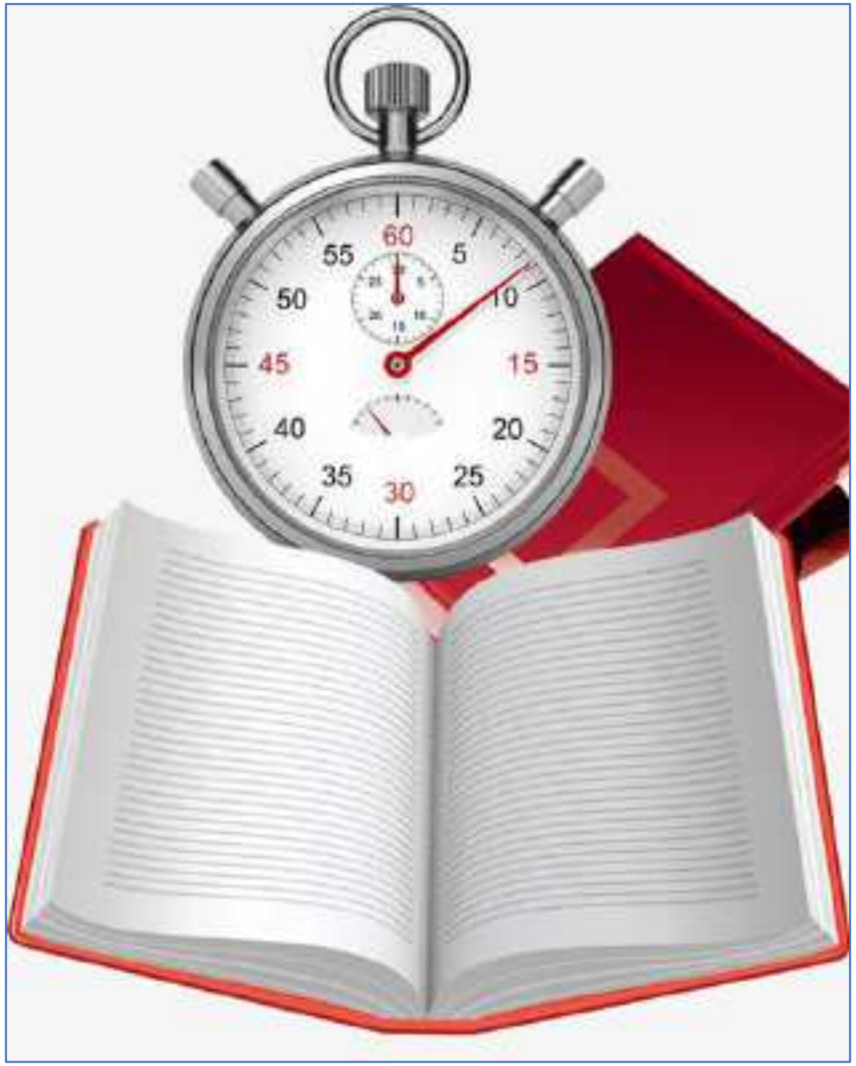
CS 103 -12
Support Vector Machine and Machine Learning

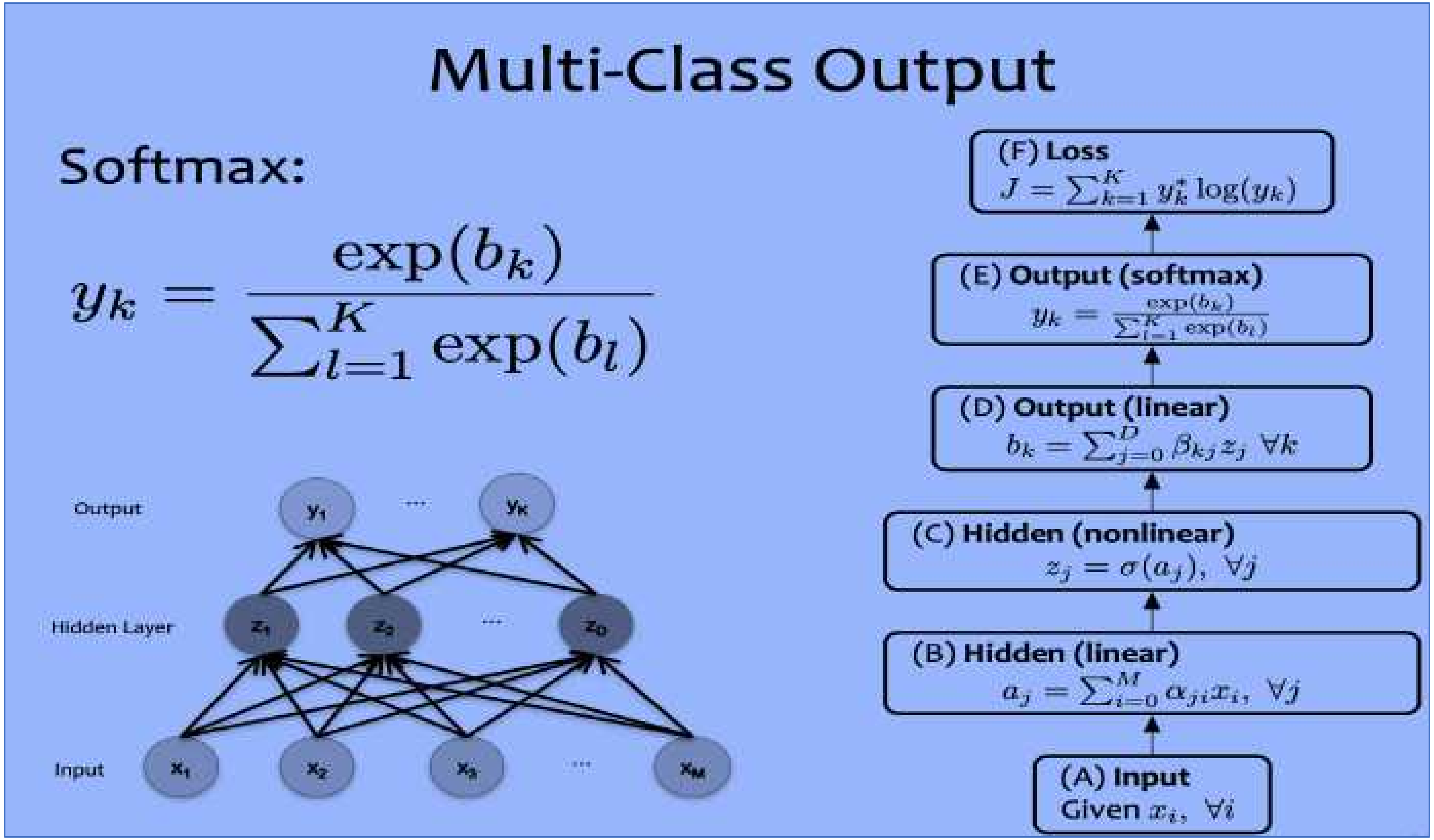
Jimmy Liu 刘江

2020-12-04

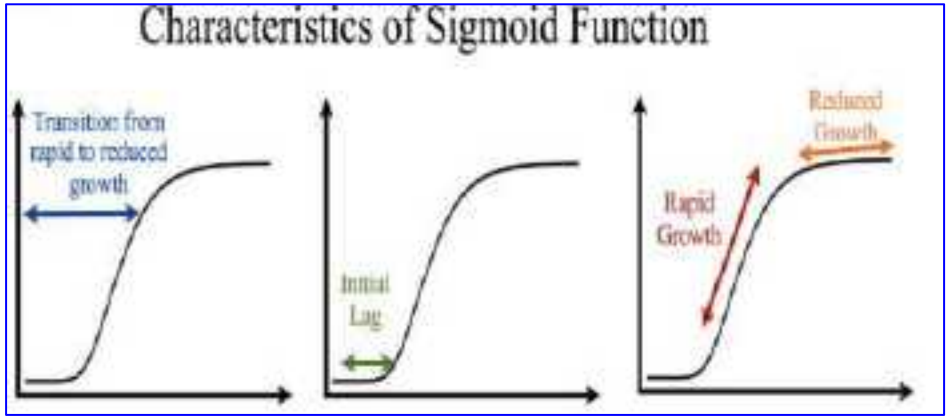
Review

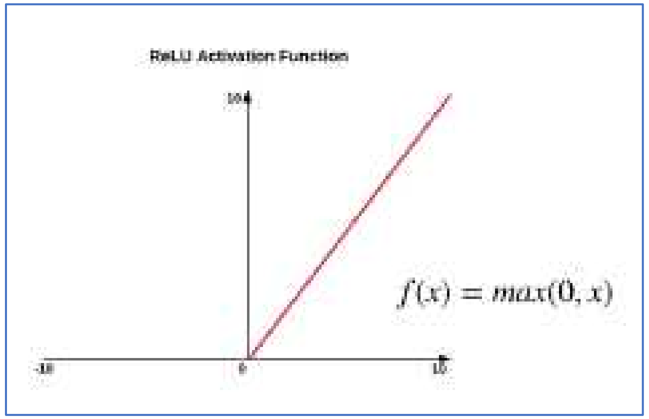


Multiple Classification with Softmax (Probability)



Sigmoid and ReLU





Sigmoid: not blowing up activation

Relu : not vanishing gradient

Relu : More computationally efficient to compute than Sigmoid like functions since Relu just needs to pick max(0, x) and not perform expensive exponential operations as in Sigmoids

Relu : In practice, networks with Relu tend to show better convergence performance than sigmoid.

BP Algorithm with Sigmoid Transfer Function

1. Set initial weight and bias a random small number

$$w_{ij}^k(t),\quad \theta_i^k(t),(k=1,\ldots,m;i=1,\ldots,p_k;j=1,\ldots,p_{k-1};t=0)$$

2. Select a sample x from the N input samples and corresponding output y

$$\boldsymbol{x}=[x_1,x_2,\ldots,x_{p1}]^{\mathrm{T}} \qquad y_i \quad (i=1,\ldots,p_m)$$

3. Calculate all the outputs in all layer

$$y_i^k \quad (i=1,\ldots,p_k;\ k=1,\ldots,m)$$

4 Calculate the output error

$$e_i=y_i-y_i^m \quad (i=1,\ldots,p_m)$$

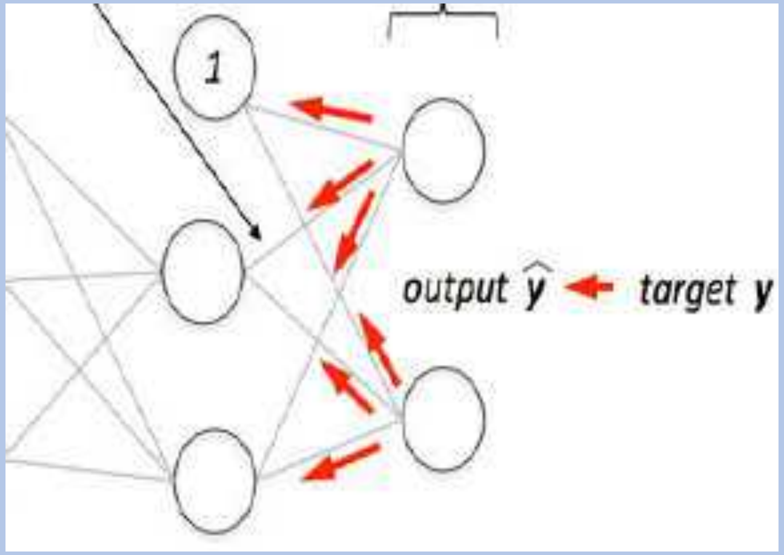
BP Algorithm with Sigmoid Transfer Function

- 5. Update the weight for both output and hidden layers

$$\Delta w_{ij}^{k} = -\alpha d_i^{k} y_j^{k-1}$$

$$d_i^{m} = y_i^{m}(1-y_i^{m})(y_i^{m}-y_i)$$

$$d_i^{k} = y_i^{k}(1-y_i^{k})\sum_{l=1}^{p_{k+1}} w_{li}^{k+1} d_l^{k+1}$$



- 6 t=t+1, repeat 2-5 until the error rate for the N samples reaches the set target or stops decreasing .

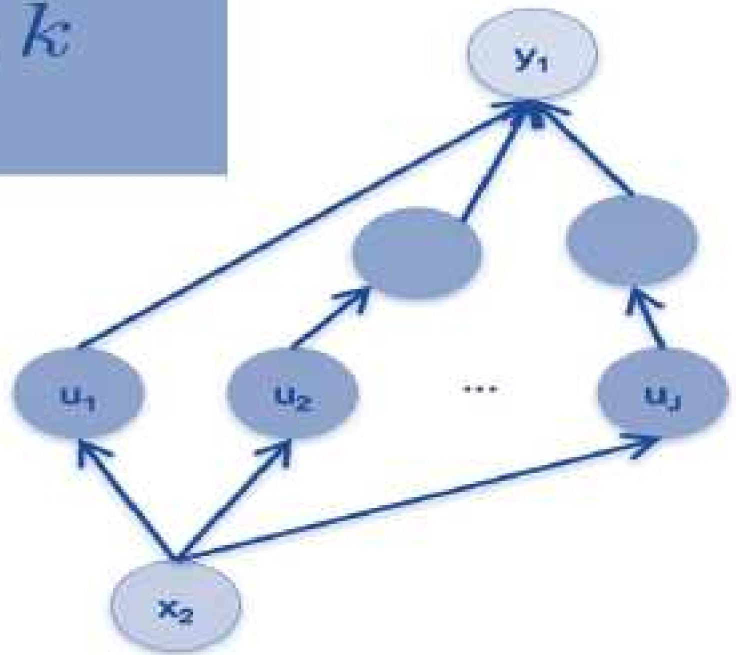
Chain Rule in General

Given: $\boldsymbol{y} = g(\boldsymbol{u})$ **and** $\boldsymbol{u} = h(\boldsymbol{x})$.

Chain Rule:

$$\frac{dy_i}{dx_k} = \sum_{j=1}^{J} \frac{dy_i}{du_j}\frac{du_j}{dx_k}, \qquad \forall i, k$$

Backpropagation is just repeated application of the **chain rule** from Calculus 101.



Machine Learning in General

1. Given training data:

$$\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$$

2. Choose each of these:

– Decision function

$$\hat{\boldsymbol{y}} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

– Loss function

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$$

3. Define goal:

$$y = F(x; w)$$

$$\arg\min_{\{w^t\}_{t=1}^{T}} \sum_{t=1}^{T} \sum_{i=1}^{N} \ell(y_i^t, F(x_i^t; w^t)) + \gamma(w^t)$$

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

4. Train with SGD:

(take small steps opposite the gradient)

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

BP and Perceptron Weight Updating

There are three core differences for the updating rule between BP and Perceptron :

1) The none-linear activation of the BP hidden unit is used.

2) The BP rule contains a term for the gradient of the activation function and use gradient descent to minimize the error

3) BP can not use Perceptron Learning Rule as no teacher values are possible for hidden units

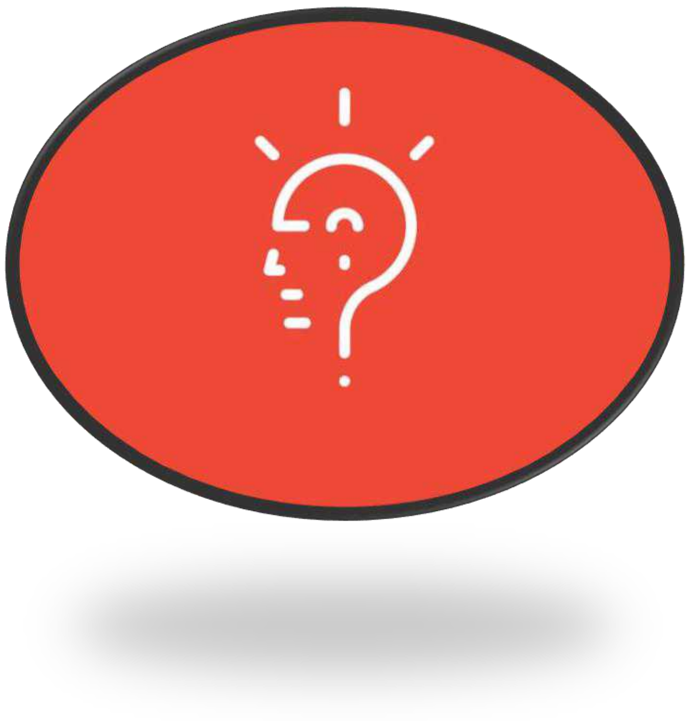
Gradient Descent Vs Stochastic Gradient Descent

GD: This will update the weights only after calculating the mean loss of all the samples. Hence in such situation it will become very costly operation and it will converge very slowly.

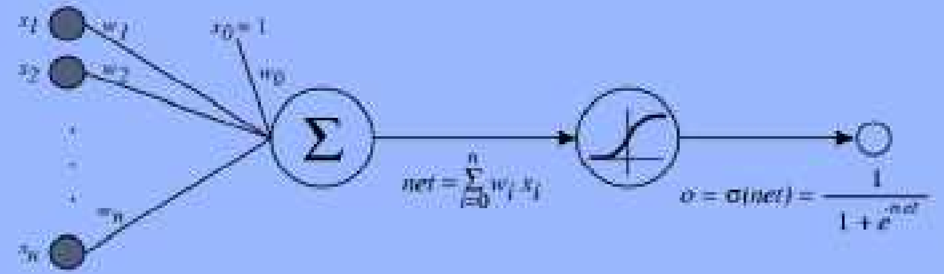
Batch GD/Mini-Batch GD: Alternative of GD. Selects a batch size and overcome the above said problem of GD. But still executes in batch.

SGD: Update weights with every sample and converge very fast.

Any Question?



