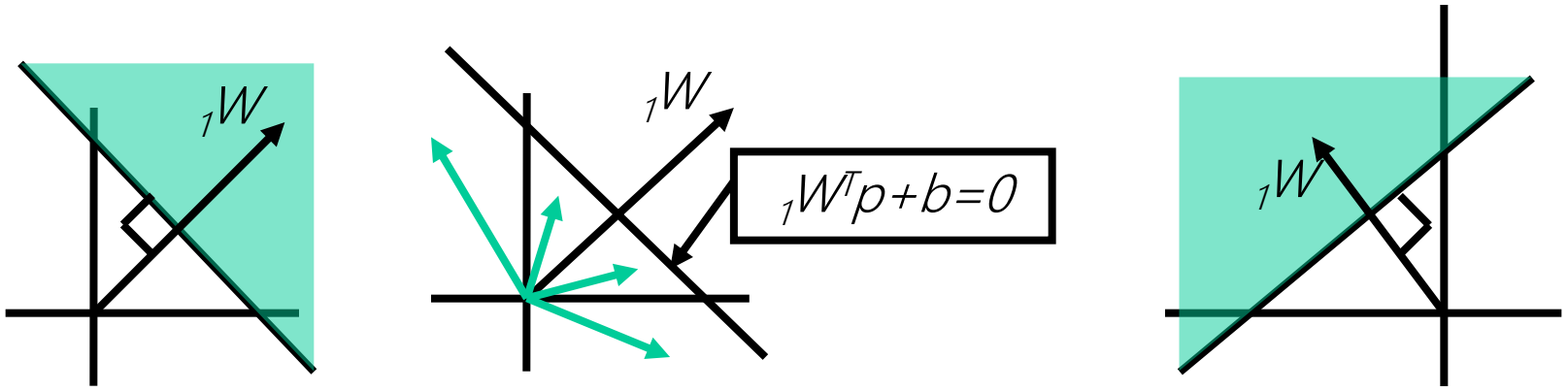


$$a = hardlim({}_1\mathbf{w}^{\mathrm{T}}\mathbf{p} + b) = hardlim(w_{1,1}p_1 + w_{1,2}p_2 + b)$$

Decision Boundary

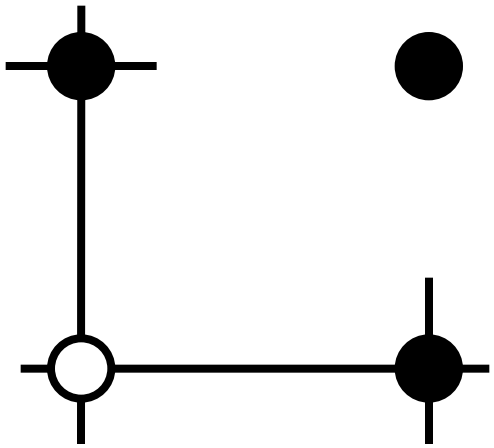
$$ {}_1\mathbf{w}^{\mathrm{T}}\mathbf{p} + b = 0 \qquad {}_1\mathbf{w}^{\mathrm{T}}\mathbf{p} = -b $$

- All points on the decision boundary have the same inner product with the weight vector.
- Therefore they have the same projection onto the weight vector, and they must lie on a line orthogonal to the weight vector

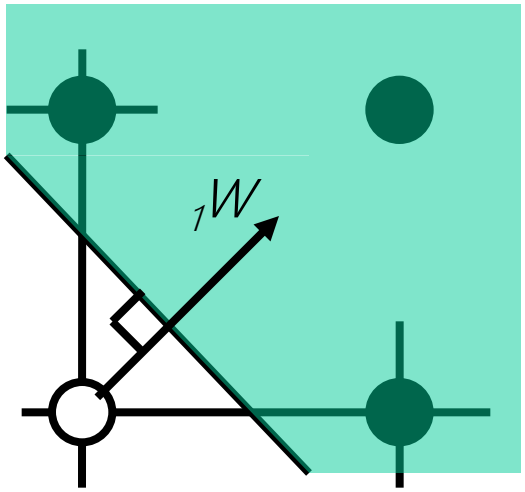


Example - OR

$$\left\{\mathbf{p}_1 = \begin{bmatrix}0\\0\end{bmatrix}, t_1 = 0\right\} \quad \left\{\mathbf{p}_2 = \begin{bmatrix}0\\1\end{bmatrix}, t_2 = 1\right\} \quad \left\{\mathbf{p}_3 = \begin{bmatrix}1\\0\end{bmatrix}, t_3 = 1\right\} \quad \left\{\mathbf{p}_4 = \begin{bmatrix}1\\1\end{bmatrix}, t_4 = 1\right\}$$



OR Solution



Weight vector should be orthogonal to the decision boundary.

$$_1\mathbf{w} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

Pick a point on the decision boundary to find the bias.

$$_1\mathbf{w}^T\mathbf{p} + b = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}\begin{bmatrix} 0 \\ 0.5 \end{bmatrix} + b = 0.25 + b = 0 \quad \Rightarrow \quad b = -0.25$$

Multiple-Neuron Perceptron

Each neuron will have its own decision boundary.

$$_i\mathbf{w}^T\mathbf{p} + b_i = 0$$

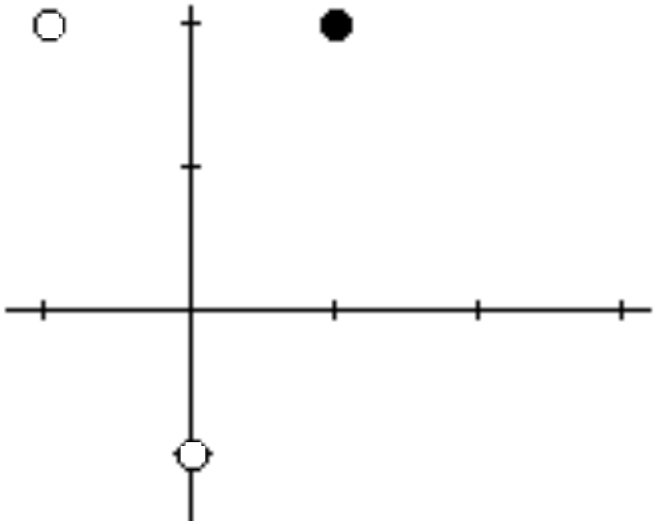
A single neuron can classify input vectors into two categories.

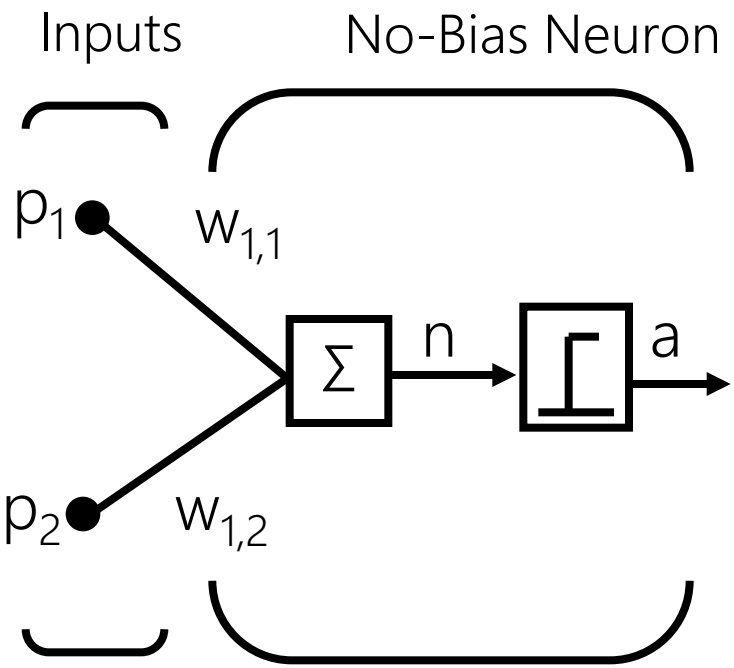
A multi-neuron perceptron can classify input vectors into 2^S categories.

Learning Rule Test Problem

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\2\end{bmatrix}, t_1 = 1\right\} \quad \left\{\mathbf{p}_2 = \begin{bmatrix}-1\\2\end{bmatrix}, t_2 = 0\right\} \quad \left\{\mathbf{p}_3 = \begin{bmatrix}0\\-1\end{bmatrix}, t_3 = 0\right\}$$



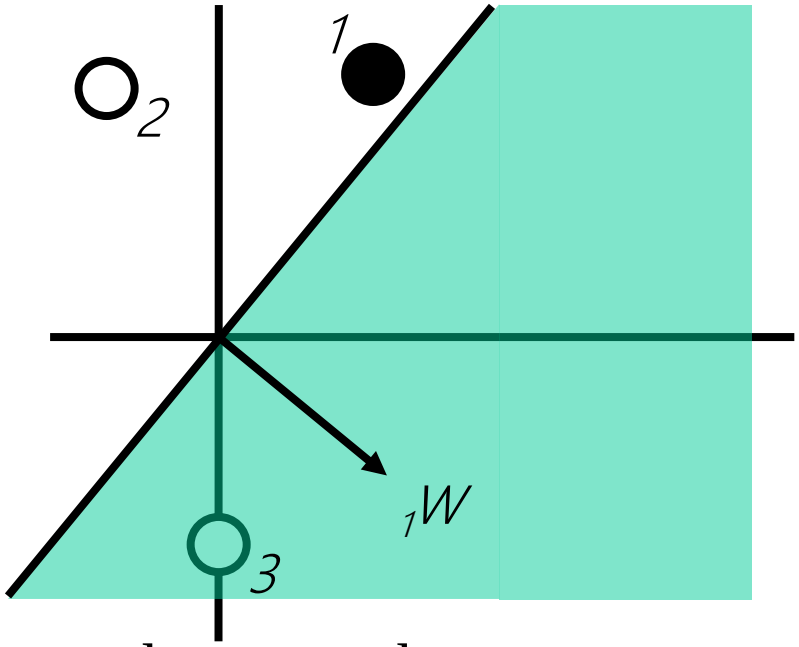


a = *hardlim* (Wp)

Starting Point

Random initial weight:

$$ {}_1\mathbf{w} = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} $$



Present $\mathbf{p}_1$ to the network:

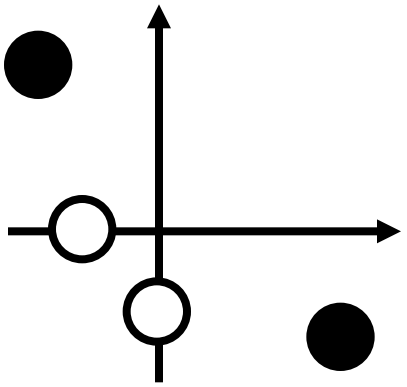
$$ a = hardlim({}_1\mathbf{w}^T\mathbf{p}_1) = hardlim\left(\begin{bmatrix} 1.0 & -0.8 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right) $$

$$ a = hardlim(-0.6) = 0 $$

Incorrect Classification.