Homework: Prove the derivative formula of Sigmoid



$\sigma(x)$ is the sigmoid function

$$\frac{1}{1+e^{-x}}$$

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

Prove:

$$
\begin{aligned}
\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\left[\frac{1}{1+e^{-x}}\right] \\
&= \frac{d}{dx}(1+e^{-x})^{-1} \\
&= -(1+e^{-x})^{-2}(-e^{-x}) \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{(1+e^{-x})-1}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right) \\
&= \frac{1}{1+e^{-x}} \cdot \left(1 - \frac{1}{1+e^{-x}}\right) \\
&= \sigma(x) \cdot (1-\sigma(x))
\end{aligned}
$$

Homework: Prove the derivative formula of ReLU Transfer Functions

Function	Formula	Derivative
Weighted input	$Z = XW$	$Z'(X) = W$ $Z'(W) = X$
ReLU activation	$R = max(0, Z)$	$R'(Z) = \begin{cases} 0 & Z < 0 \\ 1 & Z > 0 \end{cases}$
Cost function	$C = \frac{1}{2}(\hat{y} - y)^2$	$C'(\hat{y}) = (\hat{y} - y)$

Prove weighted input:

- The formula of Weighted input:

$$Z = XW$$

- Prove of derivative:

$$\begin{aligned} \frac{d}{dX}(Z) &= \frac{d}{dX}(XW) \\ &= W \end{aligned}$$

$$\begin{aligned} \frac{d}{dW}(Z) &= \frac{d}{dW}(XW) \\ &= X \end{aligned}$$

Homework: Prove the derivative formula of ReLU Transfer Functions

Prove ReLU activation:

- The formula of ReLU activation:

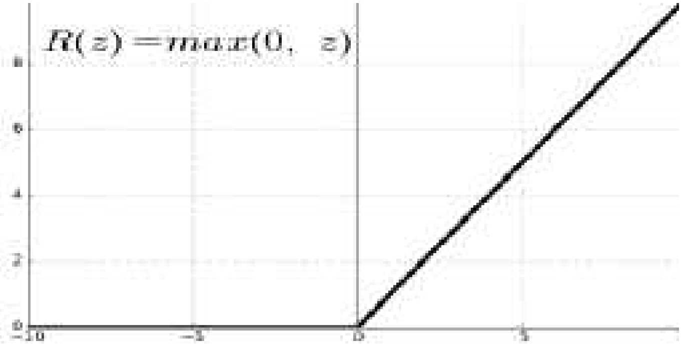
$$R = max(0,Z)$$

•When $Z < 0, R = 0:$

$$R'(Z) = \frac{d}{dZ}(R) = 0$$

•When $Z \geq 0, R = Z:$

$$R'(Z) = \frac{d}{dZ}(R) = 1$$



Prove cost function:

- The formula of Weighted input:

$$C = \frac{1}{2}(\hat{y} - y)^2$$

- Prove of derivative:

$$\begin{aligned} C'(\hat{y}) &= \frac{d}{d\hat{y}}(C) \\ &= \frac{d}{d\hat{y}}(\frac{1}{2}(\hat{y} - y)^2) \\ &= \frac{1}{2}(\hat{y} - y)(2) \\ &= (\hat{y} - y) \end{aligned}$$

prove1: $\frac{d\sigma(x)}{dx} = \frac{-(1+e^{-x})'}{(1+e^{-x})^2} = \frac{-e^{-x}\cdot(-1)}{(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})^2}$

$= \sigma(x)\times(1-\sigma(x))$

prove2: $z = xW$. z对x求偏导：$z'(x) = w$

z对w求偏导：$z'(w) = x$.

当 $z<0$ 时 $R=\max(0,z)=0 \Rightarrow R'=0$

当 $z>0$ 时 $R=\max(0,z)=z \Rightarrow R'=1$

$C=\frac{1}{2}(\hat{y}-y)^2 \Rightarrow C'(\hat{y}) = \frac{1}{2}\times 2\times(\hat{y}-y)\times 1$

$= (\hat{y}-y)$

Sigmoid:

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

$$\sigma'(x) = -\frac{1}{(1+e^{-x})^2}\cdot e^{-x}\cdot(-1)$$

$$= \frac{1}{1+e^{-x}}\cdot\frac{e^{-x}}{1+e^{-x}}$$

$$= \sigma(x)(1-\sigma(x))$$

ReLU:

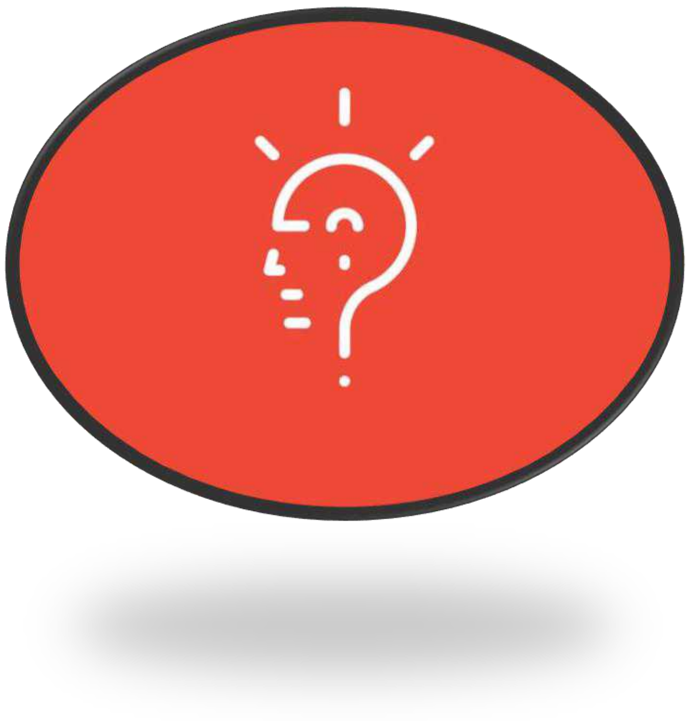
$$R(z) = \max(0, z)$$

$z<0$: $R'(z) = \frac{R(z+h)-R(z)}{h} = 0$

$z>0$: $R'(z) = \frac{R(z+h)-R(z)}{h} = \frac{h}{h} = 1$

$$\therefore R'(z) = \begin{cases} 0, & z<0 \\ 1, & z>0 \end{cases}$$

Any Question?



Group Project Update

小组项目-调研综述进展汇报

序号	题目	成员
1	AI+斗地主	孙永康、李怀武、胡鸿飞、吴一凡、杨光、张习之、金肇轩（组长）、于佳宁
2	AI+五子棋	周贤玮、韩梓辰（组长）、赵云龙、张坤龙、夏星晨
3	High Score Gamer	易辰朗、许天淇、黄北辰（组长）、赵思源、朱佳伟、宛清源
4	AI application on diabetes	周钰奇、李仪轩、董叔文、湛掌、胡钧淇（组长）、裴鸿婧
5	AI in lung cancer	夏瑞浩、李悦明、龚颖璇、吴云潇潇（组长）、姜欣瑜、王英豪
6	基于MRI图像的阿尔茨海默症分类	董廷臻、郑英炜（组长）、李博翱、朱嘉楠、李杨燊
7	AI Applications in Breast Cancer Imaging	林文心、翟靖蕾（组长）、孙瀛、林宝月、陈帅名、冀鹏宇
8	Applications of artificial intelligence in covid-19 patients	罗岁岁（组长）、周雅雯、肖雨馨、程旸、尹子宜
9	基于OCT图像的眼部多种疾病诊断和分析的调研	何忧、郭煜煊、朱寒旭、赵子璇（组长）、王子杰、张晓新
10	人工智能对白内障分级的算法综述	赵宇航、徐格蕾、陈星宇、祖博瀛、黄弋骞（组长）
11	句子图片的文本情感分	唐云龙、刘叶充、刘旭坤、马卓远、陈子蔚（组

序号	题目	成员
12	gesture recognition	车文心、张静远、张骥霄（组长）、杜鹏辉
13	AI in Lab	孙含曦、于松琦、罗西（组长）、唐家豪、孙杰欣
14	人脸识别算法的发展与应用	易翔（组长）、陈俊滔、罗景南、胡泰玮、文颖潼、吴杰翰
15	人工智能在无障碍设施领域中的使用调查	马子晗（组长）、陈沐尧、林小璐、任艺伟、王增义
16	identification of handwriting elements	刘通、谈思序、赵伯航、张皓淇
17	AI虚拟主播制作计划	王标、张倚凡（组长）、李康欣、何泽安、曾宇祺、 Zhang Kenneth
18	人工智能技术在个性化推荐系统上的应用与研究	谭雅静、刘思岑、Ooi Yee Jing、孟宇阳、杨锦涛（组长）
19	校园巴士路线优化	王祥辰、何鸿杰、吴子彧、樊青远（组长）、方琪涵、袁通
20	给线稿上色的强大AI的算法研究	韩晗（组长）、刘思语、赵晓蕾、陈松斌
21	人工智能应用于病理分析的前景与挑战	刘宇欣、李修治（组长）、沈睿琦
22	深度学习在自动驾驶中的应用	王晓轩

Group 1

Fighting the Landlord With AI

- Basic design
  - Implement the frame and basic game flow of the game·········FINISHED
  - Implement the rules of playing cards in detail····ONGOING-->FINISHED
  - Design the graphical user interfaces·············PLANNING-->ONGOING
- Adding AI
  - Learn basic AI and algorithm······························ONGOING
  - Set up the AI structure···································ONGOING
  - Train AI··PLANNING

- Made a new plan, put more weight on the report

Group 3： Report on Introduction to Artificial Intelligence

Chess game demo

AND

The survey

```python
class MCTS:
    def __init__(self, network):
        self.Q_sa = {} #Q(s,a) RESERVED
        self.N = {}  #N(s)
        self.N_sa = {}  #N(s,a)
        self.P_s = {}   #P(s)
        self.moves = {}
        self.status = {} #status for state s, -1, 1 or 0
        self.network = network

    def get_pi(self, board, temp = 1):
        res = np.zeros((8,8))
        s = board.encode_board()

        N = self.N.get(s, 0)

        if temp == True:
            for x in range(8):
                for y in range(8):
                    res[x][y] = self.N_sa.get((s,x,y), 0) / N

            return res

        else: #temp == 0
            max_N = 0
            for x in range(8):
                for y in range(8):
                    res[x][y] = self.N_sa.get((s,x,y), 0)
                    max_N = max(max_N, res[x][y])

            if max_N > 0:
                res[res < max_N] = 0
                res /= np.sum(res)
            return res

    def get_next_move(self, board, temp):
        s = board.encode_board()

        if temp == True:
            N = self.N[s]
            #print("N is : ",N)

            possible_moves = self.moves.get(s, board.get_possible_moves())
            possibilities = np.zeros(len(possible_moves[0]), dtype=np.float)

            for i,(x,y) in enumerate(zip(possible_moves[0], possible_moves[1])):
                possibilities[i] = self.N_sa.get((s,x,y), 0) / N
```

```latex
\subsection{AlphaZero on Gobang}
  \subsection{Model design}
  Othello is a typical chess game, where two players take term to play a chess, in this case, the game can be represented in a game tree.
  The game tree is a tree structure where each node is the state of the board.
  and is linked to its child nodes, which are the valid moves that can be made at this state. All the leaf nodes in the game tree is an end state
  of the game.
  In the monte-carlo tree search, the program search through the game tree of Othello game, and tries to get the best move at a giving state.
  If the root node has a depth of 1,
  then all the levels with odd depth is played by the current player, all the levels with even depth is played by the opponent.
  Since the opponent is also a rational player, at all the levels with odd depth, we must choose action from the opponent's point of view.
  The neural network is simplified version of MobileNetV2\cite{sandler_mobilenetv2_2018}, a convolutional neural network optimized for mobile computing. The input is a single $1\times1\times8\times8$
  tensor representing the board and the output is a $8\times8$ matrix $p$ representing the policy and a single value $v$ representing the value.
  \subsubsection{Neural Network Architecture}
   The table \ref{table:1} shows the architecture of convolutional block, and
   table \ref{table:2} and \ref{table:3} show the architecture of two heads, the value head and the policy head.
   \begin{table}
    \centering

     \begin{tabular}{l|l|l}
      \noalign{\global\arrayrulewidth1.5pt}
      \hline
      \noalign{\global\arrayrulewidth0.4pt}
       Layer & Kernels & Input Size\\ [0.5ex]
      \noalign{\global\arrayrulewidth1.5pt}
      \hline
      \noalign{\global\arrayrulewidth0.4pt}
       Conv & $(3,3)\times20$ & $1\times8\times8$ \\
      \hline
       Conv & $(1,1)\times96$ & $20\times8\times8$ \\
      \hline
       Conv dw & $(3,3)\times96$ & $96\times8\times8$ \\
      \hline
       Conv & $(1,1)\times20$ & $96\times8\times8$ \\
      \hline
```

Group 4

自动胰岛系统
非算法部分初稿完成。
进行算法+非算法的对接。

控病虚拟助手算法部分初稿大部分完成。

病症预期早筛
算法部分，暂且搁置集成学习的预测，先集中突破回归预测。
非算法部分重心从临床转移到现有的APP预测。

Group 6

第6组（AD分类）第12周进度汇报

- 实现了基于VAE的分类器，等待数据重新预处理完毕后开始调试
- 即将完成数据的重新预处理，为最终测试做准备
- 开始为presentation准备PPT

后续计划：

- 以统一标准进行实验并收集数据
- 完成PPT
- 安排综述报告的分工

Group 7

AI Applications in Breast Cancer Imaging

AI in detection & diagnosis: CNN

一：介绍乳腺癌的严峻形势和AI的运用前景
二：与传统方法相比，AI可以怎样更好更快的基于影像进行乳腺癌的诊断与治疗
三：AI应用的具体例子/算法简述
四：展望与预测未来AI在该领域的发展

Treatment: Watson for oncology (Watson肿瘤解决方案，IBM公司开发)

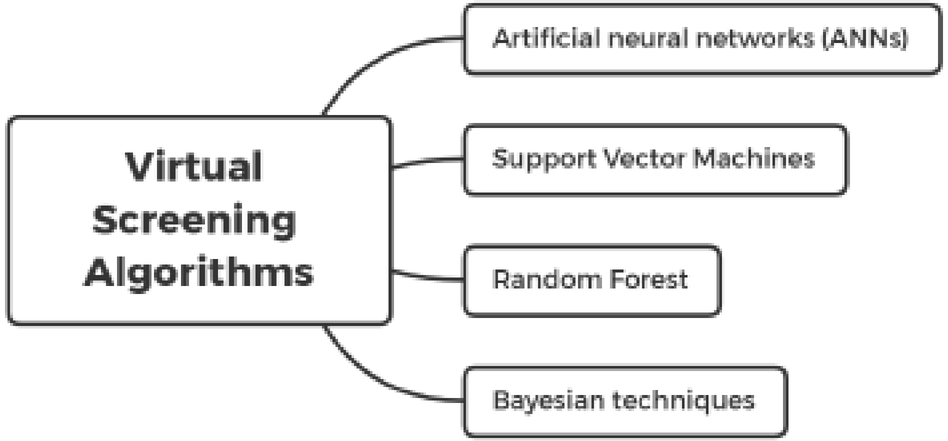
- 一项最为成熟的人工智能决策系统；
- 融合了**自然语言处理**和**机器学习**等领域的创新性技术；
- 已经学习了**大量**的学术论文和研究数据；
- 具有理解、推理、学习、互动功能；
- 通过读入大规模结构化和非结构化数据，实现**个体化分析病例**并给出治疗方案。

Prognosis: CNN

- 检测和量化癌症基因组图谱中肿瘤浸润淋巴结 (TILs) 的结构。

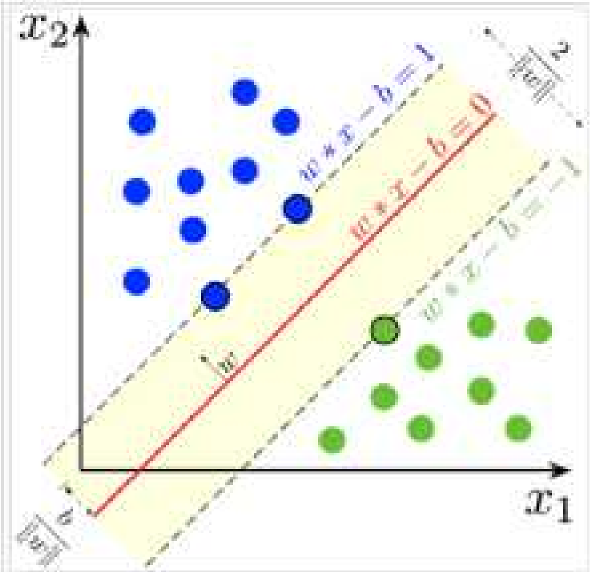
Group 8

Machine learning-based virtual screening algorithms in drug discovery
第八组



Virtual Screening Algorithms	Group Member
Introduction and discussion	周雅雯
Artificial neural networks (ANNs)	罗岁岁
Support Vector Machines	程旸
Random Forest	尹子宜
Bayesian techniques	肖雨馨

- Support vector machines (SVMs) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.
- SVMs are able to perform both linear and non-linear classification.

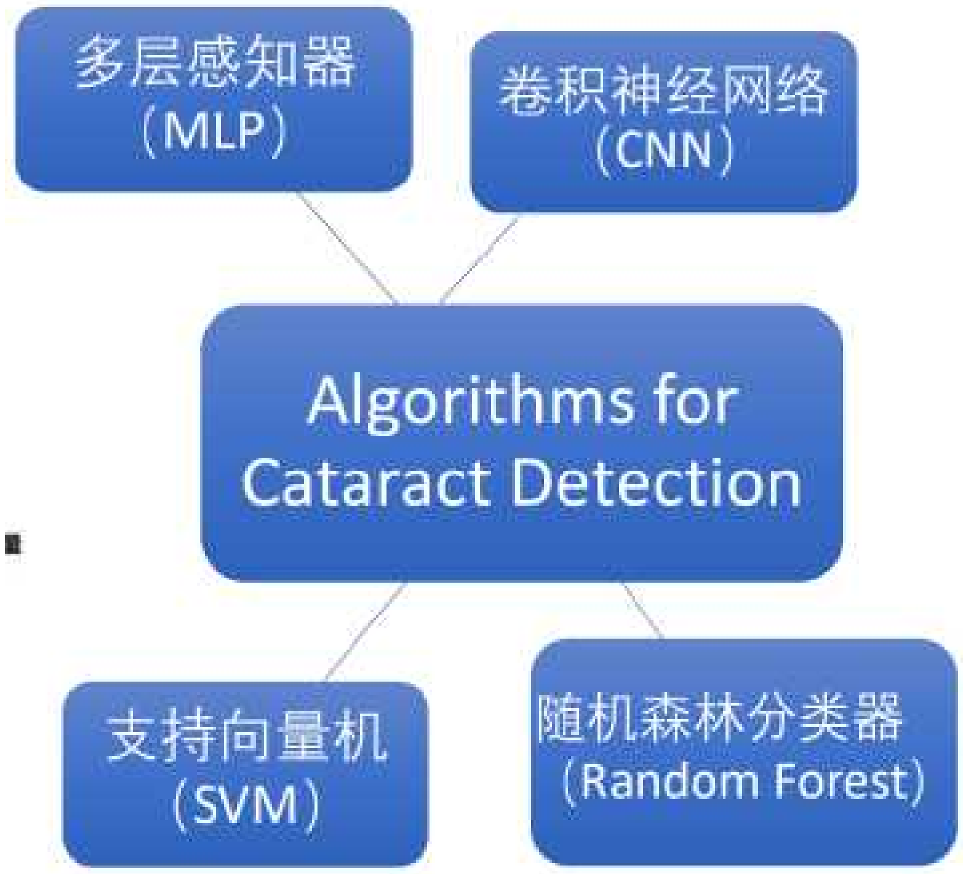


Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

Group 10

本周进展：

- 深入研读查阅的论文，了解并明确它们使用的算法
- 正在编写初稿，预计本周末能完成论文，下周进行论文的修改及校正



Group 13

Done this week:

1.clarify the definition of AI
2.integrate opinions to one conclusion
3.discuss about the AlphaFold

Thinkings：

1.Team project

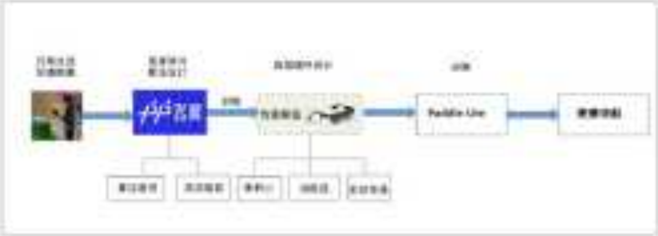
2.Our old thinking of AI

3……

Group 15

目前进展：

论文写作进展过半
关于有关算法有了一定的理解
两类产品的大概分析已基本完成



NavBelt 是一款基于高级移动机器人避障技术的计算机设备，一开始应用于机器人行走避障，后经过修改后能够帮助盲人在行走时躲避障碍。NavBelt 弥补了传统导盲杖需要使用者主动探索环境的缺点，成为了近年来被广泛使用的电子行走辅助工具(ETA)之一。

Guidecane 是一款基于机器人避障系统制作而成的类似导盲犬的设备。与其他设备相比，该设备有更易于操作，高可读性的用户接口。

Group 16：Handwriting identification

Write a Project Plan Describing Intended Algorithms and Application You Want to Research for Your Project and Project Milestones.

Week 5-6: python learning

Weak 7-11: basic algorism and math knowledge learning, and realization

Week 12-13: essay editing and modifying codes.

第十二周记录

- 已完成初稿，已完成实现，目前修改论文+优化

- 发展背景，zbh
- 流程 tsx
- 算法1 刘通
- 算法2 张hq
- ddl 11.29

Group 17： AI+Language

小组成员11911109张倚凡 11910216王标 11911307李康欣
11913022Ken 12011323何泽安 12011811曾宇祺

Report

Introduction 曾宇祺
Preprocess 张倚凡
1.分词
(1)现状
(2)算法实现
a.传统方法 基于词典 字典树
b.基于双向BiLstm神经网络的中文分词
(3)工具库实现 jieba
(4)未来研究
2.去停用词/关键词提取
(1)停用词集
a. Method 1
(2)去词
a. Method 2
b. Method 3

NLG 王标 李康欣 Ken
语料库
Conclusion 何泽安

Group 18：个性化推荐系统

小组成员 杨锦涛 刘思岑 钟悦芸 谭雅静 孟宇阳 Ooi Yee Jing

综述完成进度:

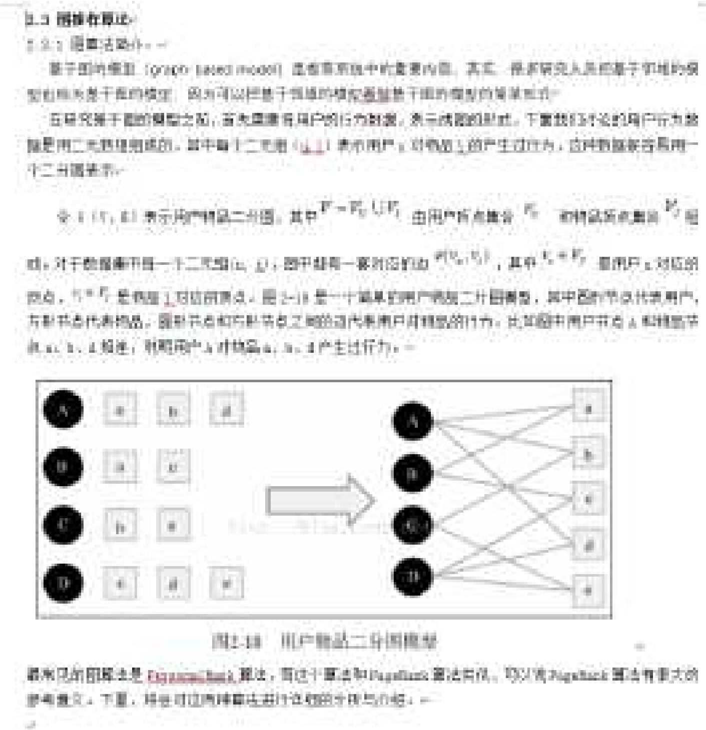
个性化推荐系统的背景和发展历程----已完成

基于内容的推荐算法----已完成

基于协同过滤的推荐算法----已完成

基于图网络的推荐算法----已完成

基于知识图谱的推荐算法----已完成



Group 19: 校园巴士路线优化

小组成员： 王祥辰 、 何鸿杰 、 吴子彧 、 樊青远 、 方琪涵、袁通

- 本周进度
  - 实际操作部分
    - 完成通过基础PLR调整权重得到的到站时间预测
    - 制作了一个通过命令行，可根据位置查看下一班车到站时间

- 论文综述部分
  - 查找不同的公交到站时间预测与规划方法
  - 开始撰写不同预测方式（统计,PLR,CNN,RNN等方式预测校园巴士到站时间）
  - 开始撰写实际操作部分的数据处理流程和神经网络优化权重的流程

- 进度预计
  - 11月30日前完成文献收集，论文框架搭建
  - 12月6日前，完成实际操作部分
  - 12月20日前，完成论文

Group21： AI and Skin Caner

小组成员：李修治 沈睿琦 刘宇欣

➢上周工作进展：

对论文的结构进行梳理

结论：以皮肤癌诊断工具为代表的AI工具应用于临床的前景广阔，但仍需要留出足够的时间给技术发展，以及回答技术之外的问题。

➢本周工作重点：

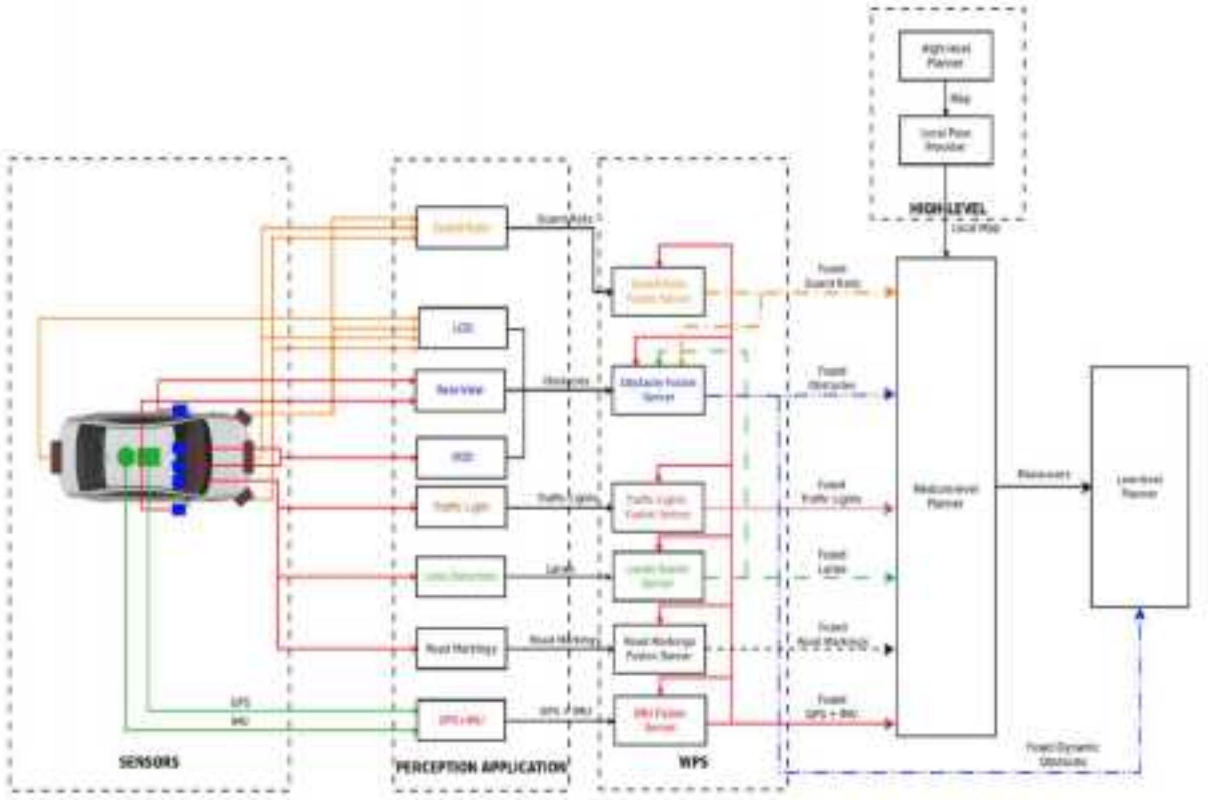
小组汇报PPT

通过其他小组的调研报汇报查漏补缺，修改论文。

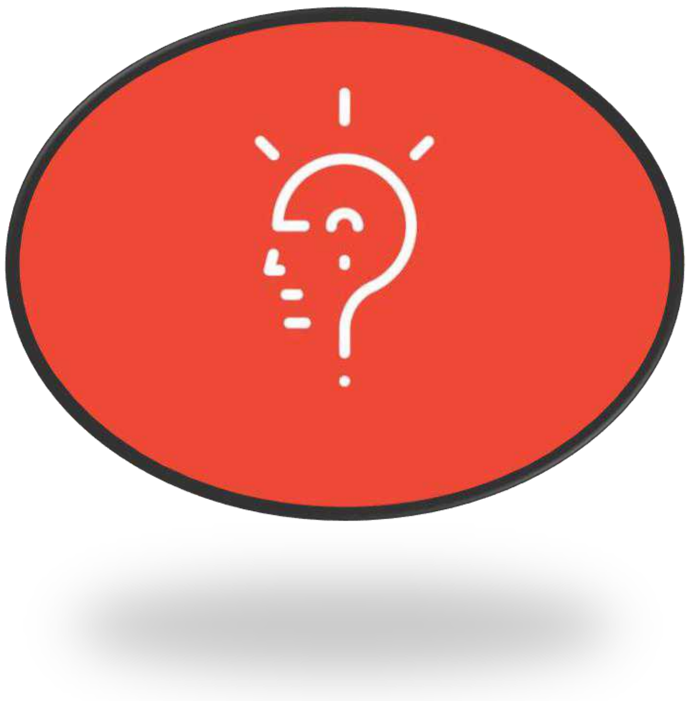
Group 22：深度学习在自动驾驶中的应用综述

小组成员：11911633王晓轩

- 调研正式结束
- 报告内容基本确定，草稿已经写完
- 报告结构还在调整中
- 主要内容：
- 研究背景
- 深度学习模型介绍
- 深度学习在在自动驾驶中的应用现状分析
- 自动驾驶技术三大模型（重点调研端到端模型）



Any Question?



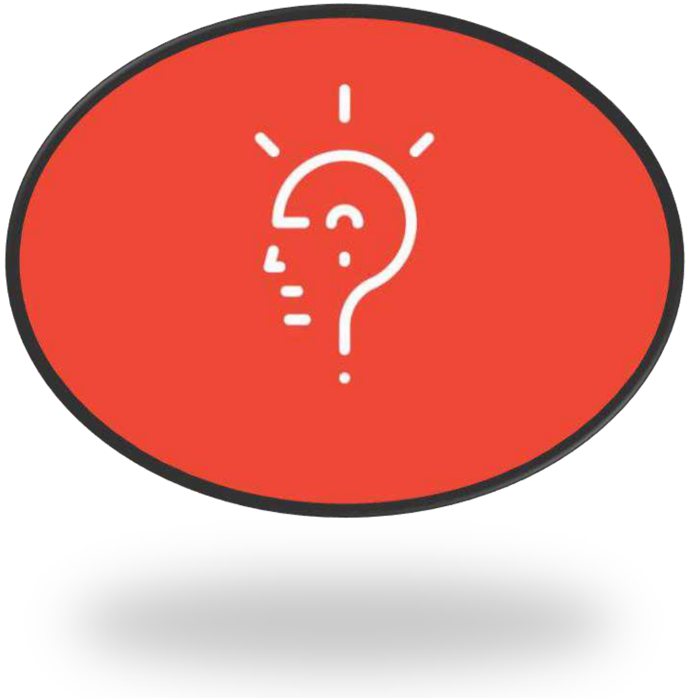
课程项目汇报时间安排

小组编号	时间	主题	成员
2	十三周（上午）	AI+五子棋	周贤玮、**韩梓辰（组长）**、赵云龙、张坤龙、夏星晨
3	十三周（上午）	High Score Gamer	易辰朗、许天淇、**黄北辰（组长）**、赵思源、朱佳伟、宛清源
4	十三周（上午）	AI application on diabetes	周钰奇、李仪轩、董叔文、湛掌、**胡钧淇（组长）**、裴鸿婧
5	十三周（上午）	AI in lung cancer	夏瑞浩、李悦明、龚颖璇、**吴云潇潇（组长）**、姜欣瑜、王英豪
10	十三周（上午）	人工智能对白内障分级的算法综述	赵宇航、徐格蕾、陈星宇、祖博瀛、**黄弋骞（组长）**
13	十四周（上午）	AI in Lab	孙含曦、于松琦、**罗西（组长）**
14	十四周（上午）	人脸识别算法的发展与应用	**易翔（组长）**、陈俊滔、罗景南、胡泰玮、文颖潼、吴杰翰
17	十四周（上午）	AI虚拟主播制作计划	王标、**张倚凡（组长）**、李康欣、何泽安、曾宇祺、Zhang Kenneth
19	十四周（上午）	校园巴士路线优化	王祥辰、何鸿杰、吴子彧、**樊青远（组长）**、方琪涵、袁通
22	十四周（上午）	深度学习在自动驾驶中的应用	**王晓轩**

课程项目汇报时间安排

小组编号	时间	主题	成员
1	十三周（下午）	AI+斗地主	孙永康、李怀武、胡鸿飞、吴一凡、杨光、张习之、**金肇轩（组长）**、于佳宁
6	十三周（下午）	基于MRI图像的阿尔茨海默症分类	董廷臻、**郑英炜（组长）**、李博翱、朱嘉楠、李杨燊
7	十三周（下午）	AI Applications in Breast Cancer Imaging	林文心、**翟靖蕾（组长）**、孙瀛、林宝月、陈帅名、冀鹏宇
8	十三周（下午）	Applications of artificial intelligence in covid-19 patients	**罗岁岁（组长）**、周雅雯、肖雨馨、程旸、尹子宜
9	十三周（下午）	基于OCT图像的眼部多种疾病诊断和分析的调研	何忧、郭煜煊、朱寒旭、**赵子璇（组长）**、王子杰、张晓新
11	十三周（下午）	句子图片的文本情感分析	唐云龙、刘叶充、**刘旭坤（组长）**、马卓远、陈子蔚、江欣乐、陈浩然
12	十四周（上午）	gesture recognition	**车文心、张静远、张骥霄（组长）**、杜鹏辉
15	十四周（下午）	人工智能在无障碍设施领域中的使用调查	**马子晗（组长）**、陈沐尧、林小璐、任艺伟、王增义
16	十四周（下午）	identification of handwriting elements	刘通、**谈思序（组长）**、赵伯航、张皓淇
18	十四周（下午）	人工智能技术在个性化推荐系统上的应用与研究	谭雅静、刘思岑、Ooi Yee Jing、孟宇阳、**杨锦涛（组长）**
20	十四周（下午）	给线稿上色的强大AI的算法研究	**韩晗（组长）**、刘思语、赵晓蕾、陈松斌
21	十四周（下午）	人工智能在皮肤癌诊断领域的可能性探索	刘宇欣、**李修治（组长）**、沈睿琦

Any Question?