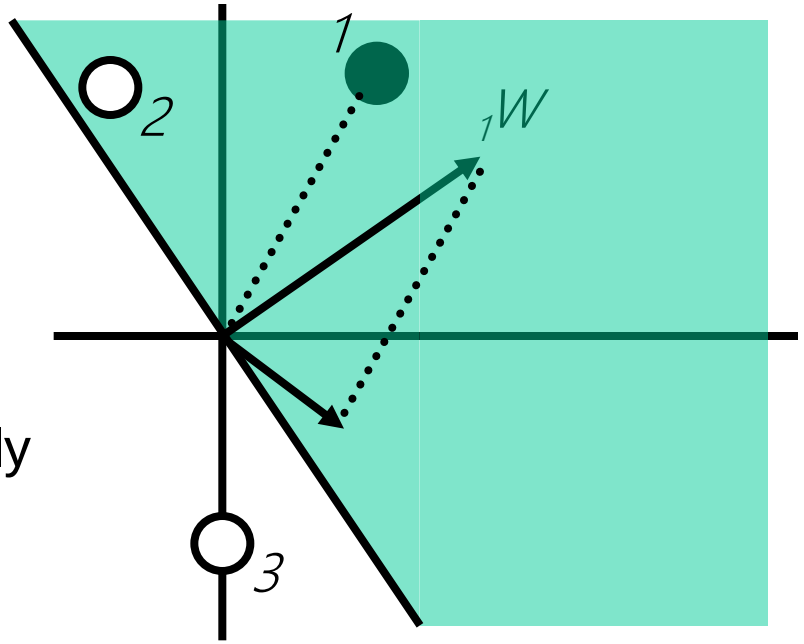
Tentative Learning Rule

- Set ${}_1\mathbf{w}$ to $\mathbf{p}_1$ ×
  - Not stable
- Add $\mathbf{p}_1$ to ${}_1\mathbf{w}$ √



Tentative Rule: If $t = 1$ and $a = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix}$$



Repeated presentations of p_1 would cause the direction of ${}_1W$ to asymptotically approach the direction of p_1

Second Input Vector

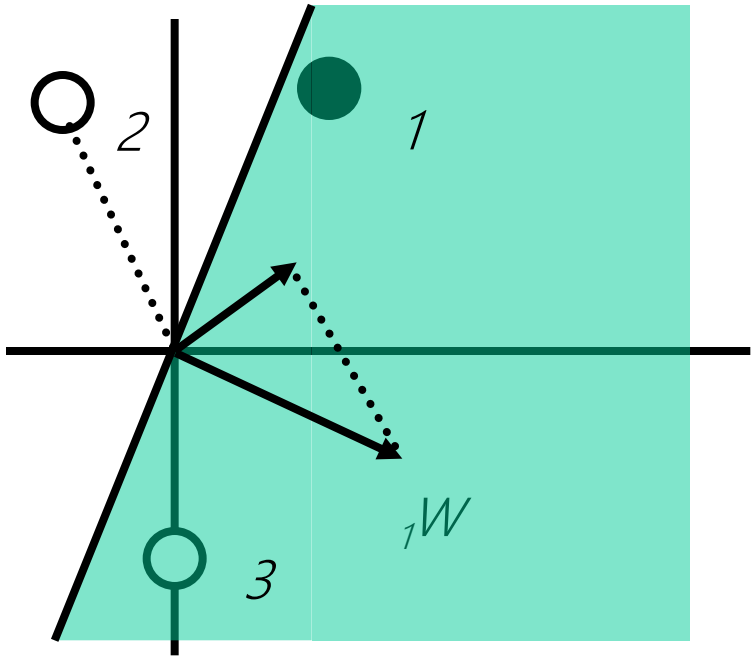
$$a = hardlim({}_1\mathbf{w}^{T}\mathbf{p}_2) = hardlim\left(\begin{bmatrix}2.0 & 1.2\end{bmatrix}\begin{bmatrix}-1\\2\end{bmatrix}\right)$$

$$a = hardlim(0.4) = 1 \quad \text{(Incorrect Classification)}$$

Modification to Rule: If $t = 0$ and $a = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_2 = \begin{bmatrix}2.0\\1.2\end{bmatrix} - \begin{bmatrix}-1\\2\end{bmatrix} = \begin{bmatrix}3.0\\-0.8\end{bmatrix}$$

Since we would like to move the weight vector ${}_1W$ away from the input, we can simply change the addition to subtraction

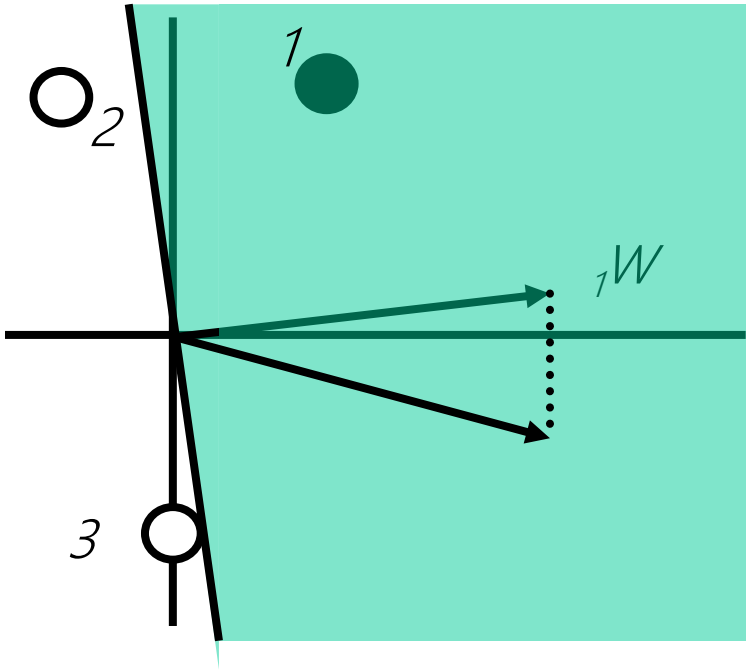


Third Input Vector

$$a = hardlim({}_1\mathbf{w}^T \mathbf{p}_3) = hardlim\left(\begin{bmatrix}3.0 & -0.8\end{bmatrix}\begin{bmatrix}0\\-1\end{bmatrix}\right)$$

$$a = hardlim(0.8) = 1 \qquad \text{(Incorrect Classification)}$$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_3 = \begin{bmatrix}3.0\\-0.8\end{bmatrix} - \begin{bmatrix}0\\-1\end{bmatrix} = \begin{bmatrix}3.0\\0.2\end{bmatrix}$$



Patterns are now correctly classified.

If $t = a$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$.

Unified Learning Rule

If $t = 1$ and $a = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$

If $t = 0$ and $a = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

If $t = a$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$

$$e = t - a$$

If $e = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$

If $e = -1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

If $e = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e\mathbf{p} = {}_1\mathbf{w}^{old} + (t-a)\mathbf{p}$$

$$b^{new} = b^{old} + e$$

A bias is a weight with an input of 1.

Multiple-Neuron Perceptrons

To update the *i*th row of the weight matrix:

$$ {}_i\mathbf{w}^{new} = {}_i\mathbf{w}^{old} + e_i\mathbf{p} $$

$$ b_i^{new} = b_i^{old} + e_i $$

Matrix form:

$$ \mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^T $$

$$ \mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e} $$

Apple/Banana Example

Training Set

$$\left\{\mathbf{p}_1 = \begin{bmatrix}-1\\1\\-1\end{bmatrix}, t_1 = \begin{bmatrix}1\end{bmatrix}\right\} \qquad \left\{\mathbf{p}_2 = \begin{bmatrix}1\\1\\-1\end{bmatrix}, t_2 = \begin{bmatrix}0\end{bmatrix}\right\}$$

Initial Weights

$$\mathbf{W} = \begin{bmatrix}0.5 & -1 & -0.5\end{bmatrix} \qquad b = 0.5$$

First Iteration

$$a = hardlim(\mathbf{W}\mathbf{p}_1 + b) = hardlim\left(\begin{bmatrix}0.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}-1\\1\\-1\end{bmatrix} + 0.5\right)$$

$$a = hardlim(-0.5) = 0 \qquad e = t_1 - a = 1 - 0 = 1$$

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix}0.5 & -1 & -0.5\end{bmatrix} + (1)\begin{bmatrix}-1 & 1 & -1\end{bmatrix} = \begin{bmatrix}-0.5 & 0 & -1.5\end{bmatrix}$$

$$b^{new} = b^{old} + e = 0.5 + (1) = 1.5$$

Second Iteration

$$a = hardlim\,(\mathbf{W}\mathbf{p}_2 + b) = hardlim\,(\begin{bmatrix}-0.5 & 0 & -1.5\end{bmatrix}\begin{bmatrix}1\\1\\-1\end{bmatrix} + (1.5))$$

$$a = hardlim\,(2.5) = 1$$

$$e = t_2 - a = 0 - 1 = -1$$

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix}-0.5 & 0 & -1.5\end{bmatrix} + (-1)\begin{bmatrix}1 & 1 & -1\end{bmatrix} = \begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}$$

$$b^{new} = b^{old} + e = 1.5 + (-1) = 0.5$$

Test

$$a = hardlim(\mathbf{W}\mathbf{p}_1 + b) = hardlim\left(\begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}-1\\1\\-1\end{bmatrix} + 0.5\right)$$

$$a = hardlim(1.5) = 1 = t_1$$

$$a = hardlim(\mathbf{W}\mathbf{p}_2 + b) = hardlim\left(\begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}1\\1\\-1\end{bmatrix} + 0.5\right)$$