Machine Learning

3 5

1 Back Propagation

2 Supporter Vector Machine

3 Machine Learning

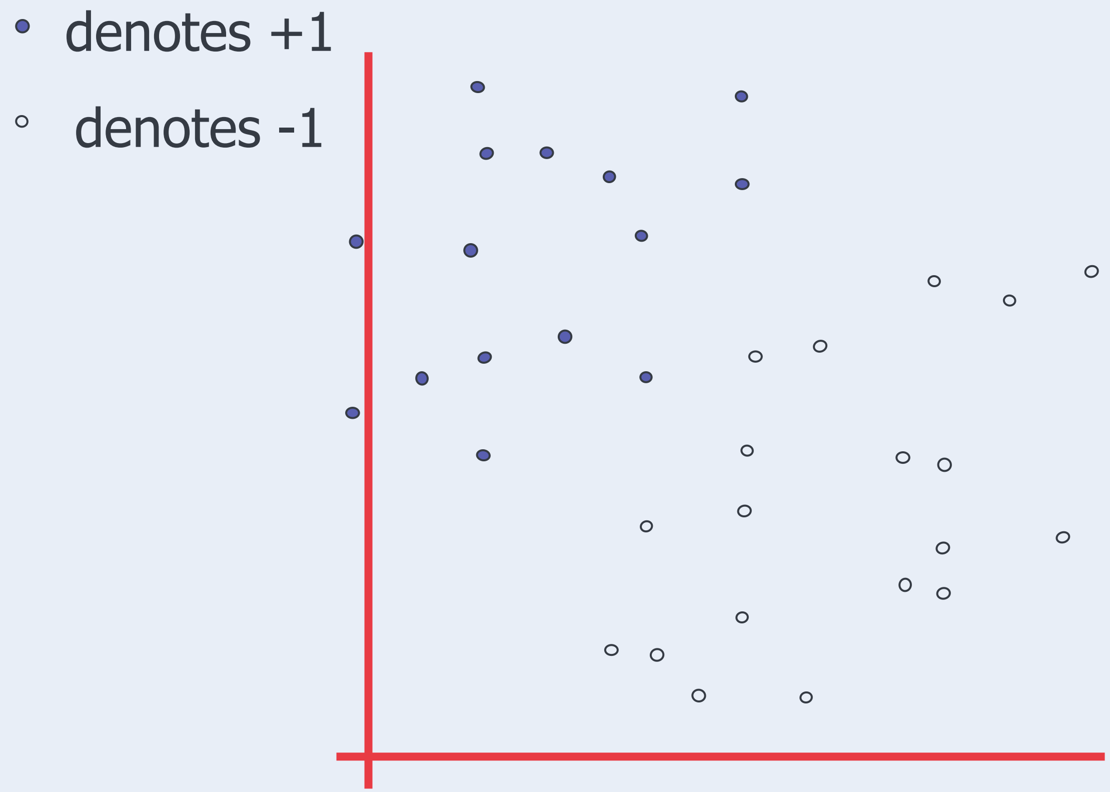
4 Knowledge

Support Vector Machine

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Developed at AT&T Bell Laboratories by Vapnik with colleagues (Boser et al., 1992, Guyon etal., 1993, Vapnik et al., 1997), it presents one of the most robust prediction methods, based on the statistical learning framework or VC theory proposed by Vapnik and Chervonenkis (1974) and Vapnik (1982, 1995). Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

Wikipedia

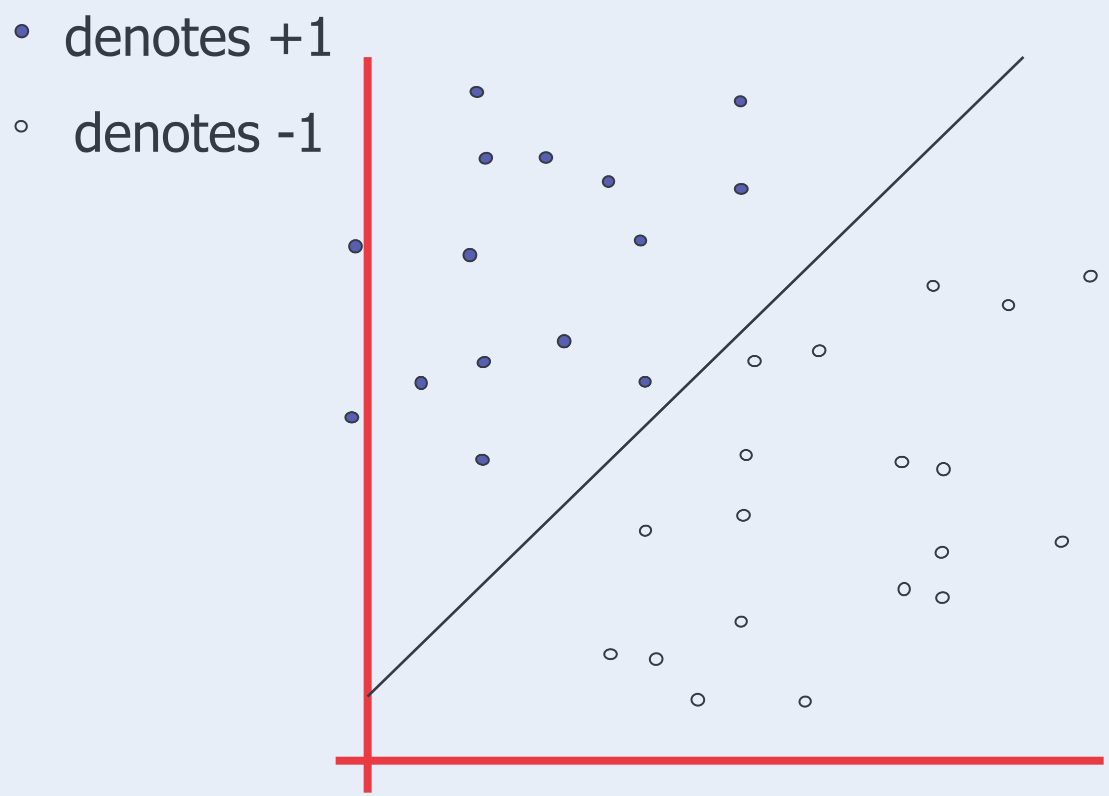
2 Class Classification: $f(x,w,b) = sign(w. x + b)$



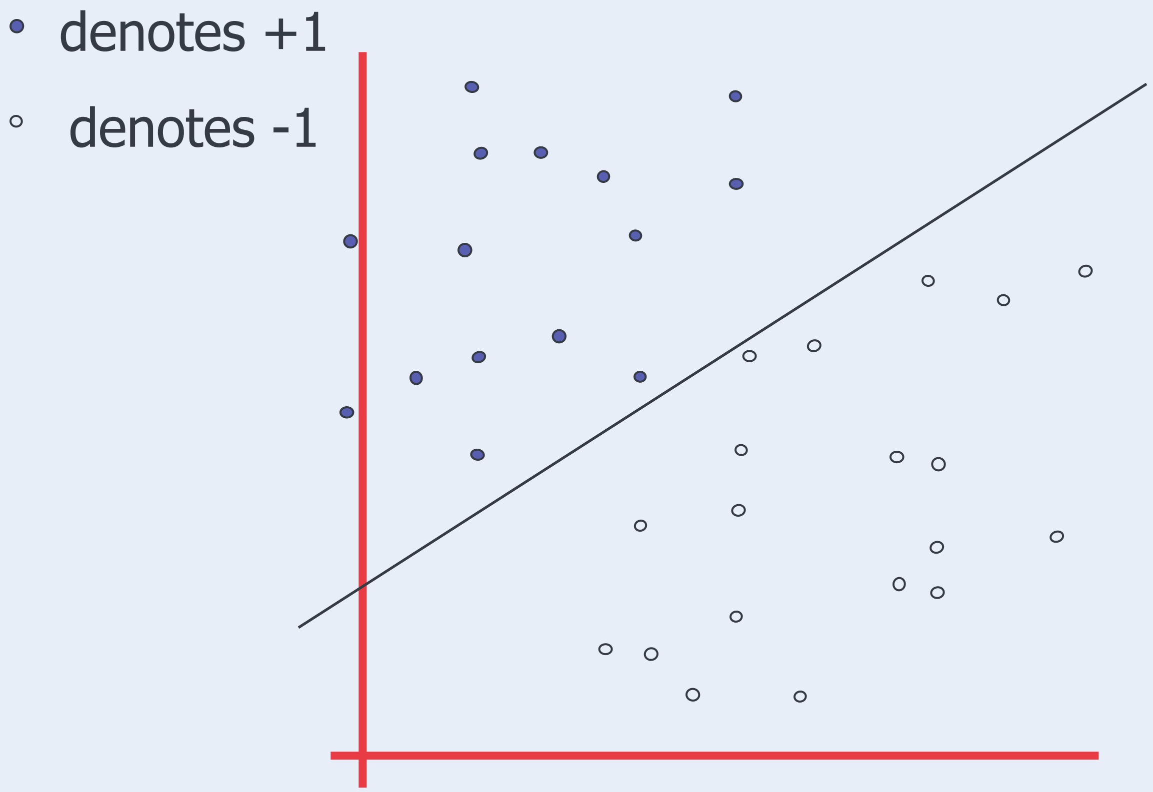
w: weight vector
x: data vector

How would you classify this data?

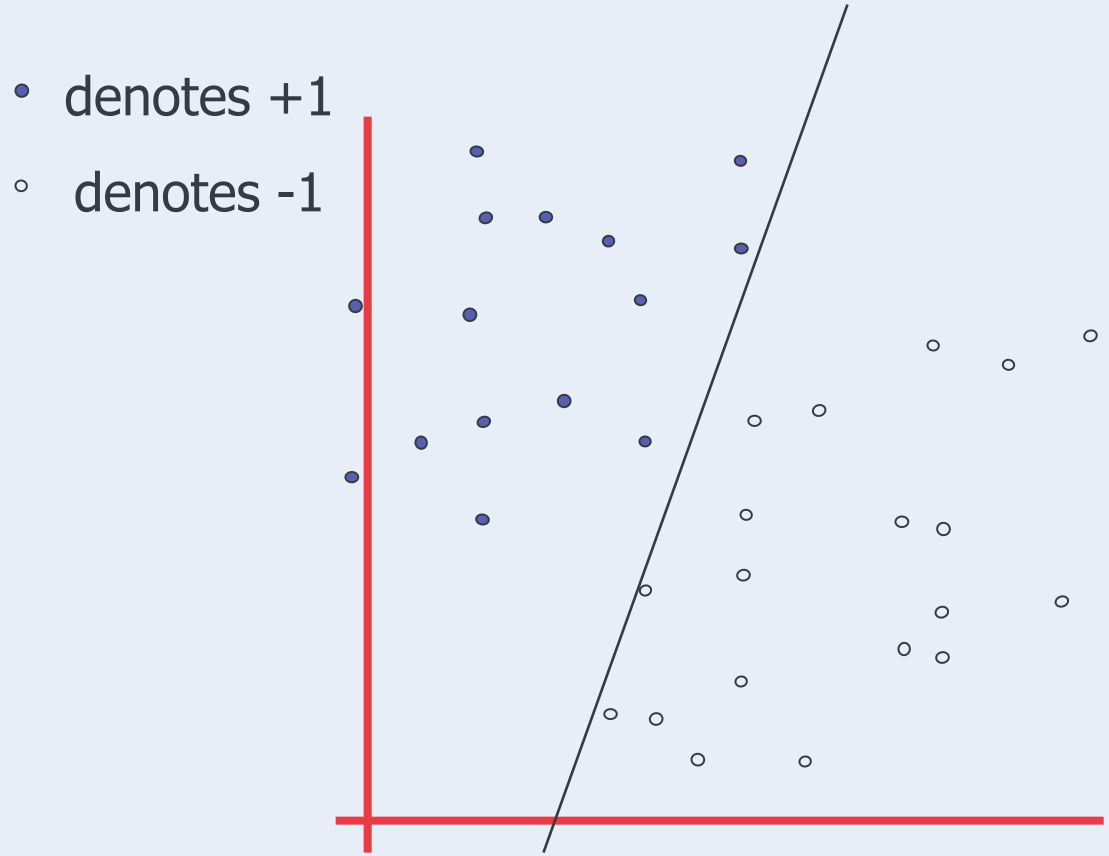
One Solution



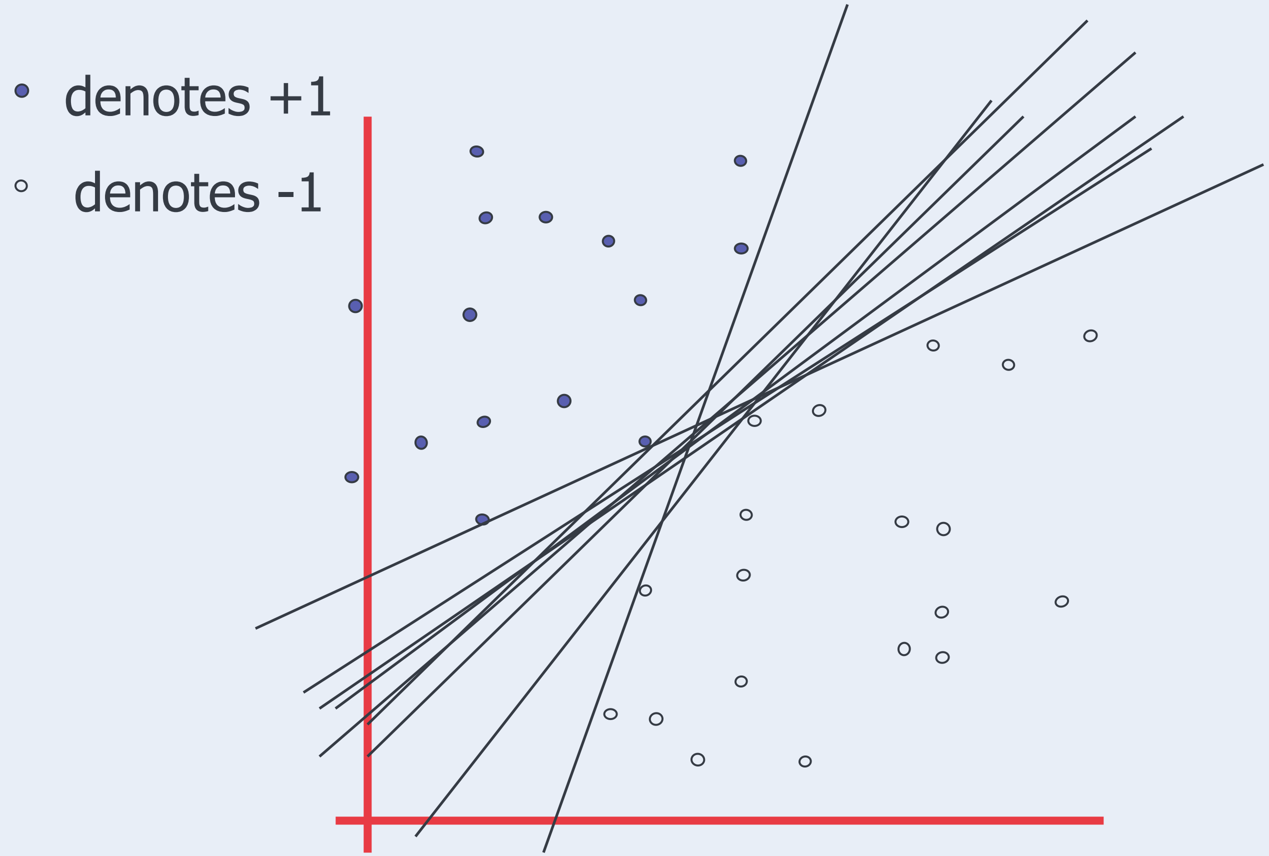
Another Solution



Third Solution

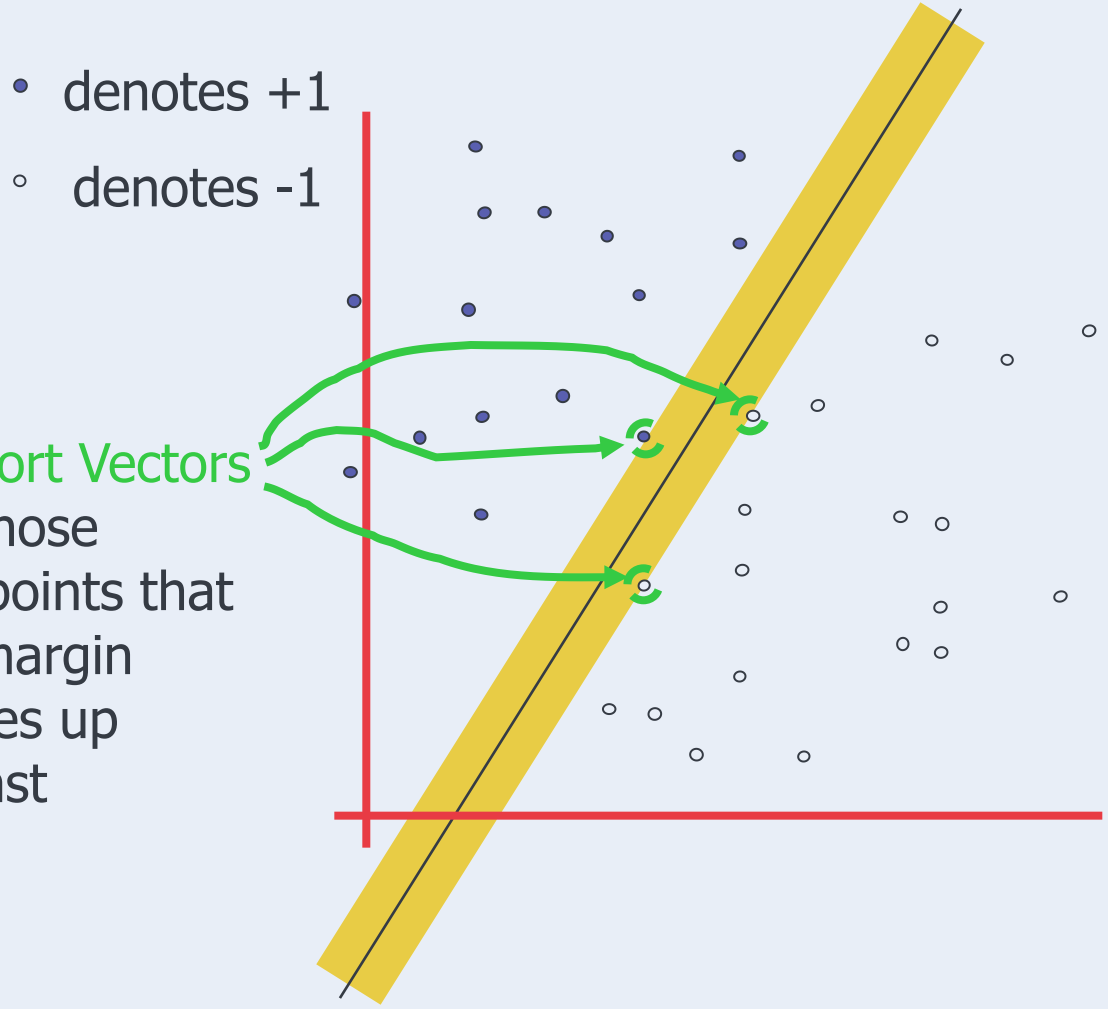


Many Solutions



$f(x,w,b) = \text{sign}(w. x + b)$

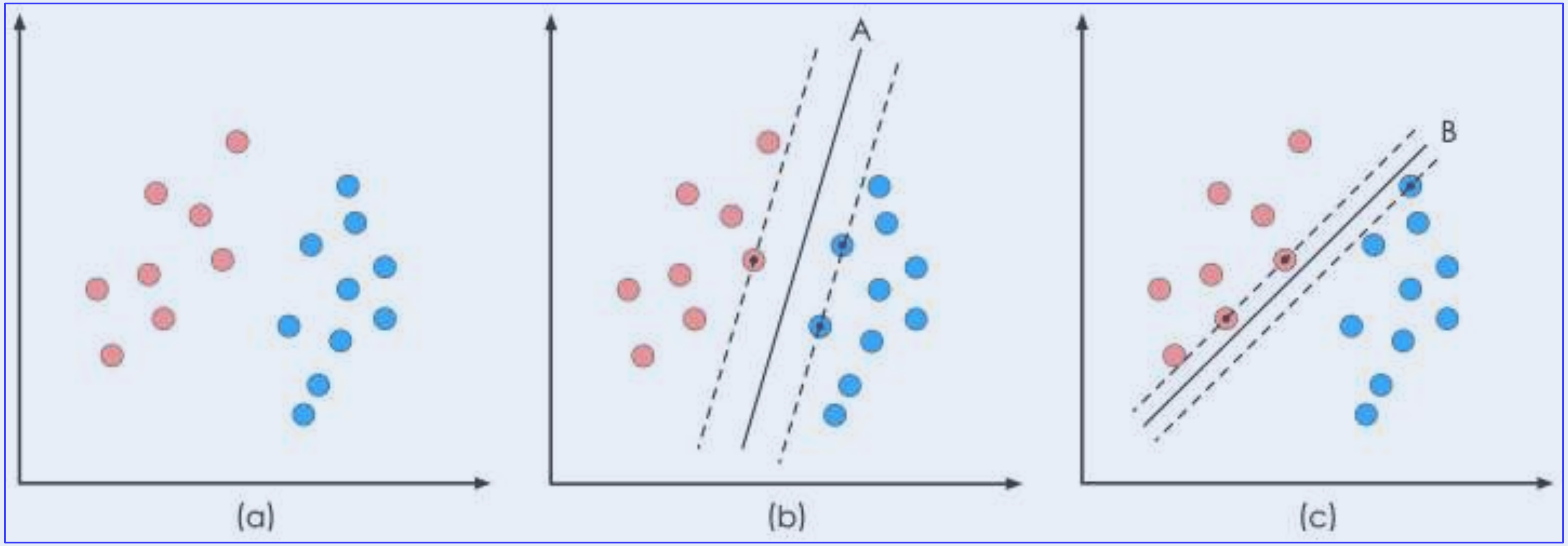
Support Vectors are those datapoints that the margin pushes up against



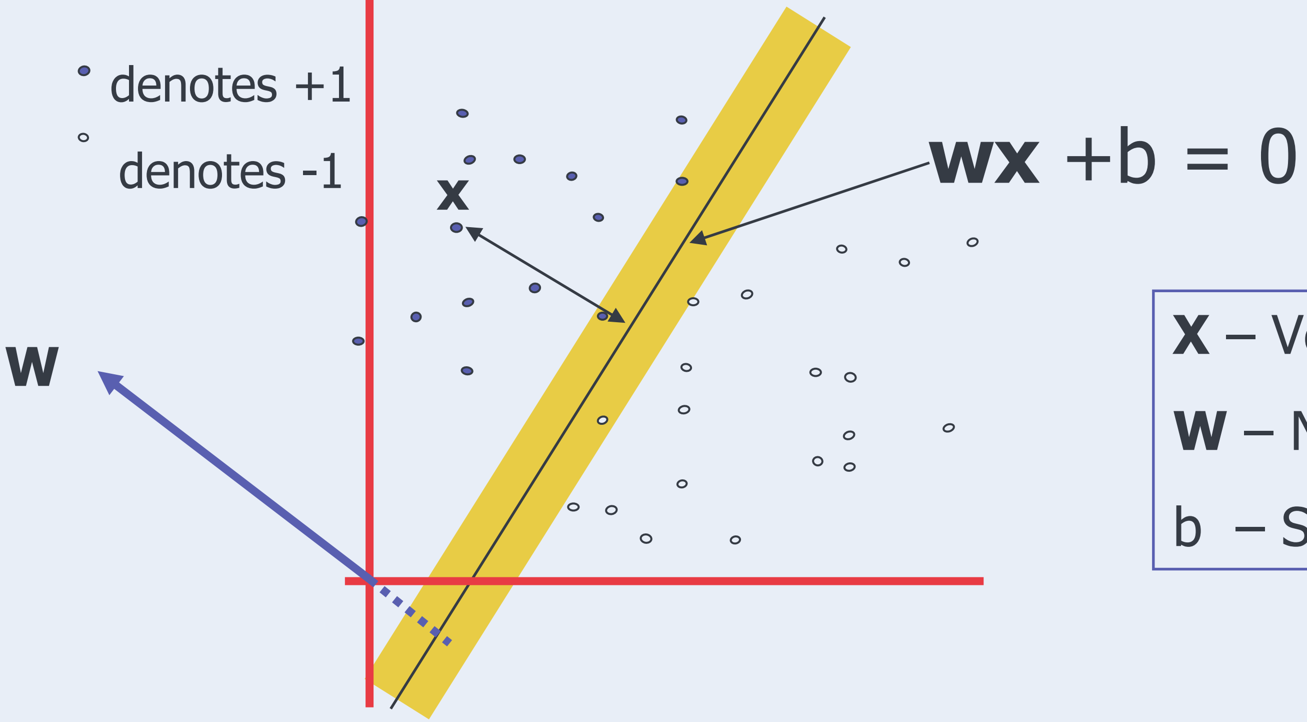
Margin of a linear classifier is the width that the boundary could be increased by before hitting a datapoint.

The maximum margin linear classifier is the Linear SVM (LSVM)

A or B Classifier?



Distance of Point to Line



X – Vector

W – Normal Vector

b – Scale Value

The distance for a point **x** to a line **wx**+b= 0 is:

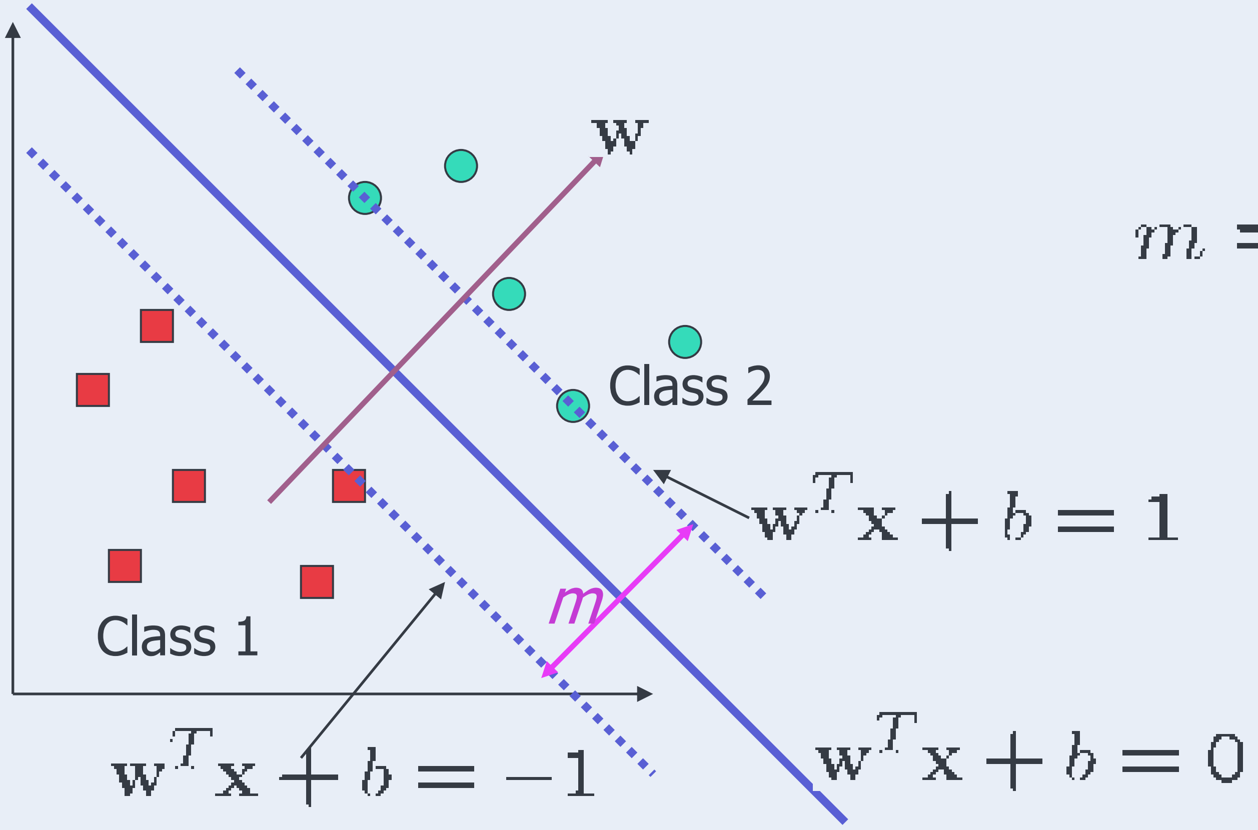
$$d(\mathbf{x}) = \frac{\left|\mathbf{x}\cdot\mathbf{w}+b\right|}{\sqrt{\|\mathbf{w}\|_2^2}} = \frac{\left|\mathbf{x}\cdot\mathbf{w}+b\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

Margin m

The decision boundary should be as far away from the data of both classes as possible

We should maximize the margin *m: smallest distance from observations to hyperplane*

Distance between the origin and the line $\mathbf{w}^t\mathbf{x}=-b$ is $b/||\mathbf{w}||$



$$m = \frac{2}{||\mathbf{w}||}$$

Solve SVM by Decision Boundary (Max Margin)

- Let $\{x_1, \ldots, x_n\}$ be our data set and let $y_i \in \{1,-1\}$ be the class label of x_i
- The decision boundary should classify all points correctly

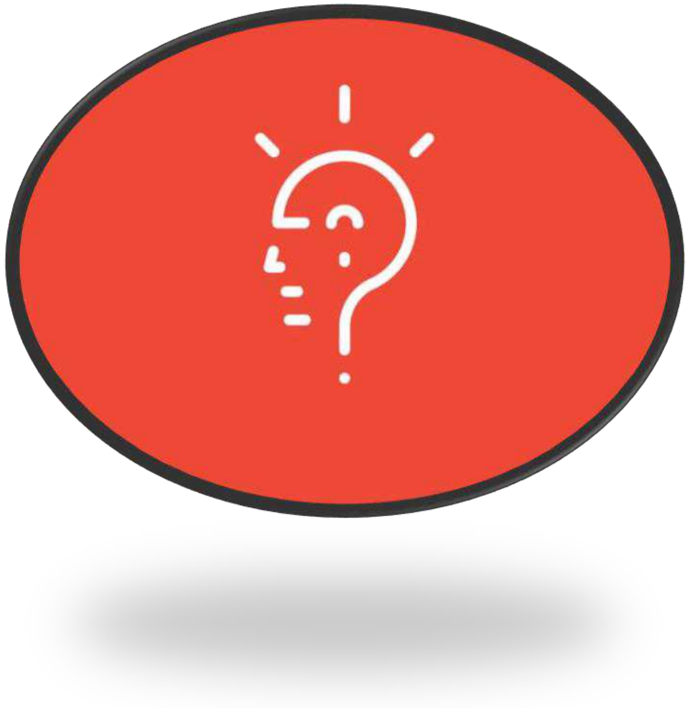
$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \qquad \forall i$$

- To see this: when y=-1, we wish (wx+b)<1, when y=1, we wish (wx+b)>1. For support vectors, we wish y(wx+b)=1.
- The decision boundary can be found by solving the following constrained optimization problem

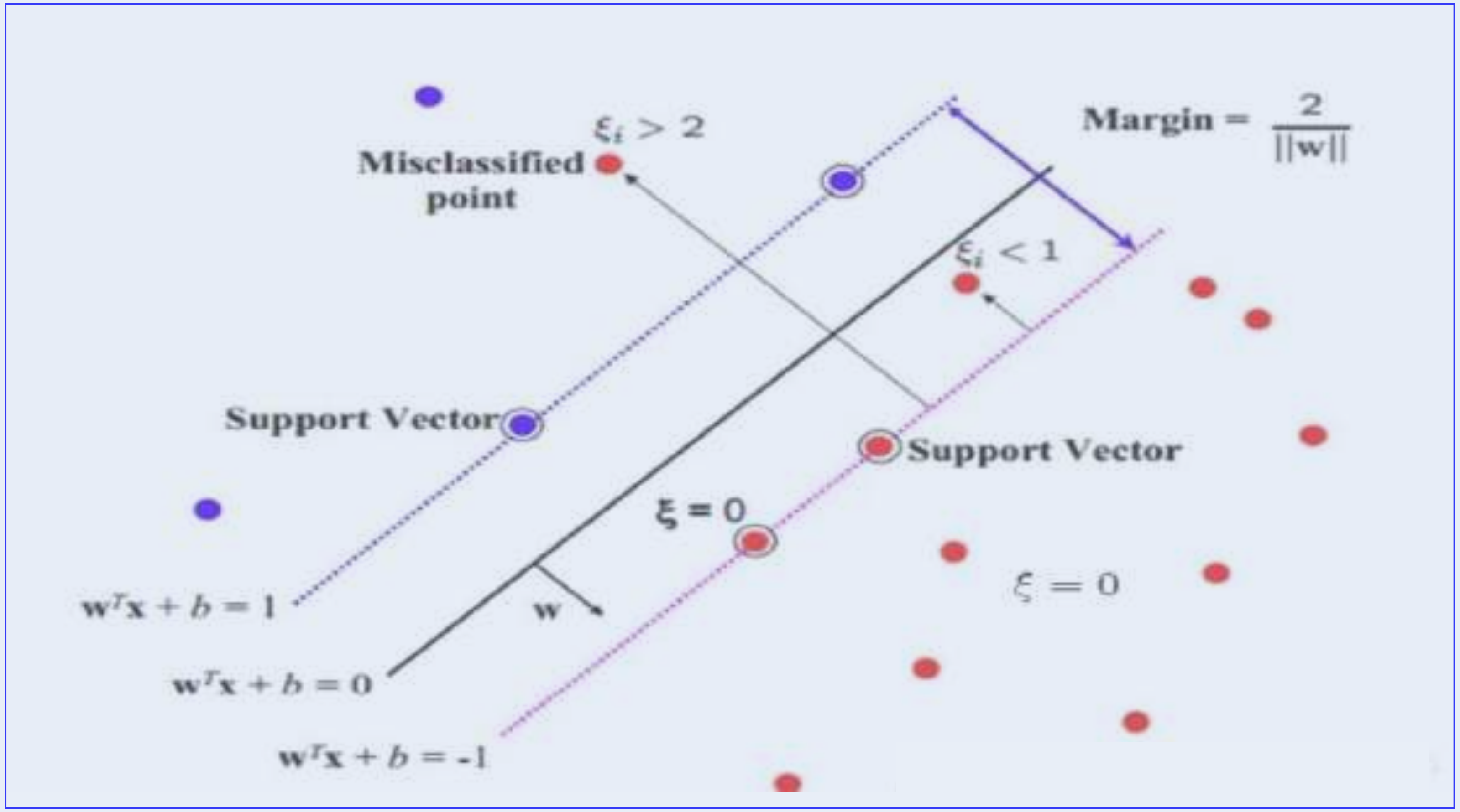
$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

Any Question?