$$a = hardlim(-1.5) = 0 = t_2$$

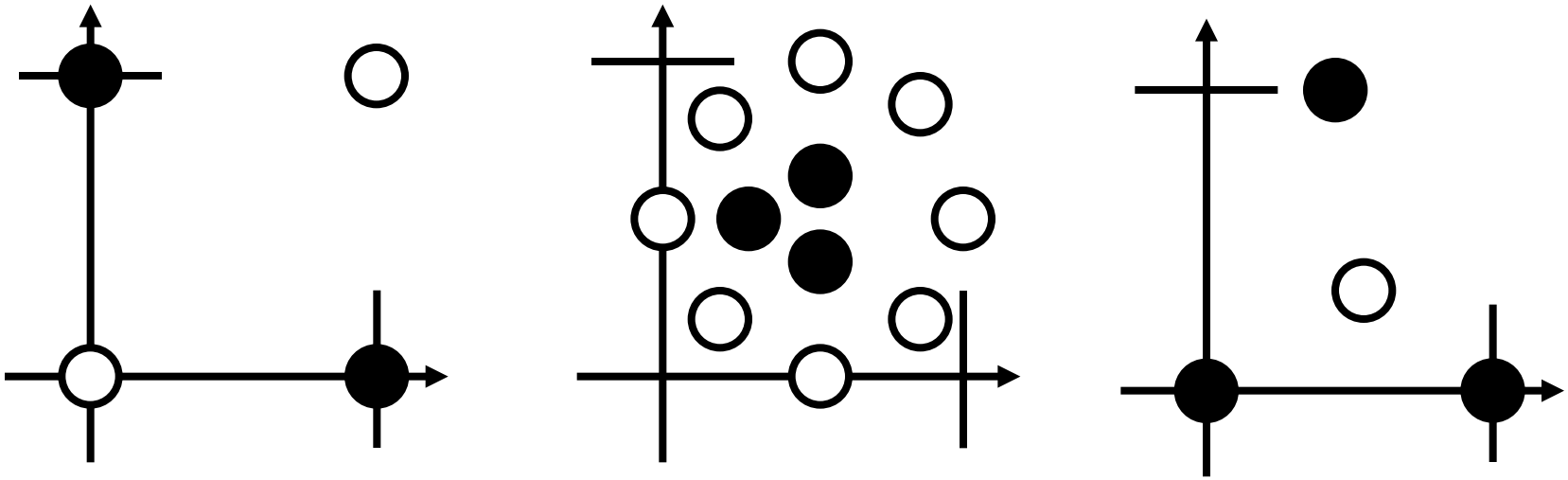
Proof of Convergence

Perceptron Limitations

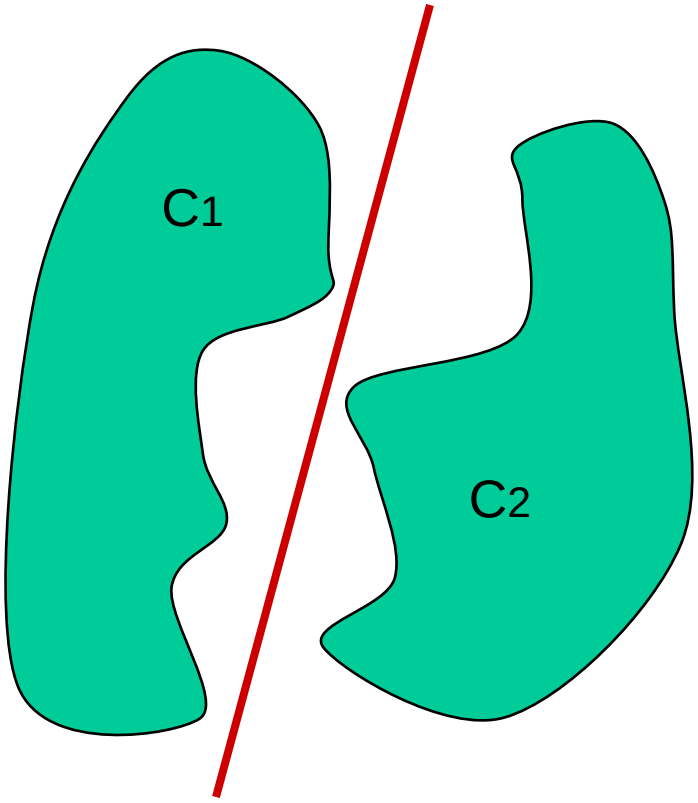
Linear Decision Boundary

$${}_1\mathbf{w}^T\mathbf{p} + b = 0$$

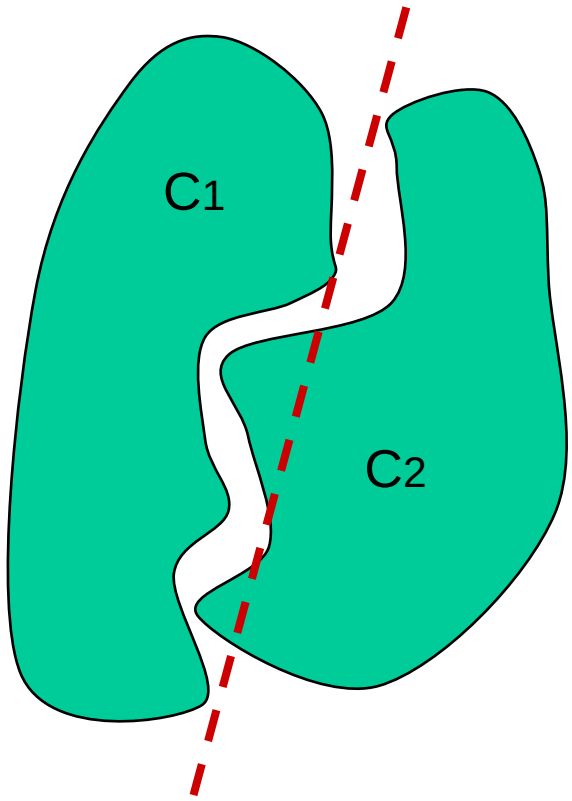
Non-linearly Separable Problems



Linearly vs Non-linearly Separable Problems



Linearly Separable

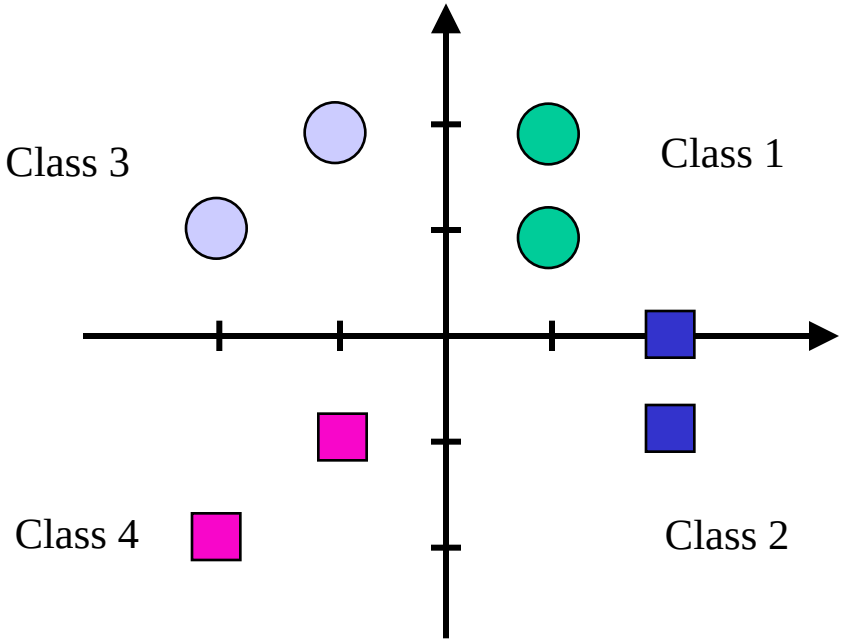


Non-linearly Separable

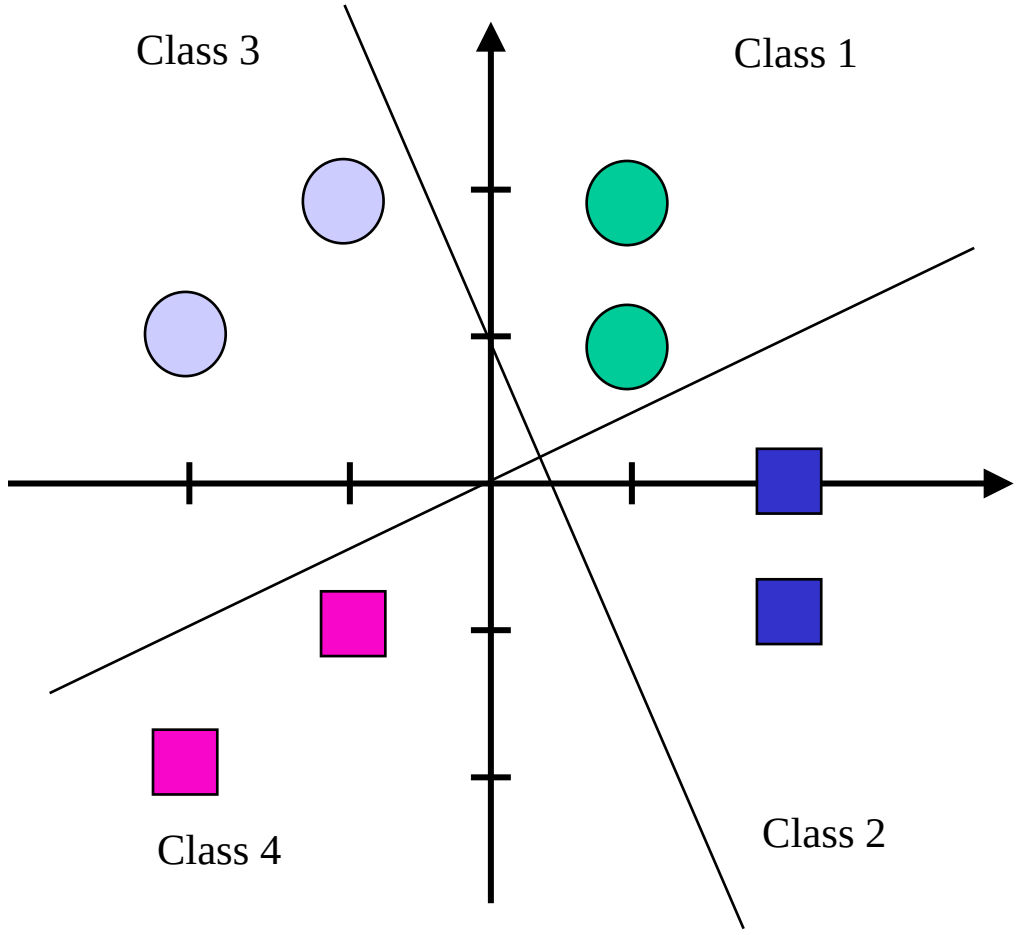
Four-Class Problem: Training Set

$$\text{Class } 1: \left\{ p_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \; p_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\}, \qquad \text{Class } 2: \left\{ p_3 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \; p_4 = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \right\},$$

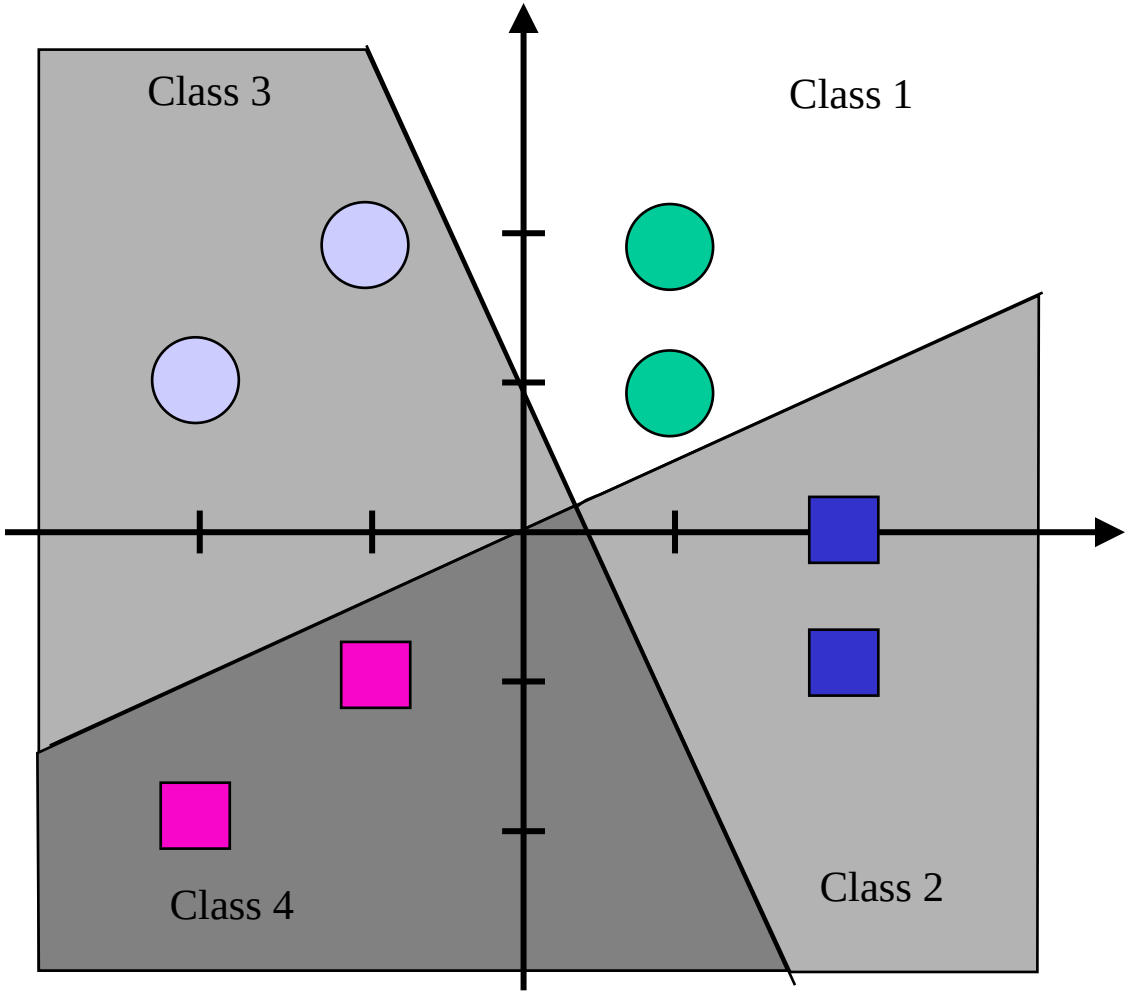
$$\text{Class } 3: \left\{ p_5 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, \; p_6 = \begin{bmatrix} -2 \\ 1 \end{bmatrix} \right\}, \quad \text{Class } 4: \left\{ p_7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \; p_8 = \begin{bmatrix} -2 \\ -2 \end{bmatrix} \right\}.$$



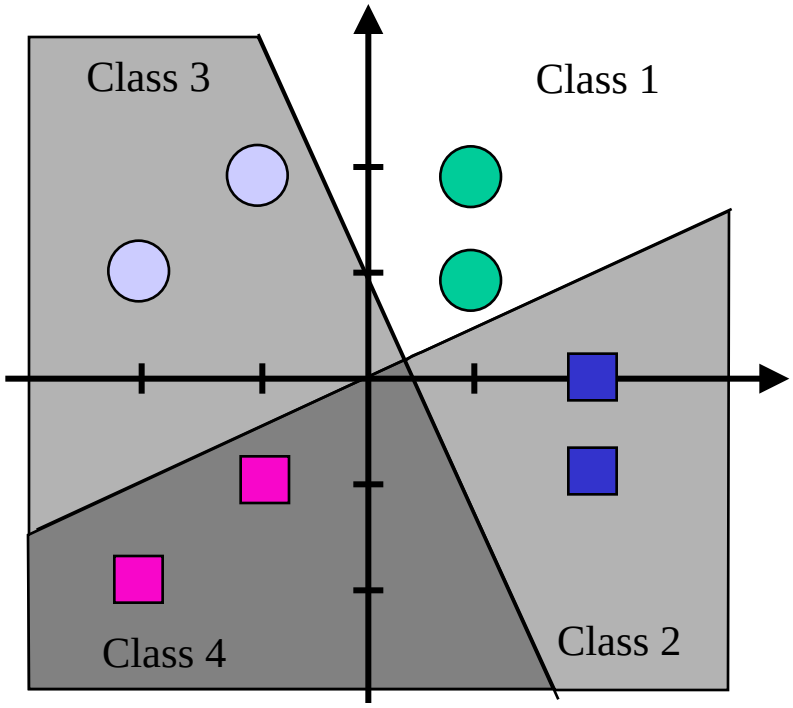
Tentative Decision Boundaries



Decision Regions



Target Representation



$$\text{Class } 1: \left\{ t_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix},\ t_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\}, \quad \text{Class } 2: \left\{ t_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},\ t_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\},$$

$$\text{Class } 3: \left\{ t_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ t_6 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\}, \quad \text{Class } 4: \left\{ t_7 = \begin{bmatrix} 1 \\ 1 \end{bmatrix},\ t_8 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}.$$

Solution

$$ {}_1W = \begin{bmatrix} -3 \\ -1 \end{bmatrix} \text{ and } {}_2W = \begin{bmatrix} 1 \\ -2 \end{bmatrix} $$

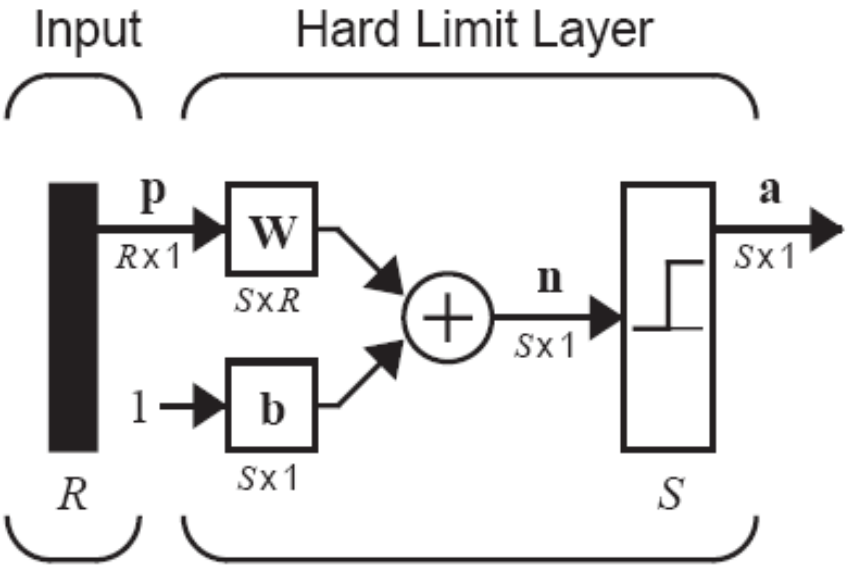
$$ b_1 = -{}_1W^T p = -\begin{bmatrix} -3 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 1 $$

$$ b_2 = -{}_2W^T p = -\begin{bmatrix} 1 & -2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0 $$

$$ W = \begin{bmatrix} {}_1W^T \\ {}_2W^T \end{bmatrix} = \begin{bmatrix} -3 & -1 \\ 1 & -2 \end{bmatrix} \text{ and } b = \begin{bmatrix} 1 \\ 0 \end{bmatrix} $$

Summary

- Perceptron Architecture



$$\mathbf{a} = \mathbf{hardlim}(\mathbf{Wp}+\mathbf{b})$$

a = hardlim(Wp + b)

$$a_i = hard\lim(n_i) = hard\lim({}_i w^T p + b_i)$$

$$W = \begin{bmatrix} {}_1 w^T \\ {}_2 w^T \\ \vdots \\ {}_S w^T \end{bmatrix}$$

Summary (2)

- Decision Boundary

The decision boundary is always orthogonal to the weight vector.
Single-layer perceptrons can only classify linearly separable vectors.

- Perceptron Learning Rule

$$W^{new} = W^{old} + ep^T$$

$$b^{new} = b^{old} + e$$

$$\text{where } e = t - a$$

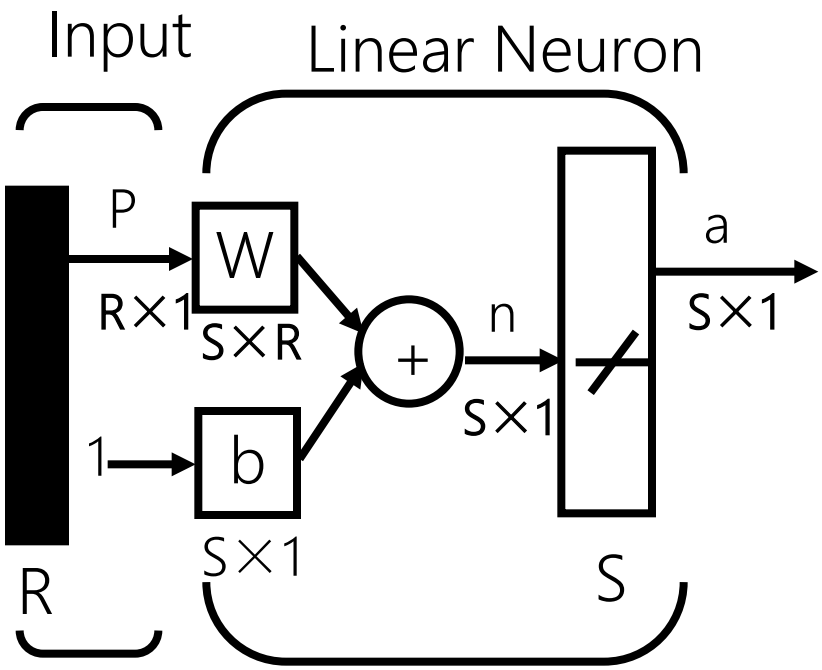
Widrow-Hoff Learning

(Least Mean Square Algorithm)