SVM with Slacks for Red Class



Hinge Loss

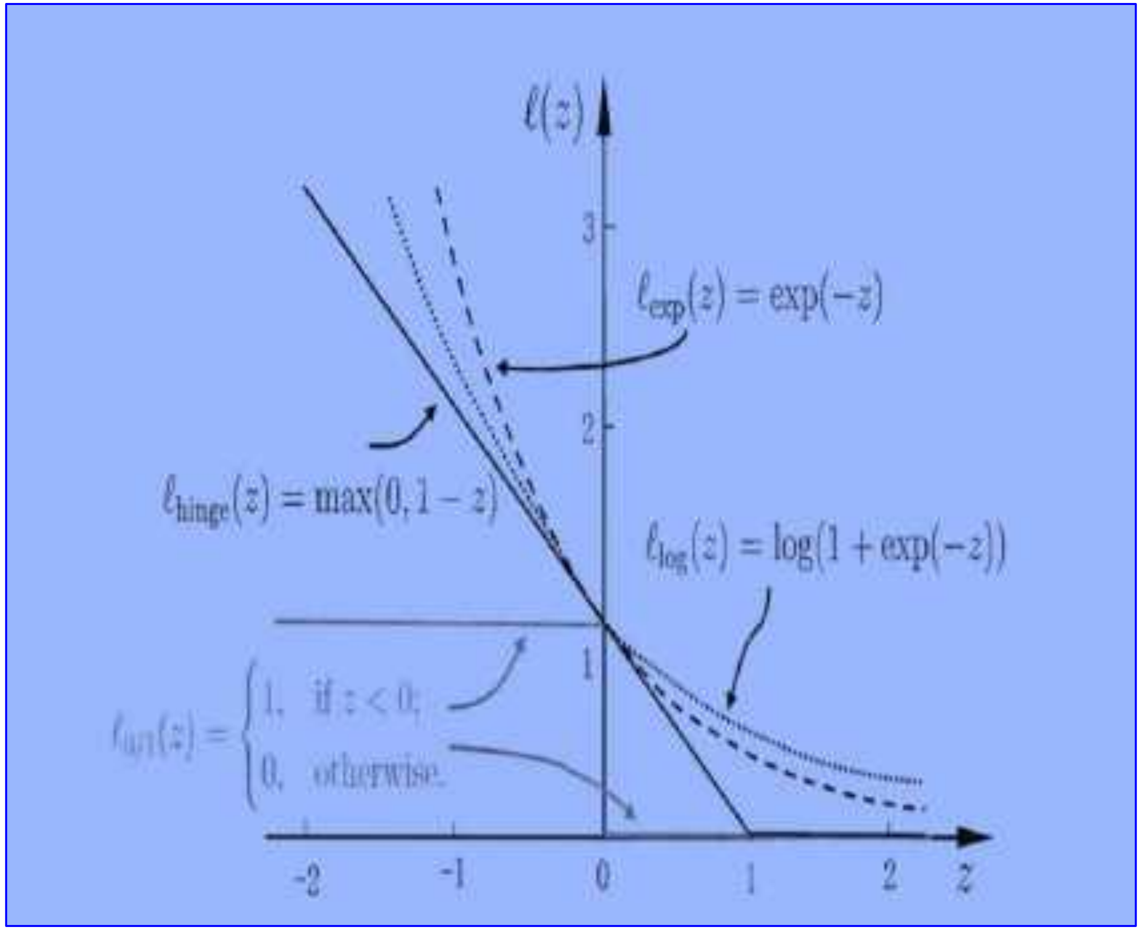
In machine learning, the hinge loss is a loss function used for training classifiers. The hinge loss is used for "maximum-margin" classification, most notably for support vector machines (SVMs). For an intended output $t = \pm 1$ and a classifier score y, the hinge loss of the prediction y is defined as

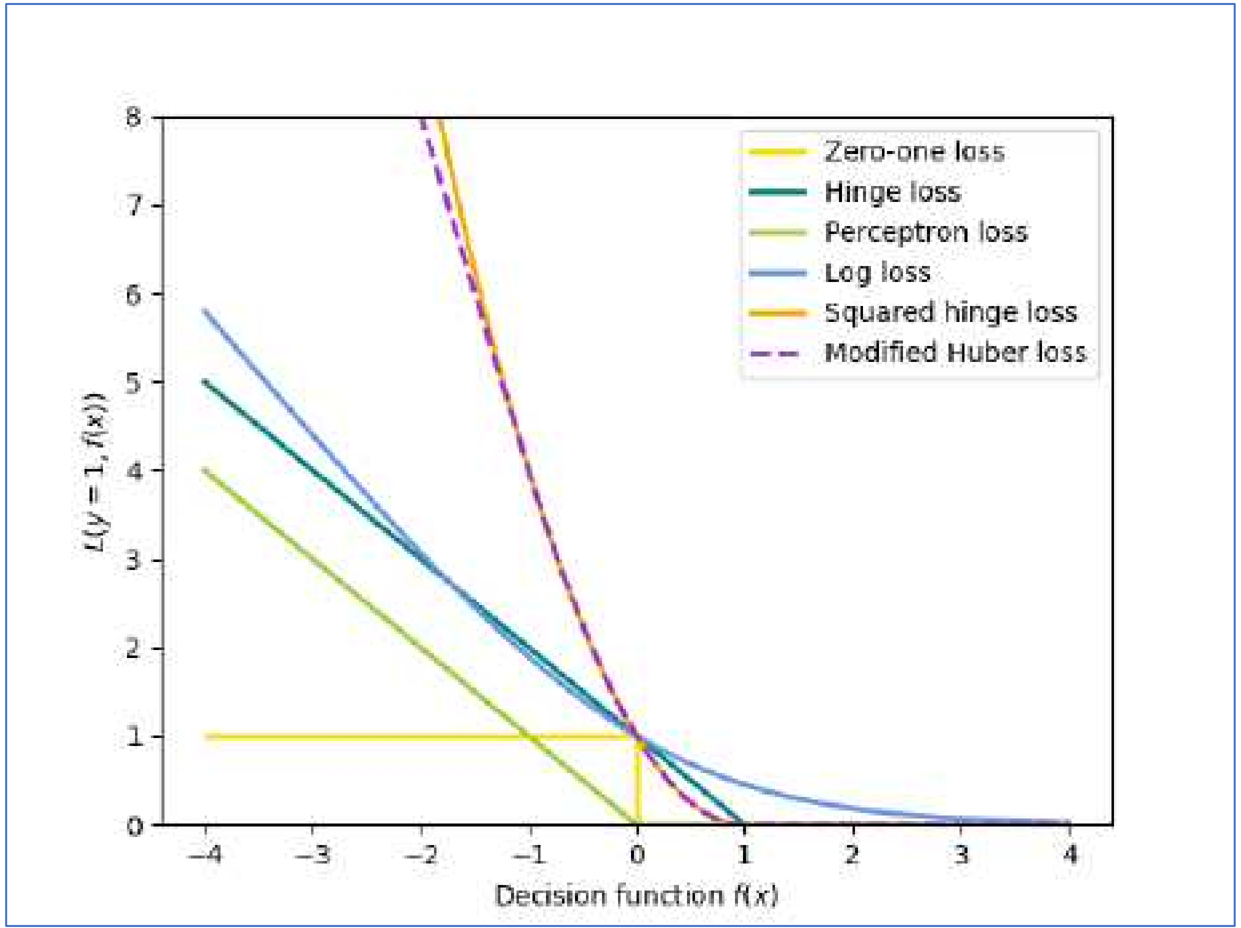
$$\ell(y)=\max(0,1-t\cdot y)$$

Note that y should be the "raw" output of the classifier's decision function, not the predicted class label.

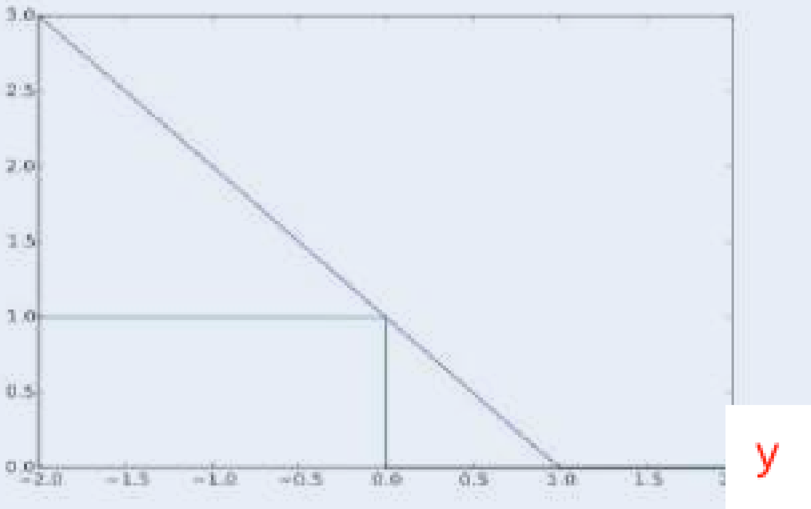
Wikipedia

Other Loss to surrogate non-continuous non-convex 0/1 loss

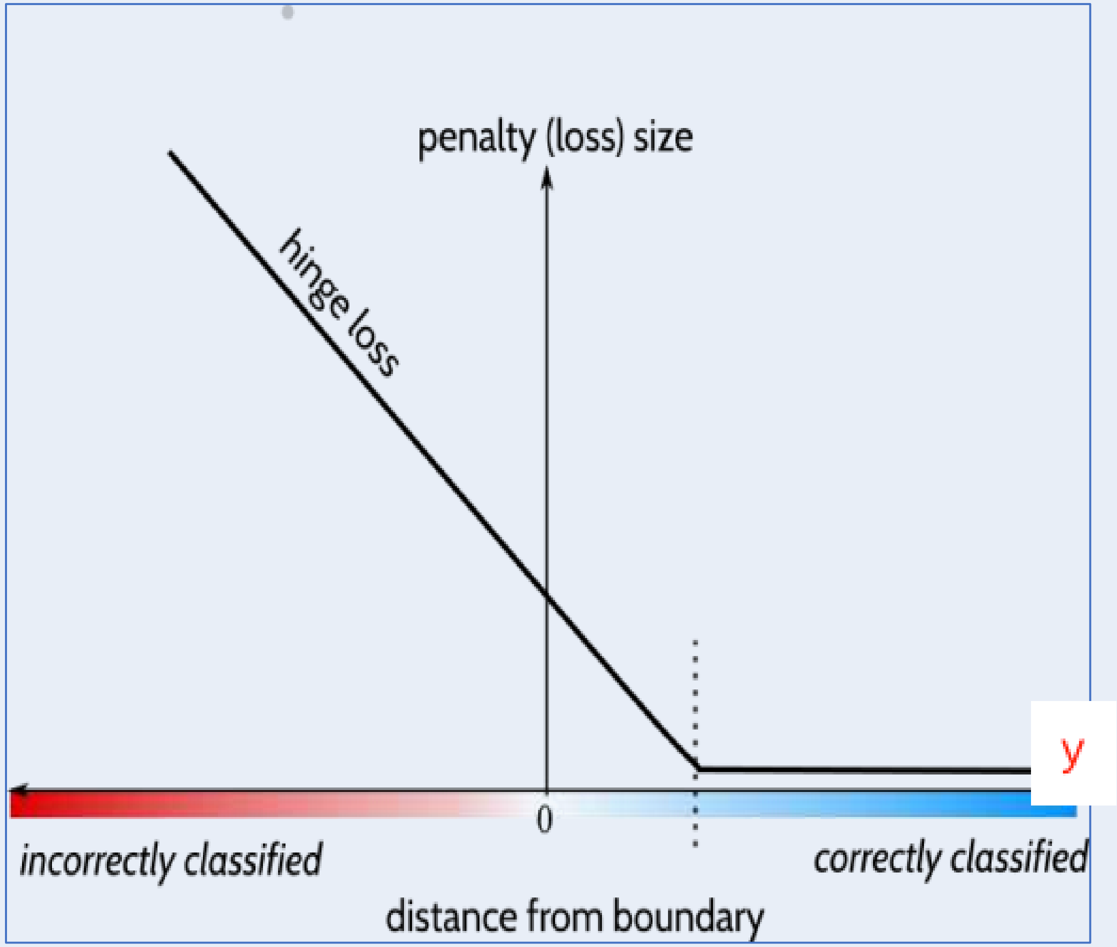




Maximize Margin = Minimize Hinge Loss (y>1, Loss =0)



Plot of hinge loss (blue, measured vertically) vs. zero-one loss (measured vertically; misclassification, green: $y < 0$) for $t = 1$ and variable y (measured horizontally). Note that the hinge loss penalizes predictions $y < 1$, corresponding to the notion of a margin in a support vector machine.



SVM Unique Solution

Optimization Problem

(Cortes and Vapnik, 1995)

Constrained optimization:

$$\min_{\mathbf{w},b,\xi} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i$$

subject to $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i \ \wedge \ \xi_i \geq 0, i \in [1, m].$

Properties:

- $C \geq 0$ trade-off parameter.
- Convex optimization.
- Unique solution.

Dual Formulation for Slack Variables

Primal formulation:

$$\text{Minimize } \underbrace{\boxed{\tfrac{1}{2}\sum_{i=1}^{n} w_i^2 + C\sum_{i=1}^{N}\xi_i}}_{\text{Objective function}} \text{ subject to } \underbrace{\boxed{y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1 - \xi_i}}_{\text{Constraints}} \text{ for } i = 1,\ldots,N$$

Dual formulation:

$$\text{Minimize } \underbrace{\boxed{\sum_{i=1}^{n}\alpha_i - \tfrac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j \vec{x}_i\cdot\vec{x}_j}}_{\text{Objective function}} \text{ subject to } \underbrace{\boxed{0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{N}\alpha_i y_i = 0}}_{\text{Constraints}}$$

for $i = 1,\ldots,N$.

Extend SVM with Slack Variables

- If we minimize $\sum_i \xi_i$, ξ_i can be computed by

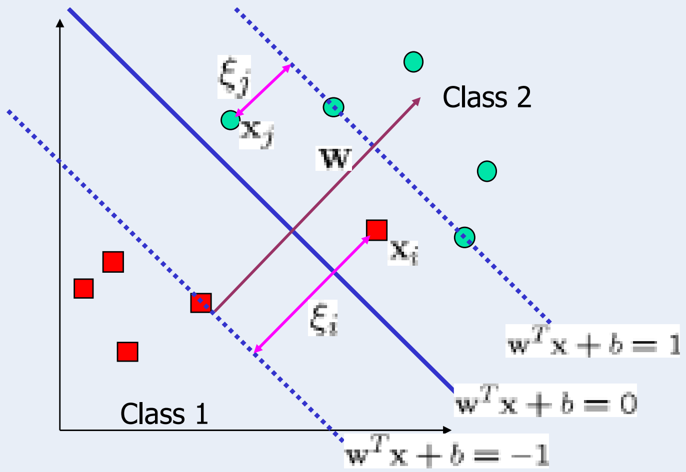
$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

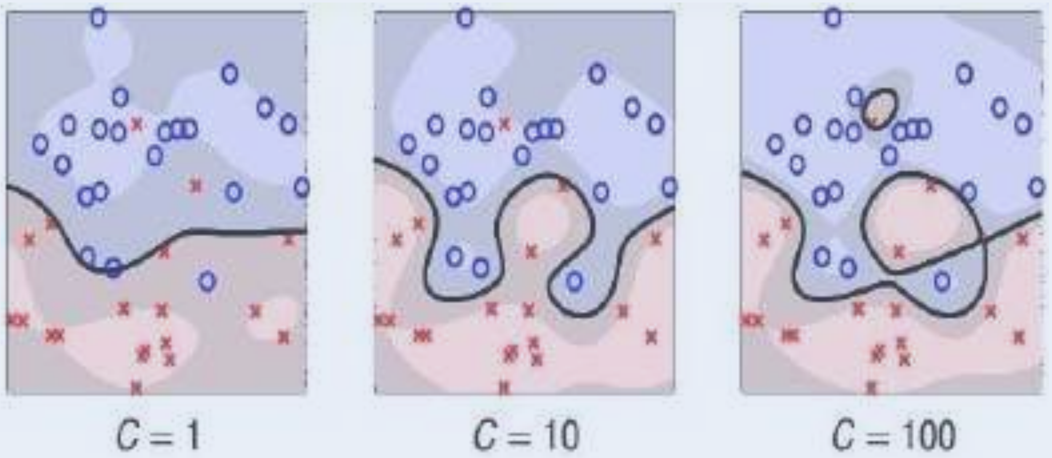
  - ξ_i are "slack variables" in optimization, for fixed w and b, they are the hinge loss
  - Note that ξ_i=0 if there is no error for $\mathbf{x}_i$
  - ξ_i is an upper bound of the number of errors
- We want to minimize $\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^n \xi_i$
  - *C* : tradeoff parameter between error and margin
- The optimization problem becomes

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^n \xi_i$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

SVM with Soft Margins

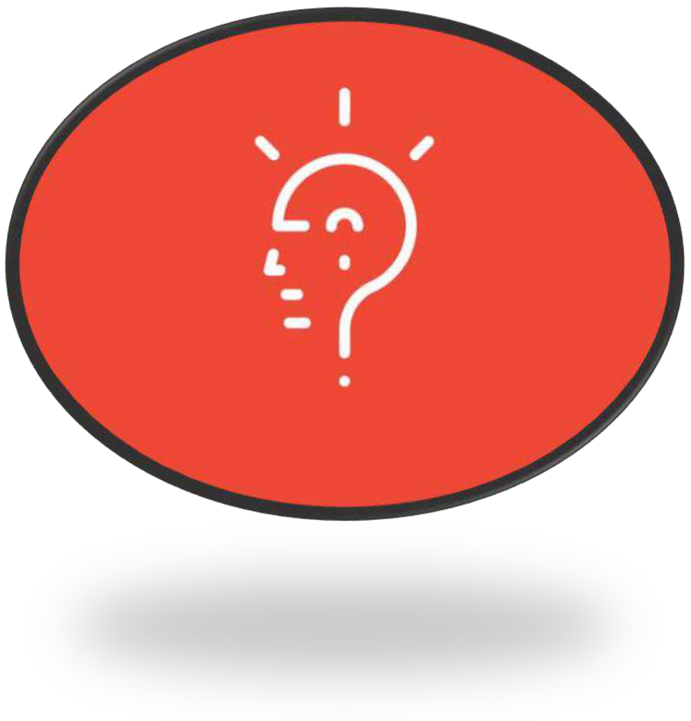




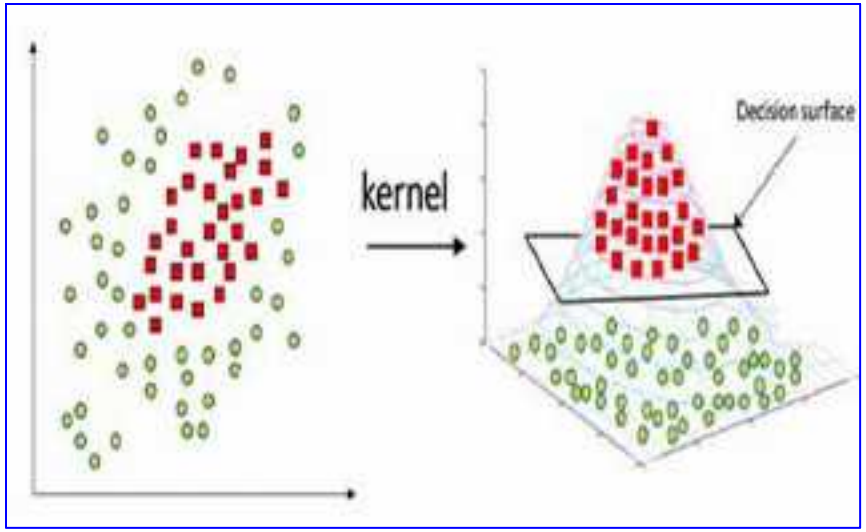
soft-margin SVM: $\min_{b,\mathbf{w},\xi} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\cdot\sum_{n=1}^{N}\xi_n$