最小均方算法

Prof. Bernard Widrow

ADALINE (ADAptive LInear NEuron) Network



$$\mathbf{a} = \mathbf{purelin}(\mathbf{Wp} + \mathbf{b}) = \mathbf{Wp} + \mathbf{b}$$

a=purelin(Wp+b)

$$a_i = purelin(n_i) = purelin({}_i\mathbf{w}^T\mathbf{p} + b_i) = {}_i\mathbf{w}^T\mathbf{p} + b_i$$

$${}_i\mathbf{w} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix}$$

Taylor Series Expansion

$$F(x) = F(x^*) + \frac{d}{dx}F(x)\Big|_{x=x^*}(x - x^*)$$

$$+ \frac{1}{2}\frac{d^2}{dx^2}F(x)\Big|_{x=x^*}(x - x^*)^2 + \cdots$$

$$+ \frac{1}{n!}\frac{d^n}{dx^n}F(x)\Big|_{x=x^*}(x - x^*)^n + \cdots$$

Example

$$F(x) = e^{-x}$$

Taylor series of $F(x)$ about $x^* = 0$:

$$F(x) = e^{-x} = e^{-0} - e^{-0}(x-0) + \frac{1}{2}e^{-0}(x-0)^2 - \frac{1}{6}e^{-0}(x-0)^3 + \ldots$$

$$F(x) = 1 - x + \frac{1}{2}x^2 - \frac{1}{6}x^3 + \ldots$$

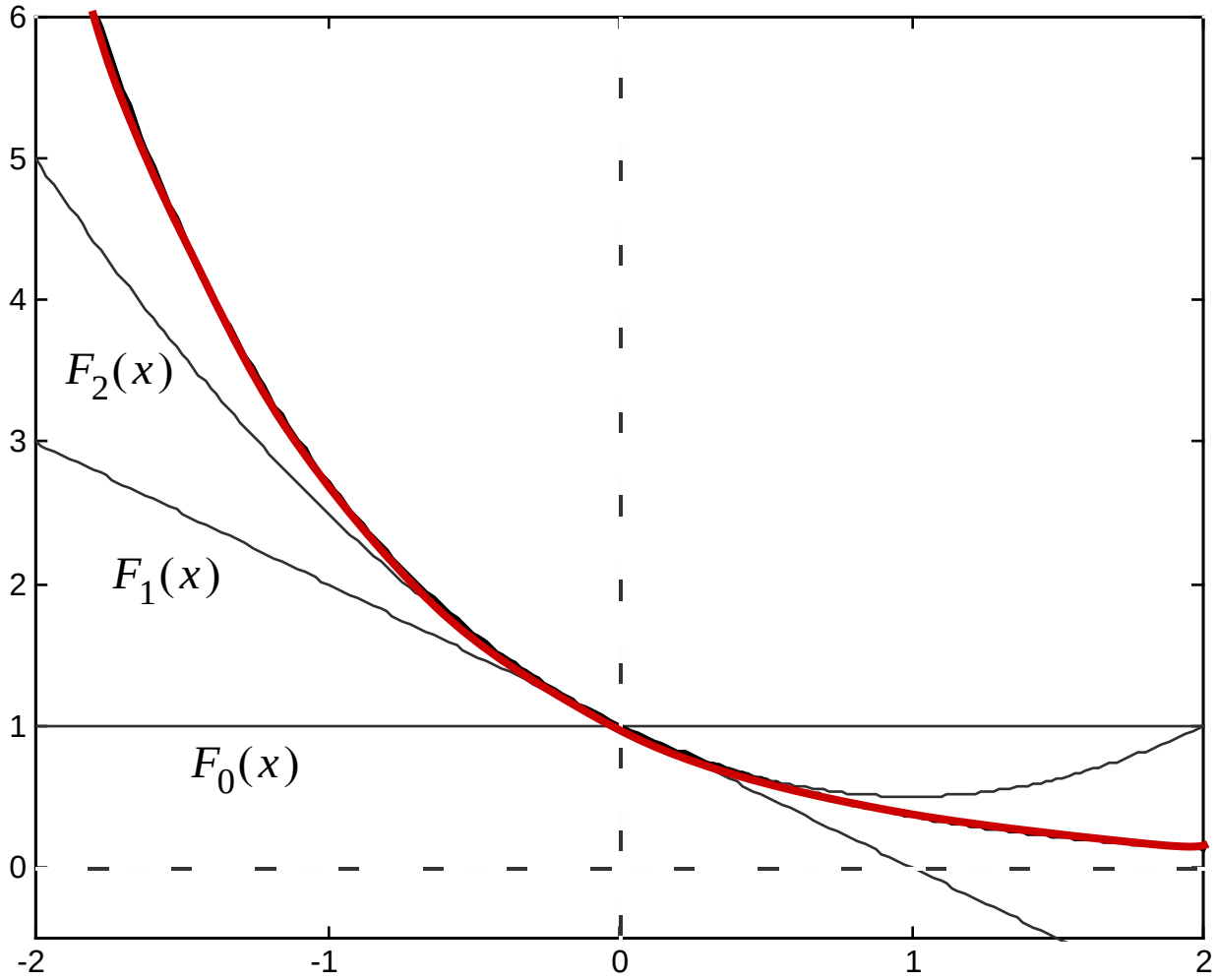
Taylor series approximations:

$$F(x) \approx F_0(x) = 1$$

$$F(x) \approx F_1(x) = 1 - x$$

$$F(x) \approx F_2(x) = 1 - x + \frac{1}{2}x^2$$

Plot of Approximations



Vector Case

$$F(x) = F(x_1, x_2, \ldots x_n)$$

$$F(x) = F(x^*) + \frac{d}{dx} F(x)\Big|_{x=x^*} (x - x^*)$$

$$+ \frac{1}{2} \frac{d^2}{dx^2} F(x)\Big|_{x=x^*} (x - x^*)^2 + \ldots$$

$$+ \frac{1}{n!} \frac{d^n}{dx^n} F(x)\Big|_{x=x^*} (x - x^*)^n + \ldots$$

Matrix Form

$$F(x) = F(x^*) + \nabla F(x)^T \big|_{x=x^*} (x - x^*)$$

$$+ \frac{1}{2}(x - x^*)^T \nabla^2 F(x)^T \big|_{x=x^*} (x - x^*) + \cdots$$

Gradient

$$\nabla F(x) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(x) \\ \frac{\partial}{\partial x_2} F(x) \\ \vdots \\ \frac{\partial}{\partial x_n} F(x) \end{bmatrix}$$

Hessian

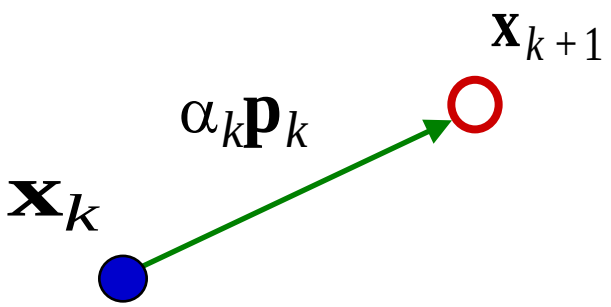
$$\nabla^2 F(x) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2} F(x) & \frac{\partial^2}{\partial x_1 \partial x_2} F(x) & \cdots & \frac{\partial^2}{\partial x_1 \partial x_n} F(x) \\ \frac{\partial^2}{\partial x_2 \partial x_1} F(x) & \frac{\partial^2}{\partial x_2^2} F(x) & \cdots & \frac{\partial^2}{\partial x_2 \partial x_n} F(x) \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2}{\partial x_n \partial x_1} F(x) & \frac{\partial^2}{\partial x_n \partial x_2} F(x) & \cdots & \frac{\partial^2}{\partial x_n^2} F(x) \end{bmatrix}$$

Basic Optimization Algorithm

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$$

or

$$\Delta \mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k \mathbf{p}_k$$



$\mathbf{p}_k$ - Search Direction

α_k - Learning Rate

Steepest Descent（最速下降）

Choose the next step so that the function decreases:

$$F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k)$$

For small changes in $\mathbf{x}$ we can approximate $F(\mathbf{x})$:

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta\mathbf{x}_k$$

where

$$\mathbf{g}_k \equiv \nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_k}$$

If we want the function to decrease:

$$\mathbf{g}_k^T \Delta\mathbf{x}_k = \alpha_k \mathbf{g}_k^T \mathbf{p}_k < 0$$

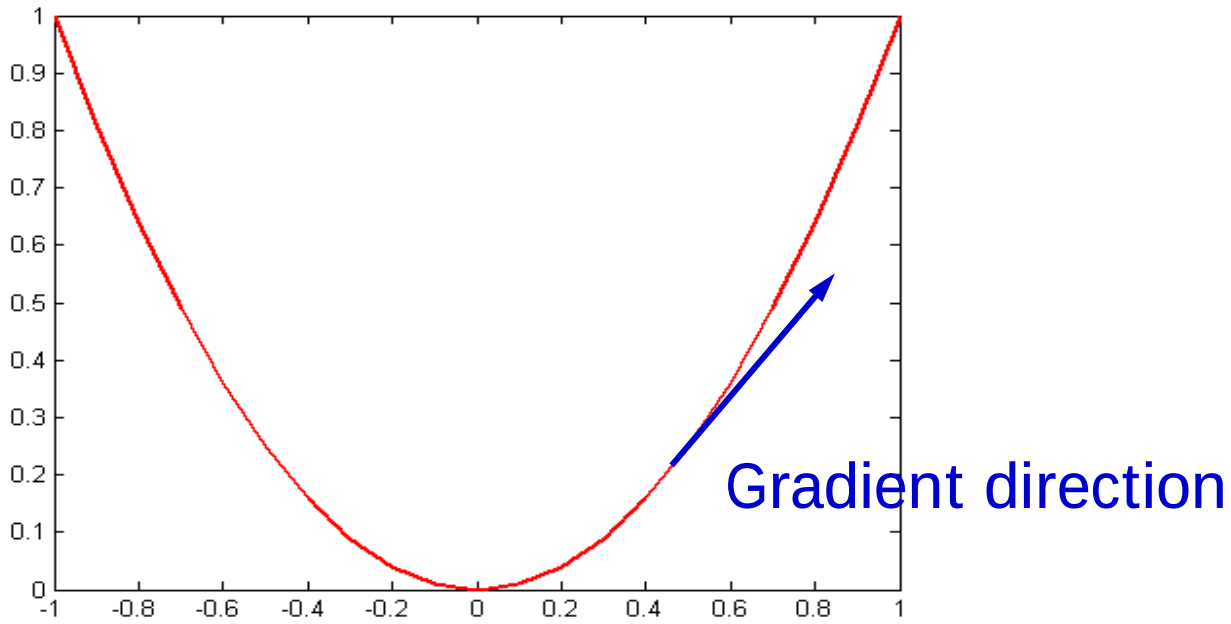
We can maximize the decrease by choosing:

$$\mathbf{p}_k = -\mathbf{g}_k$$

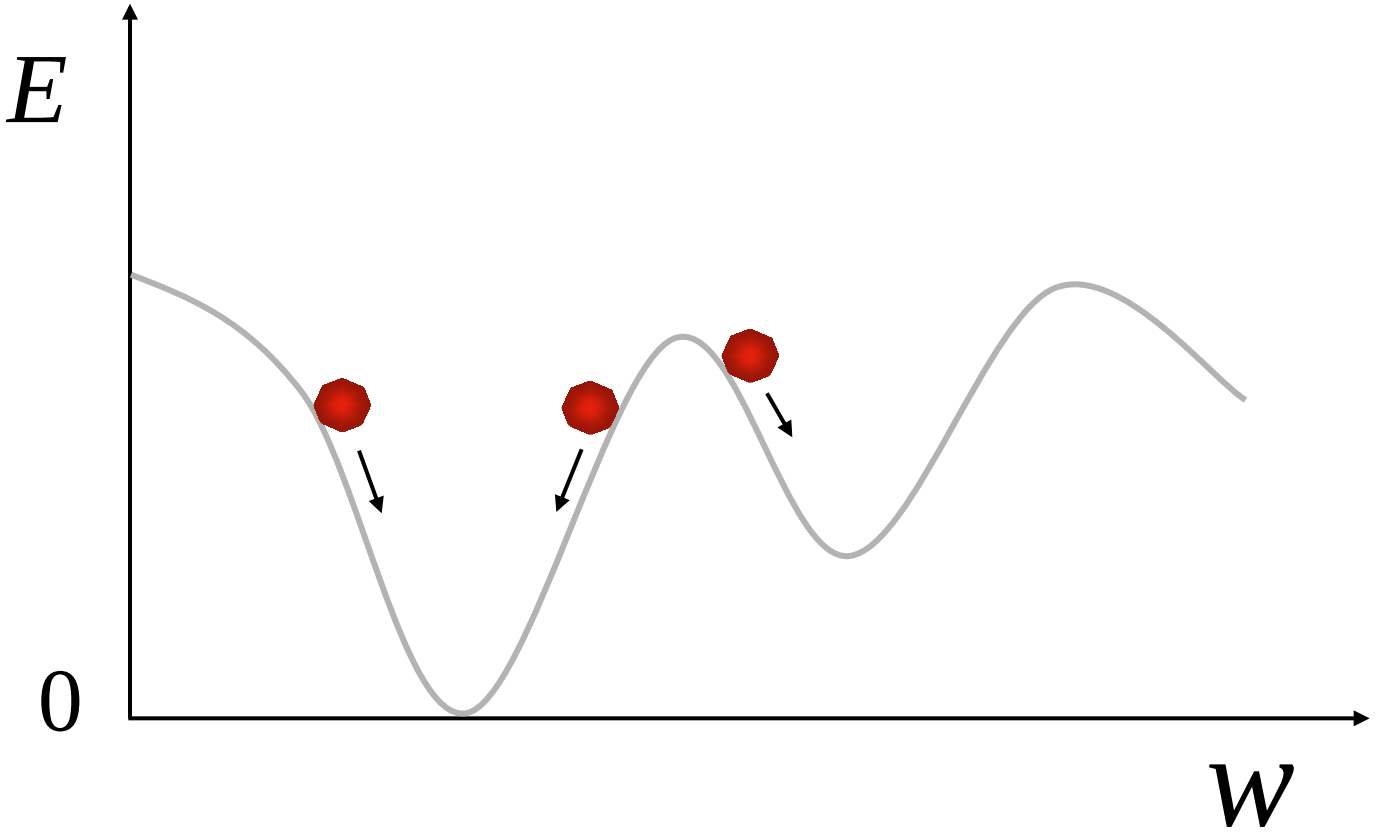
$$\boxed{\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k}$$

Gradient Method

Gradient method could be thought of as a ball rolling down from a hill: the ball will roll down and finally stop at the valley



Local Minimum Problem



Example

$$F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2 + x_1$$

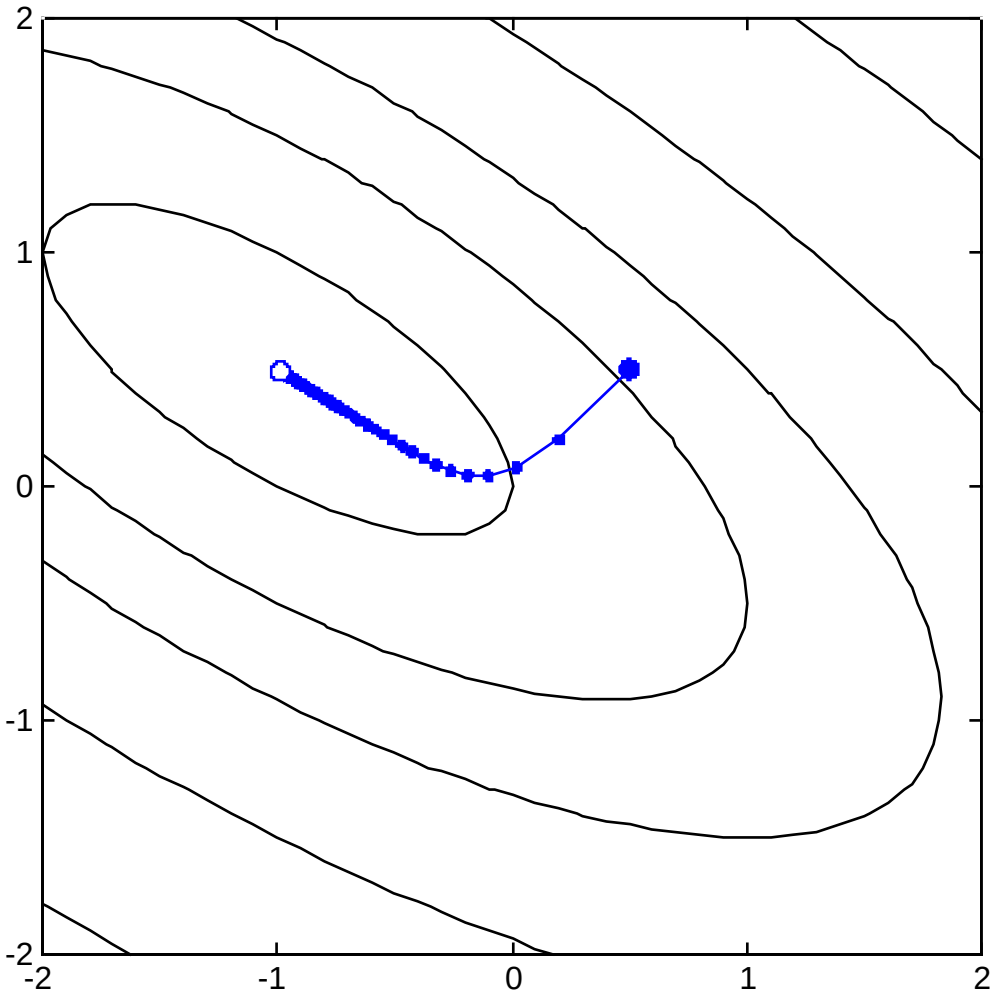
$$\mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \qquad \alpha = 0.1$$

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 + 2x_2 + 1 \\ 2x_1 + 4x_2 \end{bmatrix} \qquad \mathbf{g}_0 = \nabla F(\mathbf{x})\Big|_{\mathbf{x} = \mathbf{x}_0} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha \mathbf{g}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - 0.1 \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix}$$

$$\mathbf{x}_2 = \mathbf{x}_1 - \alpha \mathbf{g}_1 = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} - 0.1 \begin{bmatrix} 1.8 \\ 1.2 \end{bmatrix} = \begin{bmatrix} 0.02 \\ 0.08 \end{bmatrix}$$

Plot



Stable Learning Rates (Quadratic)

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{d}^T\mathbf{x} + c$$

$$\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha\mathbf{g}_k = \mathbf{x}_k - \alpha(\mathbf{A}\mathbf{x}_k + \mathbf{d}) \Rightarrow \mathbf{x}_{k+1} = [\mathbf{I} - \alpha\mathbf{A}]\mathbf{x}_k - \alpha\mathbf{d}$$

Stability is determined by the eigenvalues of this matrix.

This is linear dynamic system, which will be stable if the eigenvalues of the matrix [**I** - α**A**] are less than one in magnitude .

Stable Learning Rates (Quadratic) -2

$$[\mathbf{I} - \alpha\mathbf{A}]\mathbf{z}_i = \mathbf{z}_i - \alpha\mathbf{A}\mathbf{z}_i = \mathbf{z}_i - \alpha\lambda_i\mathbf{z}_i = \underbrace{(1 - \alpha\lambda_i)}\mathbf{z}_i$$

(λ_i - eigenvalue of $\mathbf{A}$)

Eigenvalues of [$\mathbf{I}$ - $\alpha\mathbf{A}$].

Stability Requirement:

$$\left|(1 - \alpha\lambda_i)\right| < 1 \qquad \alpha < \frac{2}{\lambda_i}$$

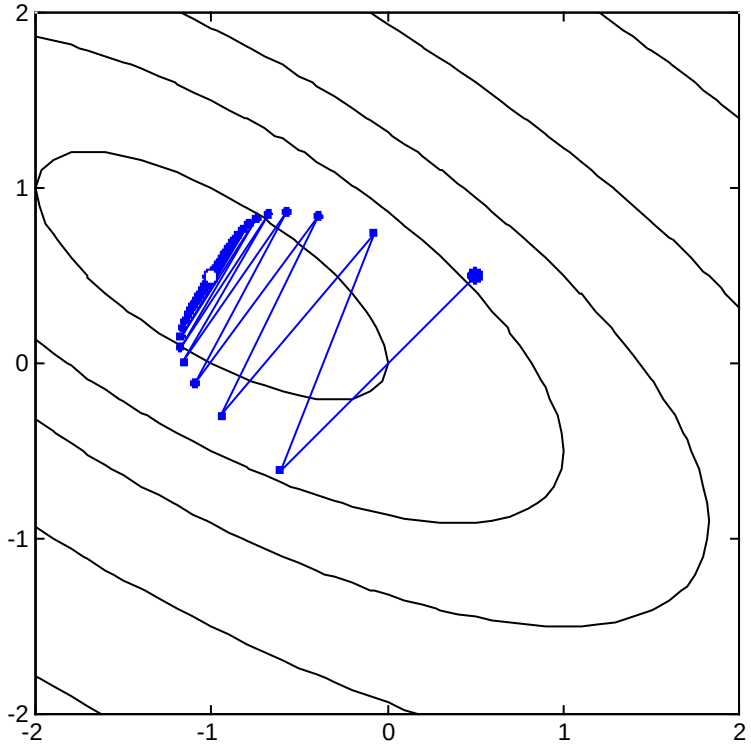
$$\alpha < \frac{2}{\lambda_{max}}$$

Example

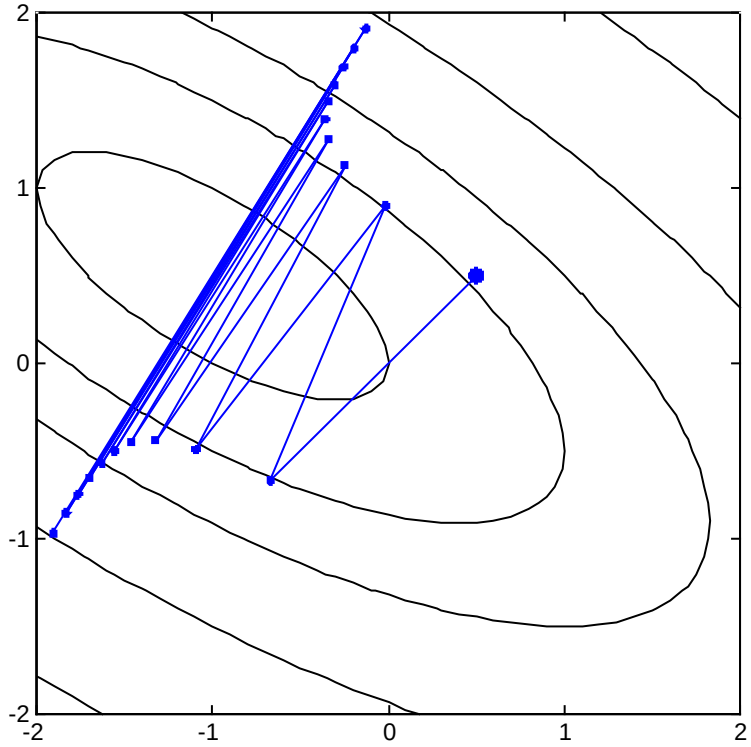
$$\mathbf{A} = \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \qquad \left\{(\lambda_1 = 0.764), \left(\mathbf{z}_1 = \begin{bmatrix} 0.851 \\ -0.526 \end{bmatrix}\right)\right\}, \left\{\lambda_2 = 5.24, \left(\mathbf{z}_2 = \begin{bmatrix} 0.526 \\ 0.851 \end{bmatrix}\right)\right\}$$