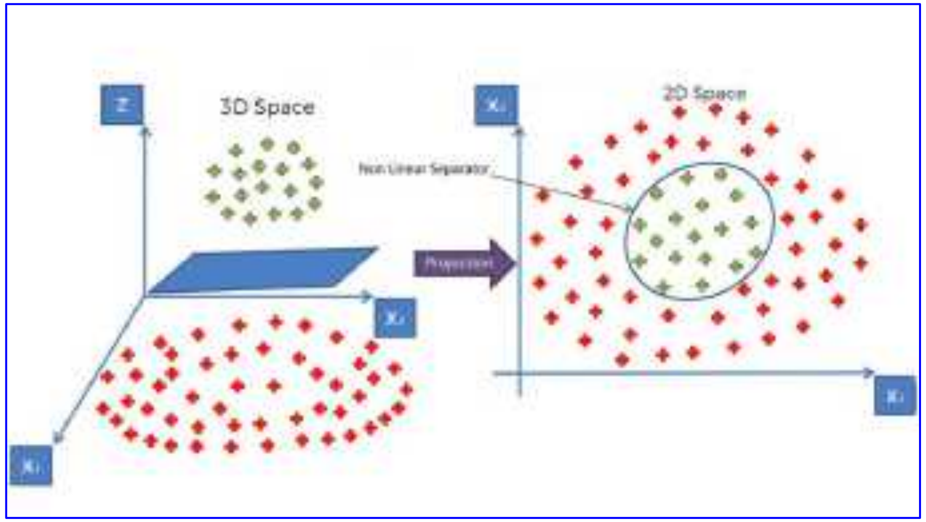
s.t. $\quad y_n(\mathbf{w}^T\mathbf{z}_n + b) \geq 1 - \xi_n$ and $\xi_n \geq 0$ for all n

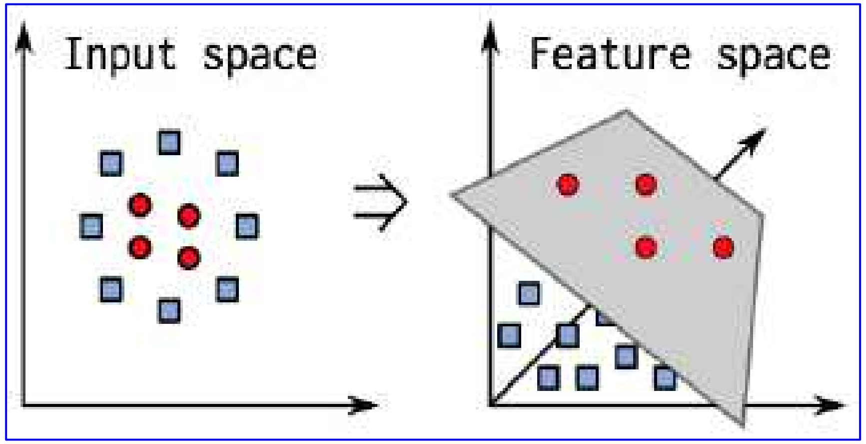
Any Question?

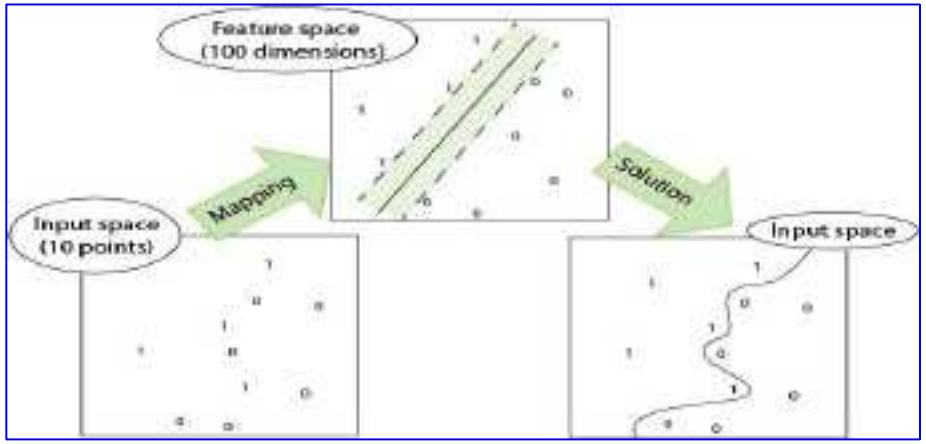


Non-Linear SVM Classifier with Kernel Mapping





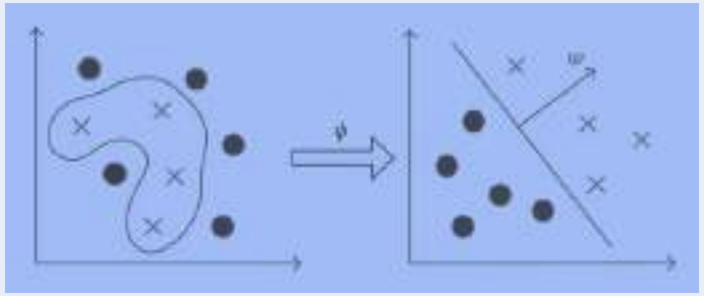




What is Kernel in SVM

Kernel is a way of computing the dot product of two vectors $\mathbf{x}$ and $\mathbf{y}$ in some (possibly very high dimensional) feature space, which is why kernel functions are sometimes called "generalized dot product".

Suppose we have a mapping $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}^m$ that brings our vectors in $\mathbb{R}^n$ to some feature space $\mathbb{R}^m$. Then the dot product of $\mathbf{x}$ and $\mathbf{y}$ in this space is $\varphi(\mathbf{x})^T \varphi(\mathbf{y})$. A kernel is a function k that corresponds to this dot product, i.e. $k(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x})^T \varphi(\mathbf{y})$.



Kernel Trick

Kernel Trick is an approach consisting in the use of **kernel functions**, operating in a high-dimensional, implicit feature space without ever computing the coordinates of the data in that space, but rather by simply **computing the inner products between the images of all pairs of data in the feature space**.

Kernel Trick

Directly computing $K(x, z)$ can be faster than "feature transformation + inner product" sometimes.

$$
\begin{aligned}
K(x, z) &= (x \cdot z)^2 \\
&= (x_1 z_1 + x_2 z_2 + \cdots + x_k z_k)^2 \\
&= {x_1}^2 {z_1}^2 + {x_2}^2 {z_2}^2 + \cdots + {x_k}^2 {z_k}^2 \\
&+ 2 x_1 x_2 z_1 z_2 + 2 x_1 x_3 z_1 z_3 + \cdots \\
&+ 2 x_2 x_3 z_2 z_3 + 2 x_2 x_4 z_2 z_4 + \cdots \\
&= \phi(x) \cdot \phi(z)
\end{aligned}
$$

$$
x = \begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix} \quad z = \begin{bmatrix} z_1 \\ \vdots \\ z_k \end{bmatrix}
$$

$$
\phi(x) = \begin{bmatrix} {x_1}^2 \\ \vdots \\ {x_k}^2 \\ \sqrt{2} x_1 x_2 \\ \sqrt{2} x_1 x_3 \\ \vdots \\ \sqrt{2} x_2 x_3 \\ \vdots \end{bmatrix}
$$

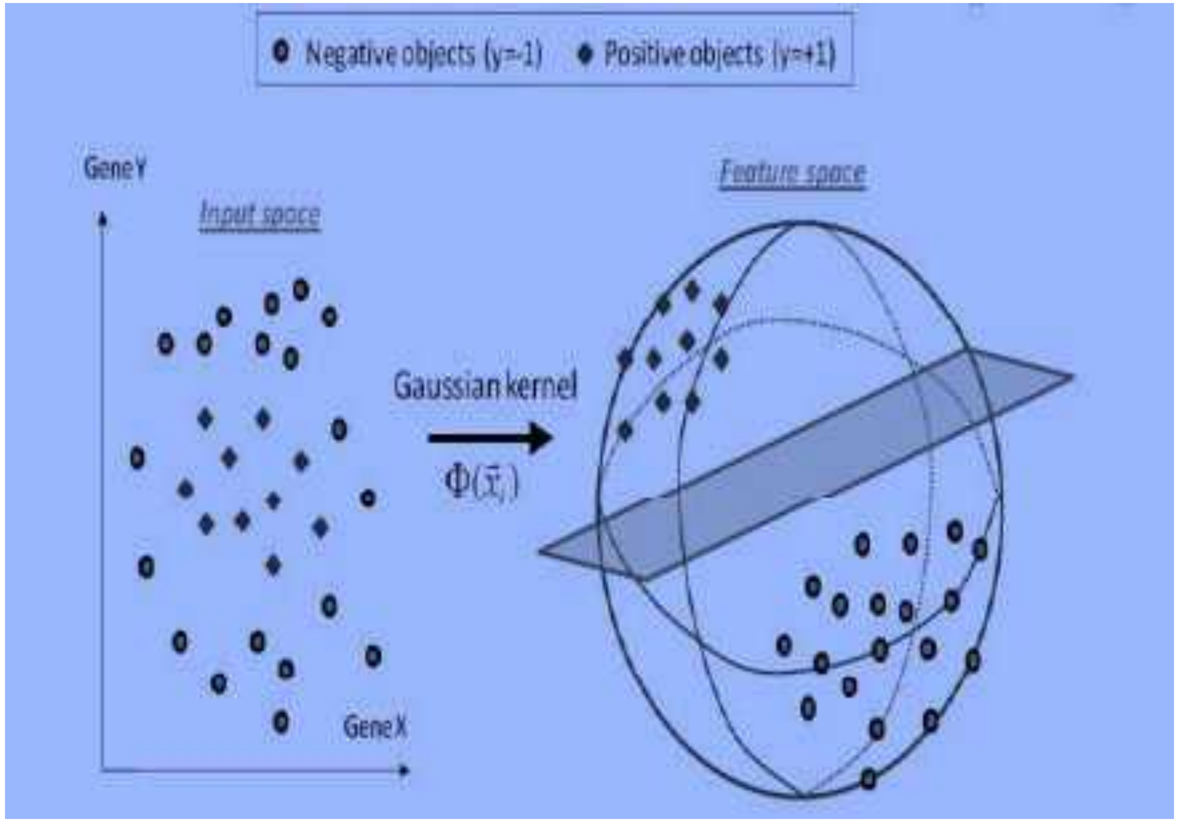
Other Common SVM Kernels and RBF Kernel Trick

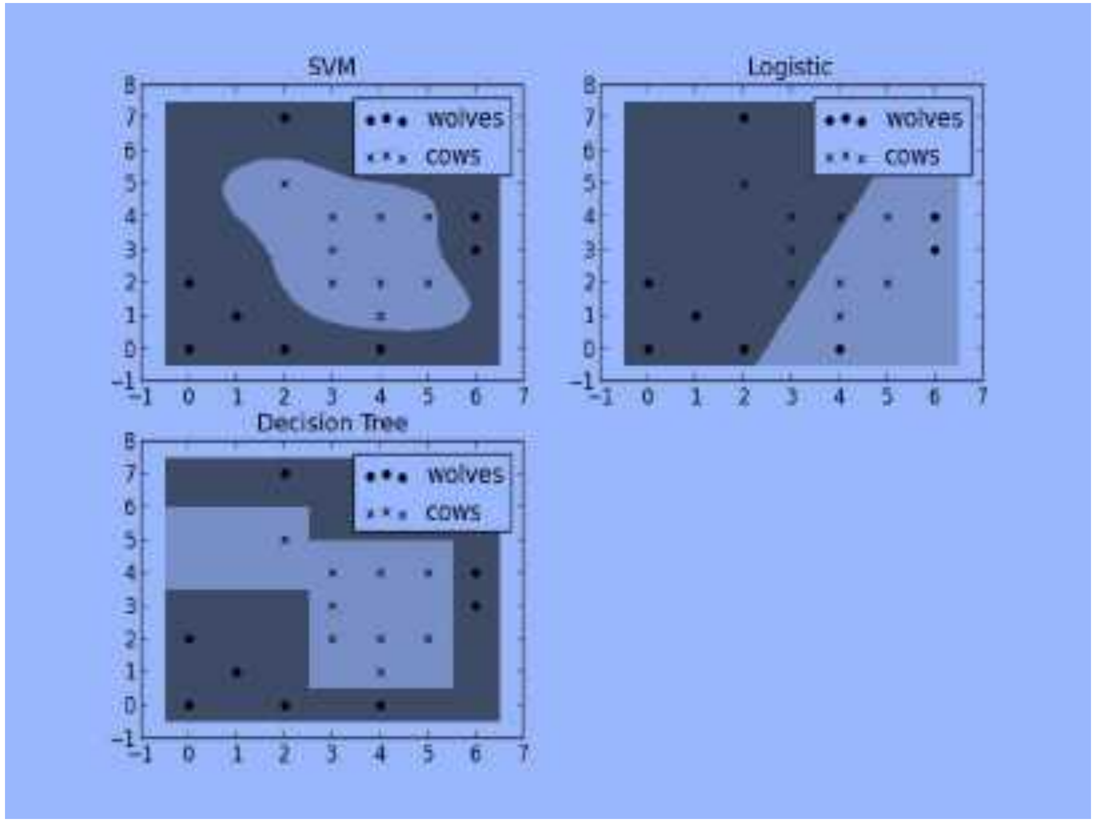
Kernel name	Kernel function
Linear kernel	$K(x, y) = x \times y$
Polynomial kernel	$K(x, y) = (x \times y + 1)^d$
RBF kernel	$K(x, y) = e^{-\gamma\|x-y\|^2}$

$$
\begin{aligned}
e^{\frac{-1}{2\sigma^2}(x_i - x_j)^2} &= e^{\frac{-x_i^2 - x_j^2}{2\sigma^2}} \left(1 + \frac{2x_i x_j}{1!} + \frac{(2x_i x_j)^2}{2!} + \dots\right) \\
&= e^{\frac{-x_i^2 - x_j^2}{2\sigma^2}} \left(1 \cdot 1 + \sqrt{\frac{2}{1!}} x_i \cdot \sqrt{\frac{2}{1!}} x_j + \sqrt{\frac{(2)^2}{2!}} (x_i)^2 \cdot \sqrt{\frac{(2)^2}{2!}} (x_j)^2 + \dots\right) \\
&= \phi(x_i)^T \phi(x_j)
\end{aligned}
$$

$$
\text{where, } \phi(x) = e^{\frac{-x^2}{2\sigma^2}} \left(1, \sqrt{\frac{2}{1!}} x, \sqrt{\frac{2^2}{2!}} x^2, \dots\right)
$$

SVM Kernel and Classifiers





Additional Concepts: Lagrange, Duality, Kernel

$$\mathcal{L}(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\alpha_i\left(y_i(w^Tx_i+b)-1\right)$$

$$\theta(w)=\max_{\alpha_i\geq 0}\mathcal{L}(w,b,\alpha)$$

$$\min_{w,b}\theta(w)=\min_{w,b}\max_{\alpha_i\geq 0}\mathcal{L}(w,b,\alpha)=p^*$$

$$\max_{\alpha_i\geq 0}\min_{w,b}\mathcal{L}(w,b,\alpha)=d^*$$

$$\frac{\partial\mathcal{L}}{\partial w}=0\Rightarrow w=\sum_{i=1}^{n}\alpha_iy_ix_i$$

$$\frac{\partial\mathcal{L}}{\partial b}=0\Rightarrow \sum_{i=1}^{n}\alpha_iy_i=0$$

$$\mathcal{L}(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\alpha_i\left(y_i(w^Tx_i+b)-1\right)$$

$$\begin{aligned}\mathcal{L}(w,b,\alpha)&=\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_jy_iy_jx_i^Tx_j-\sum_{i,j=1}^{n}\alpha_i\alpha_jy_iy_jx_i^Tx_j-b\sum_{i=1}^{n}\alpha_iy_i+\sum_{i=1}^{n}\alpha_i\\&=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_jy_iy_jx_i^Tx_j\end{aligned}$$

$$K(\mathbf{x},\mathbf{z})=\langle\phi(\mathbf{x})\cdot\phi(\mathbf{z})\rangle,$$

$$f(\mathbf{x})=\sum_{i=1}^{\ell}\alpha_iy_i\langle\boldsymbol{\phi}(\mathbf{x}_i)\cdot\boldsymbol{\phi}(\mathbf{x})\rangle+b.$$

$$\begin{aligned}&\max_{\alpha}\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_jy_iy_j\kappa(x_i,x_j)\\&s.t.,\ \alpha_i\geq 0, i=1,\ldots,n\\&\qquad\sum_{i=1}^{n}\alpha_iy_i=0\end{aligned}$$

Similarities of SVM and NN – Non-linear and Parametric

Both SVM and NN can **map the input data to a higher dimensional space to assign a decision boundary**. Both classes of algorithms can approximate non-linear decision functions, with different approaches.

For SVM, it is done by using kernel tricks whereas
NN via non-linear activation functions.

Both parametric, though for different reasons.

In the case of the SVM, the typical parameters are:

- **the soft-margin parameter** C
- **the parameter of the kernel function** γ

Neural networks also use parameters, though they require significantly more of them.

- **The most important parameters concern the number of layers and their size, but also the number of training epochs and the learning rate**.

Both models are similar insofar as they are both parametric, but dissimilar with regards to the type and number of parameters that they require.

Difference Between SVM and NN – Training Data

The amount of information required to train the algorithm:

Support vector machines effectively use only a subset of a dataset as training data. This is because they reliably identify the decision boundary on the basis of the sole support vectors. **As a consequence, for well-separated classes, the number of observations required to train an SVM isn't high**.