$$\alpha < \frac{2}{\lambda_{max}} = \frac{2}{5.24} = 0.38$$

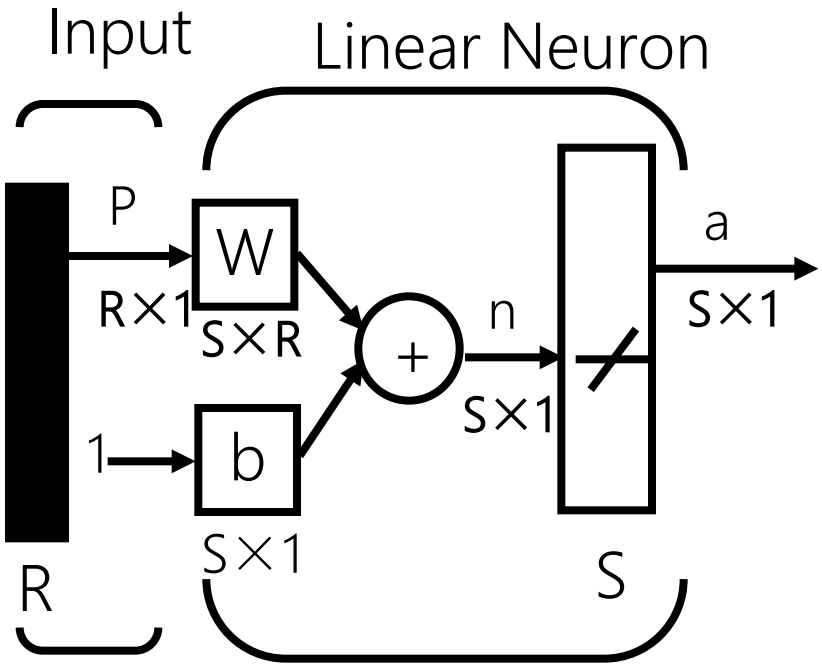
$\alpha = 0.37$



$\alpha = 0.39$



ADALINE (ADAptive LInear NEuron) Network



$$\mathbf{a} = \mathbf{purelin}(\mathbf{Wp} + \mathbf{b}) = \mathbf{Wp} + \mathbf{b}$$

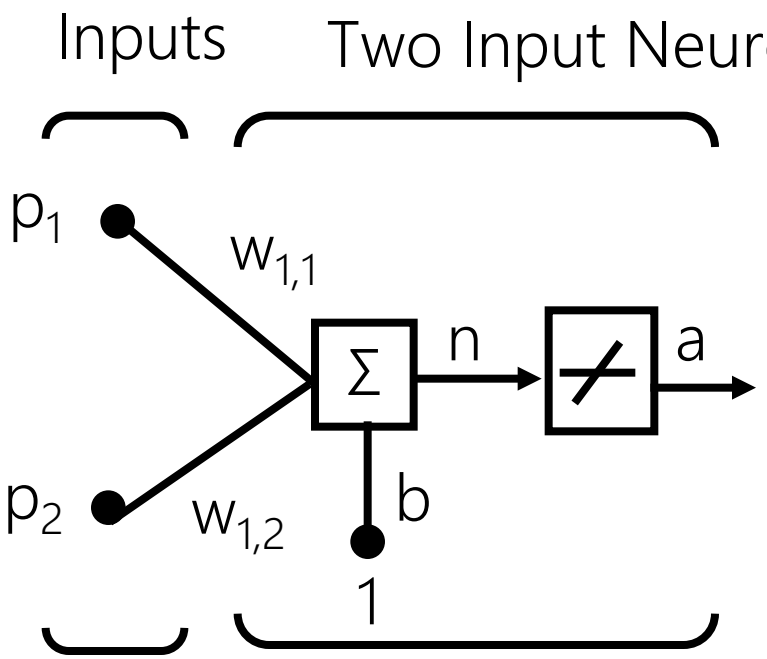
a=purelin(Wp+b)

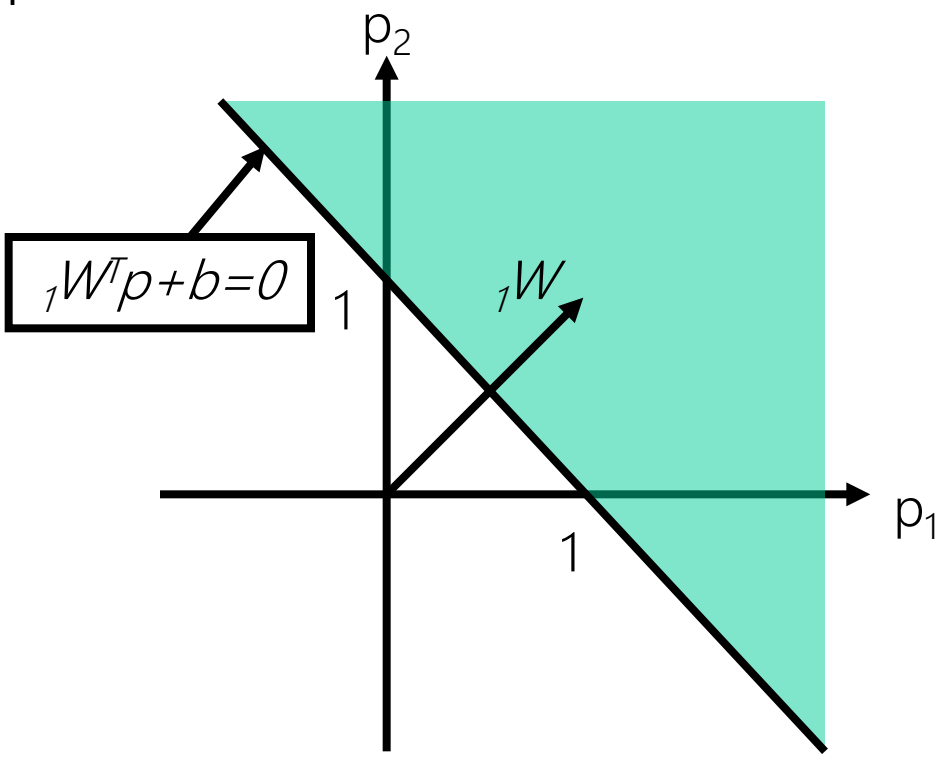
$$a_i = purelin(n_i) = purelin({}_i\mathbf{w}^T\mathbf{p} + b_i) = {}_i\mathbf{w}^T\mathbf{p} + b_i$$

$${}_i\mathbf{w} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix}$$

Two-Input ADALINE

$w_{1,1} = 1 \qquad w_{1,2} = 1 \qquad b = -1$





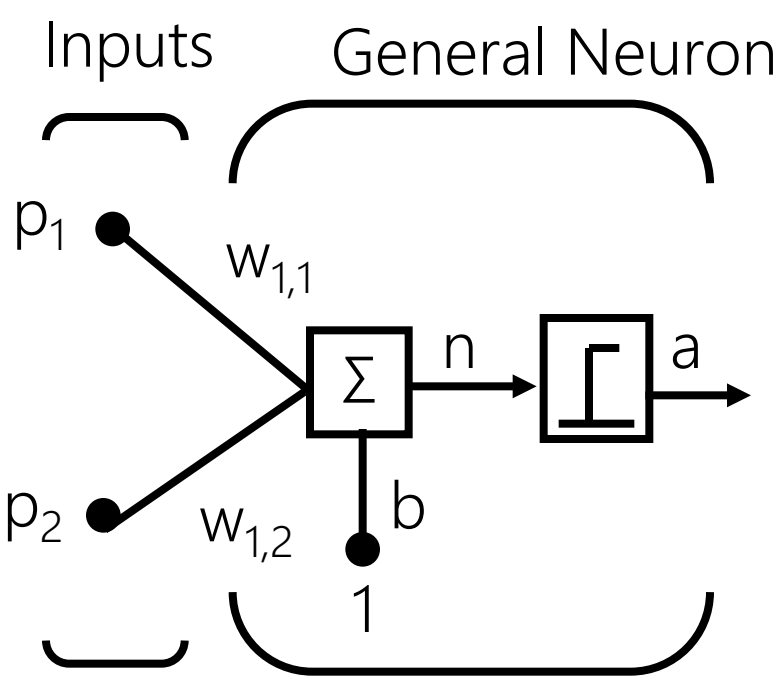
a=*purelin*(Wp+b)

$$a = purelin(n) = purelin({}_1\mathbf{w}^T\mathbf{p} + b) = {}_1\mathbf{w}^T\mathbf{p} + b$$

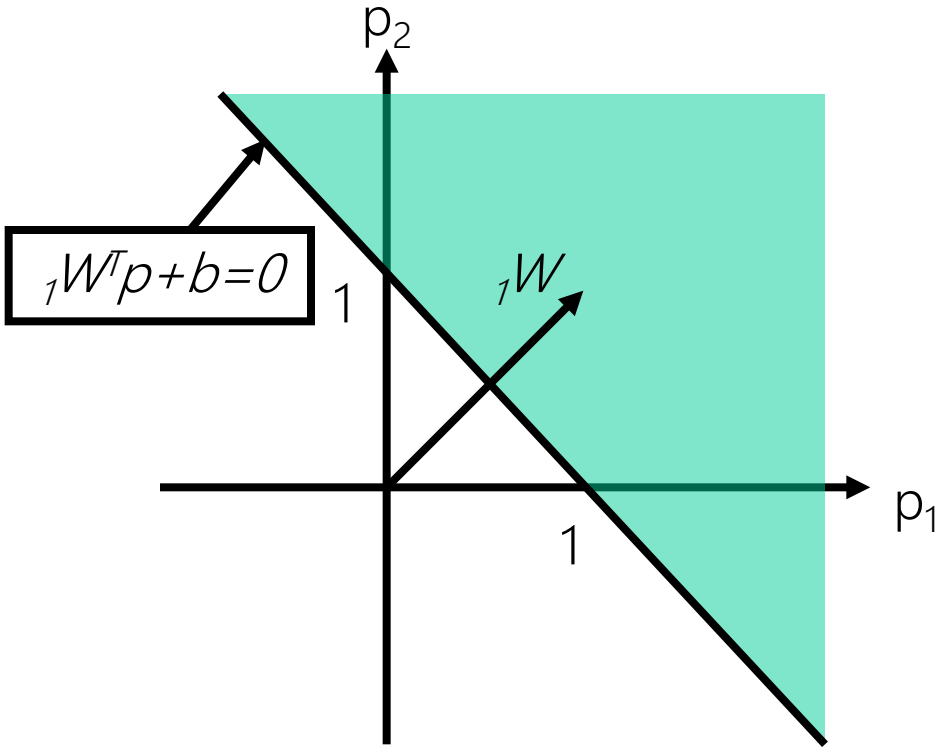
$$a = {}_1\mathbf{w}^T\mathbf{p} + b = w_{1,1}p_1 + w_{1,2}p_2 + b$$

Single-Neuron Perceptron

$w_{1,1} = 1 \qquad w_{1,2} = 1 \qquad b = -1$



a = *hardlim*(Wp+b)



$$a = hardlim({}_1\mathbf{w}^{T}\mathbf{p} + b) = hardlim(w_{1,1}p_1 + w_{1,2}p_2 + b)$$

Mean Square Error

Training Set:

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

Input: $\mathbf{p}_q$ Target: $\mathbf{t}_q$

Notation:

$$\mathbf{x} = \begin{bmatrix} {}_1\mathbf{w} \\ b \end{bmatrix} \quad \mathbf{z} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} \quad a = {}_1\mathbf{w}^T\mathbf{p} + b \Rightarrow a = \mathbf{x}^T\mathbf{z}$$

Mean Square Error (均方误差):

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2] = E[(t-\mathbf{x}^T\mathbf{z})^2]$$

$E[\,]$ 表示期望值

Error Analysis

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2] = E[(t-\mathbf{x}^T\mathbf{z})^2]$$

$$F(\mathbf{x}) = E[t^2 - 2t\mathbf{x}^T\mathbf{z} + \mathbf{x}^T\mathbf{z}\mathbf{z}^T\mathbf{x}]$$

$$F(\mathbf{x}) = E[t^2] - 2\mathbf{x}^T E[t\mathbf{z}] + \mathbf{x}^T E[\mathbf{z}\mathbf{z}^T]\mathbf{x}$$

$$\boxed{F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x}}$$

$$c = E[t^2] \qquad \mathbf{h} = E[t\mathbf{z}] \qquad \mathbf{R} = E[\mathbf{z}\mathbf{z}^T]$$

The mean square error for the ADALINE Network is a quadratic function:

$$F(\mathbf{x}) = c + \mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} \qquad \mathbf{d} = -2\mathbf{h} \qquad \mathbf{A} = 2\mathbf{R}$$

向量h给出输入向量和对应目标输出之间的相关系数
R 是输入的相关矩阵

Quadratic Functions

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{d}^T\mathbf{x} + c \qquad \text{(Symmetric } \mathbf{A}\text{)}$$

Gradient and Hessian:

Useful properties of gradients:

$$\nabla(\mathbf{h}^T\mathbf{x}) = \nabla(\mathbf{x}^T\mathbf{h}) = \mathbf{h}$$

$$\nabla\mathbf{x}^T\mathbf{Q}\mathbf{x} = \mathbf{Q}\mathbf{x} + \mathbf{Q}^T\mathbf{x} = 2\mathbf{Q}\mathbf{x} \quad \text{(for symmetric } \mathbf{Q}\text{)}$$

Gradient of Quadratic Function:

$$\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d}$$

Hessian of Quadratic Function:

$$\nabla^2 F(\mathbf{x}) = \mathbf{A}$$

Stationary Point

Hessian Matrix:

$$\mathbf{A} = 2\mathbf{R}$$

The correlation matrix **R** must be at least positive semidefinite. If there are any zero eigenvalues, the performance index will either have a weak minimum or else no stationary point, otherwise there will be a unique global minimum **x***.

$$\nabla F(\mathbf{x}) = \nabla\left(c + \mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}\right) = \mathbf{d} + \mathbf{A}\mathbf{x} = -2\mathbf{h} + 2\mathbf{R}\mathbf{x}$$

$$-2\mathbf{h} + 2\mathbf{R}\mathbf{x} = \mathbf{0}$$

If R is positive definite:

$$\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h}$$

Approximate Steepest Descent

Approximate mean square error (one sample):

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2] = E[(t-\mathbf{x}^T\mathbf{z})^2]$$

$$\hat{F}(\mathbf{x}) = (t(k)-a(k))^2 = e^2(k)$$

Approximate gradient:

$$\hat{\nabla} F(\mathbf{x}) = \nabla e^2(k)$$

$$[\nabla e^2(k)]_j = \frac{\partial e^2(k)}{\partial w_{1,j}} = 2e(k)\frac{\partial e(k)}{\partial w_{1,j}} \qquad j = 1, 2, \ldots, R$$

$$[\nabla e^2(k)]_{R+1} = \frac{\partial e^2(k)}{\partial b} = 2e(k)\frac{\partial e(k)}{\partial b}$$

Approximate Gradient Calculation

$$\frac{\partial e(k)}{\partial w_{1,j}} = \frac{\partial [t(k) - a(k)]}{\partial w_{1,j}} = \frac{\partial}{\partial w_{1,j}} [t(k) - ({}_1\mathbf{w}^T \mathbf{p}(k) + b)]$$

$$\frac{\partial e(k)}{\partial w_{1,j}} = \frac{\partial}{\partial w_{1,j}} \left[t(k) - \left(\sum_{i=1}^{R} w_{1,i} p_i(k) + b \right) \right]$$

$$\frac{\partial e(k)}{\partial w_{1,j}} = -p_j(k) \qquad \frac{\partial e(k)}{\partial b} = -1$$

$$\hat{\nabla} F(\mathbf{x}) = \nabla e^2(k) = -2e(k)\mathbf{z}(k)$$

LMS Algorithm

LMS Learning Rule

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_k}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + 2\alpha e(k)\mathbf{z}(k)$$

$${}_1\mathbf{w}(k+1) = {}_1\mathbf{w}(k) + 2\alpha e(k)\mathbf{p}(k)$$

$$b(k+1) = b(k) + 2\alpha e(k)$$

Perceptron Learning Rule

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e\mathbf{p} = {}_1\mathbf{w}^{old} + (t-a)\mathbf{p}$$

$$b^{new} = b^{old} + e$$

Multiple-Neuron Case

$$_i\mathbf{w}(k+1) = {}_i\mathbf{w}(k) + 2\alpha e_i(k)\mathbf{p}(k)$$

$$b_i(k+1) = b_i(k) + 2\alpha e_i(k)$$

Matrix Form:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha\mathbf{e}(k)\mathbf{p}^T(k)$$

$$\mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha\mathbf{e}(k)$$

Example

Banana $\left\{ \mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} -1 \end{bmatrix} \right\}$ Apple $\left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix} 1 \end{bmatrix} \right\}$

$$\mathbf{R} = E[\mathbf{p}\mathbf{p}^T] = \frac{1}{2}\mathbf{p}_1\mathbf{p}_1^T + \frac{1}{2}\mathbf{p}_2\mathbf{p}_2^T$$ (Input Correlation Matrix)

$$\mathbf{R} = \frac{1}{2}\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}\begin{bmatrix} -1 & 1 & -1 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}\begin{bmatrix} 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

$$\lambda_1 = 1.0, \qquad \lambda_2 = 0.0, \qquad \lambda_3 = 2.0$$

$$\alpha < \frac{1}{\lambda_{max}} = \frac{1}{2.0} = 0.5$$

Iteration One

Banana:

$$a(0) = \mathbf{W}(0)\mathbf{p}(0) = \mathbf{W}(0)\mathbf{p}_1 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} = 0$$

$$e(0) = t(0) - a(0) = t_1 - a(0) = -1 - 0 = -1$$

$$\mathbf{W}(1) = \mathbf{W}(0) + 2\alpha e(0)\mathbf{p}^T(0)$$

$$\mathbf{W}(1) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} + 2(0.2)(-1) \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}^T = \begin{bmatrix} 0.4 & -0.4 & 0.4 \end{bmatrix}$$

Iteration Two

Apple:

$$a(1) = \mathbf{W}(1)\mathbf{p}(1) = \mathbf{W}(1)\mathbf{p}_2 = \begin{bmatrix} 0.4 & -0.4 & 0.4 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = -0.4$$

$$e(1) = t(1) - a(1) = t_2 - a(1) = 1 - (-0.4) = 1.4$$

$$\mathbf{W}(2) = \begin{bmatrix} 0.4 & -0.4 & 0.4 \end{bmatrix} + 2(0.2)(1.4) \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}^T = \begin{bmatrix} 0.96 & 0.16 & -0.16 \end{bmatrix}$$

Iteration Three

$$a(2) = \mathbf{W}(2)\mathbf{p}(2) = \mathbf{W}(2)\mathbf{p}_1 = \begin{bmatrix} 0.96 & 0.16 & -0.16 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} = -0.64$$

$$e(2) = t(2) - a(2) = t_1 - a(2) = -1 - (-0.64) = -0.36$$

$$\mathbf{W}(3) = \mathbf{W}(2) + 2\alpha e(2)\mathbf{p}^T(2) = \begin{bmatrix} 1.1040 & 0.0160 & -0.0160 \end{bmatrix}$$

$$\mathbf{W}(\infty) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

Perceptron rule stops as soon as the patterns are correctly classified, even though some patterns may be close to the boundaries.

The LMS algorithm minimizes the mean square error. Therefore it tries to move the decision boundaries as far from the reference patterns as possible.

Recommended Textbook

- *Neural Network Design*, 1st ed. Martin T. Hagan, Howard. B. Demuth, and Mark H. Beale, PWS Publ. Company, 1996

  (**神经网路设计,机械工业出版社**,2002)

小结

- 学习规则
- 基于记忆的学习
- 人工神经网络结构
- 误差修正学习
- 感知器
- 梯度下降法
- Delta学习规则

发邮件给助教

请下课后发邮件给助教刘伟（liuwei@liujr.com）：

- 你的常用邮箱
- 姓名
- 学号

第一次作业（2月28日18:00之前提交）

- 2月28日18:00之前提交（上传到课程网站）
- 文件名：姓名+学号

课件下载和作业上传FTP

- FTP地址：

ftp://bcmi.sjtu.edu.cn:2122/

User name: mpml

Password: mpml