With regards to neural networks, instead, the training takes place on the basis of the batches of data that feed into it. This means that the specific decision boundary that the neural network learns is highly dependent on the order in which the batches of data are presented to it. **This, in turn, requires processing the whole training dataset; or otherwise, the network may perform extremely poorly**

Difference between SVM and NN – Training Time

One further difference relates to the time required to train the algorithm.

SVMs are generally very fast to train, which is a consequence of the point we made in the previous section. The same is however not valid for neural networks.

Some particularly large NNs require in fact several days, sometimes weeks, in order to be trained.

This means that restarting the training and initializing the random weights differently, for example, is possible for SVMs but very expensive for NNs.

Difference between SVM and NN – Train Mode

Difference algorithms that neural networks and SVMs use for optimization:

Typically, optimization for neural network takes place through gradient descent, as this is the most common technique. The usage of gradient descent is however also one of the reasons why neural networks sometimes can't learn a function if their initial configuration places them at a function's local minimum.

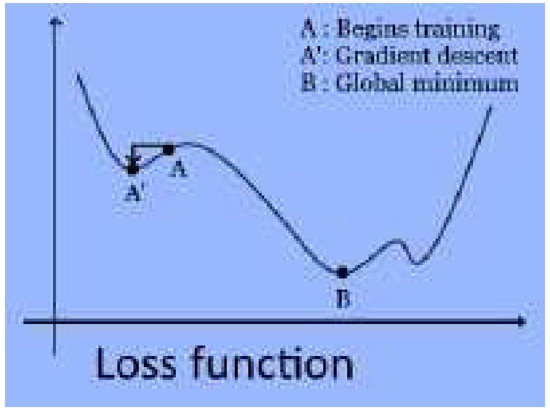
SVM use, instead, the method called quadratic programming. **Quadratic programming (QP)** consists of the optimization of a function according to linear constraints on its variables. Quadratic programming for SVMs is solved in practice with **Sequential Minimal Optimization (SMO),** which allows the identification of a likely solution by iteratively computing the analytical solution to a subset of the problem.

Difference between SVM and NN – Initialization

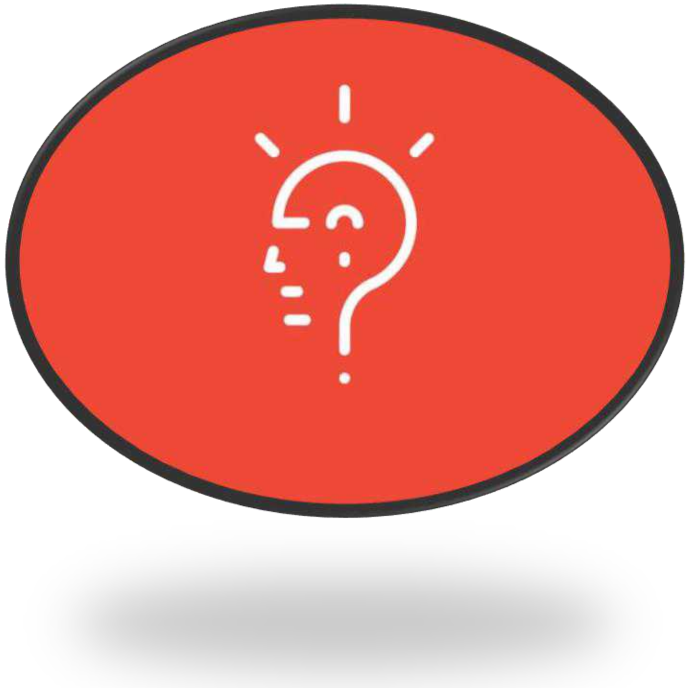
Initial configuration affects optimization:

NNs use gradient descent, this makes them sensitive to the initial randomization of its weight matrix. This is because, if the initial randomization places the neural network close to a local minimum of the optimization function, the accuracy will never increase past a certain threshold.

SVMs are more reliable instead, and they guarantee convergence to a global minimum regardless of their initial configuration.



Any Question?



Machine Learning

3 5

1 Back Propagation

2 Supporter Vector Machine

3 Machine Learning

4 Knowledge

TOP 10 Machine Learning Algorithms

1. **K-MEANS CLUSTERING**

   Aims to find groups in given data set. The number of groups is represented by a variable called K.

2. **NAIVE BAYES CLASSIFIER**

   A family of algorithms which assume that values of the features used in the classification are independent.

3. **K-NEAREST NEIGHBOR (KNN)**

   A simple algorithm that stores all existing data objects and classifies the new data objects based on a similarity measure.

TOP 10 Machine Learning Algorithms

4 **SUPPORT VECTOR MACHINES (SVM)**

Used to sort two data sets by similar classification. Draw lines (hyperplanes) that separate the groups according to some patterns.

5 **DECISION TREE**

A machine learning technique for data mining that creates classification or regression models in the shape of a tree structure.

6 **GENERALIZED LINEAR MODELS (GLM)**

Combines a number of models including linear regression models, logistic regression, Poisson regression, ANOVA, log-linear models and etc.

7 **NEURAL NETWORKS**

Nonlinear models which represent a metaphor for the functioning of the human brain.

TOP 10 Machine Learning Algorithms

8 ASSOCIATION RULES

If/then statements that aim to uncover some relationships between unrelated data in a given database.

9 GENETIC ALGORITHMS

A family of stochastic search algorithms witch mechanism is inspired by the process of neo-Darwinian evolution.

10 LATENT DIRICHLET ALLOCATION (LDA)

A generative probabilistic model designed for collections of discrete data.

1. K-means Clustering

1. Based on the value k,
2. Initialize the k cluster centroids (many ways).
3. Cluster the n inputs by assigning them to the nearest cluster centroids.
4. Re-calculate the new k cluster centroids based on the inputs.
5. Comparing with the previous clustering centroids, if none of the n inputs changed cluster in the last iteration, exit. Otherwise go to 3.

1. P-Norm Distances

For a point (x_1, x_2, ...,x_n) and a point (y_1, y_2, ...,y_n), the **Minkowski distance** of order p (**p-norm distance**) is defined as:

$$\text{1-norm distance} = \sum_{i=1}^{n} |x_i - y_i|$$

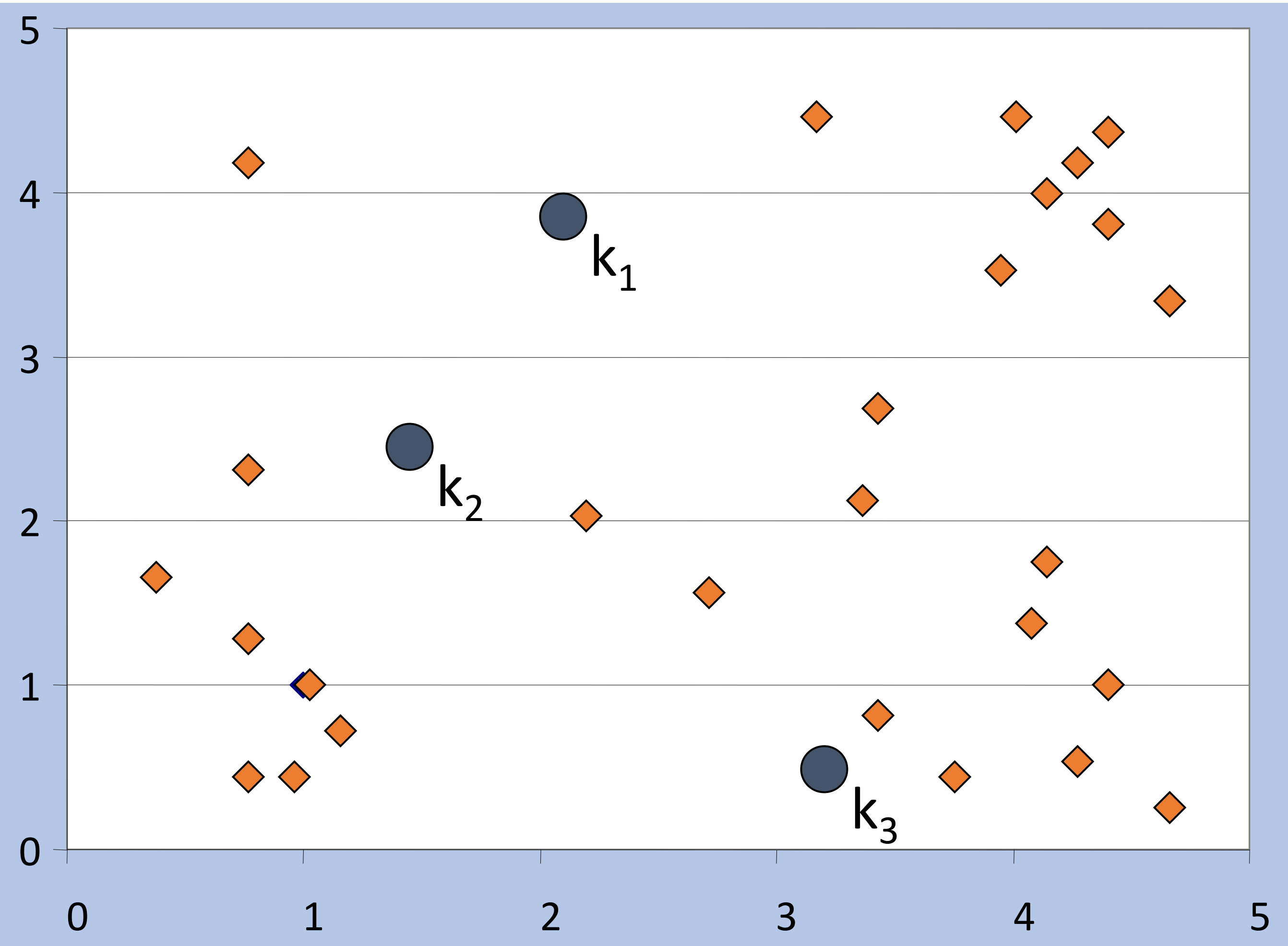
$$\text{2-norm distance} = \left(\sum_{i=1}^{n} |x_i - y_i|^2\right)^{1/2}$$

$$p\text{-norm distance} = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

$$\text{infinity norm distance} = \lim_{p\to\infty} \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

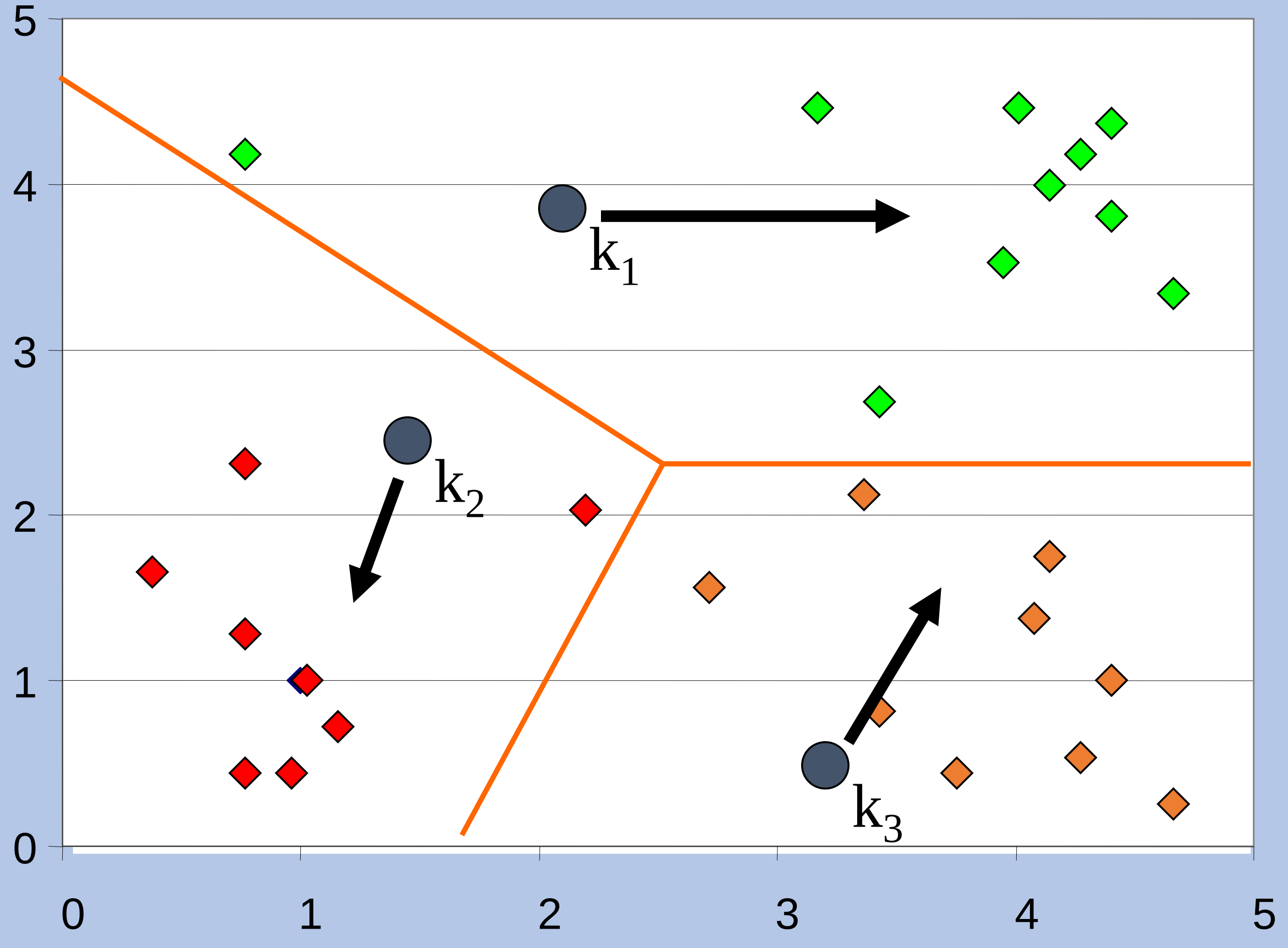
$$= \max\left(|x_1 - y_1|, |x_2 - y_2|, \ldots, |x_n - y_n|\right).$$

1. K-means for Clustering : Step 1



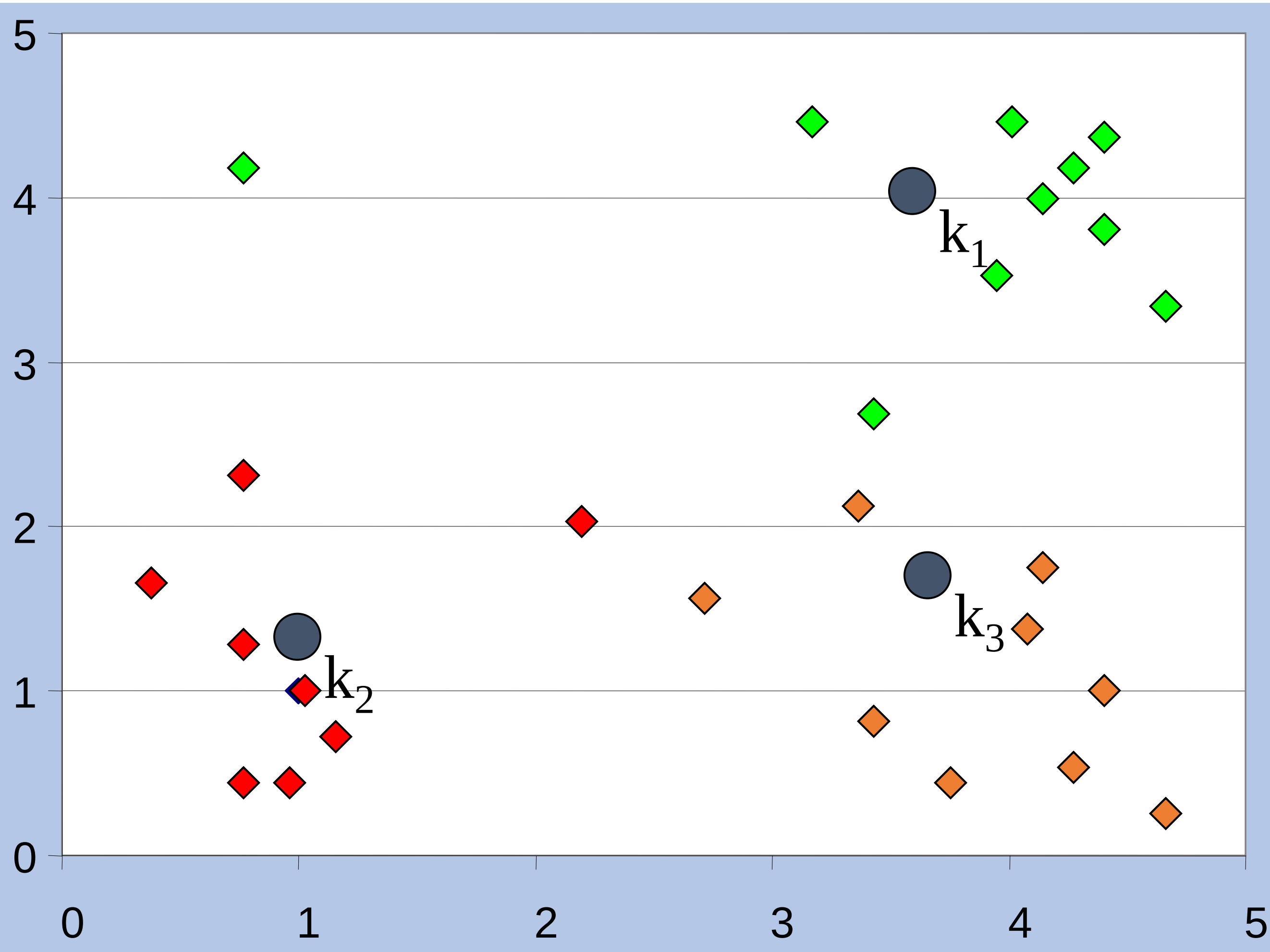
Feature: Position, Distance Metric: Euclidean Distance

1. K-means for Clustering : Step 2



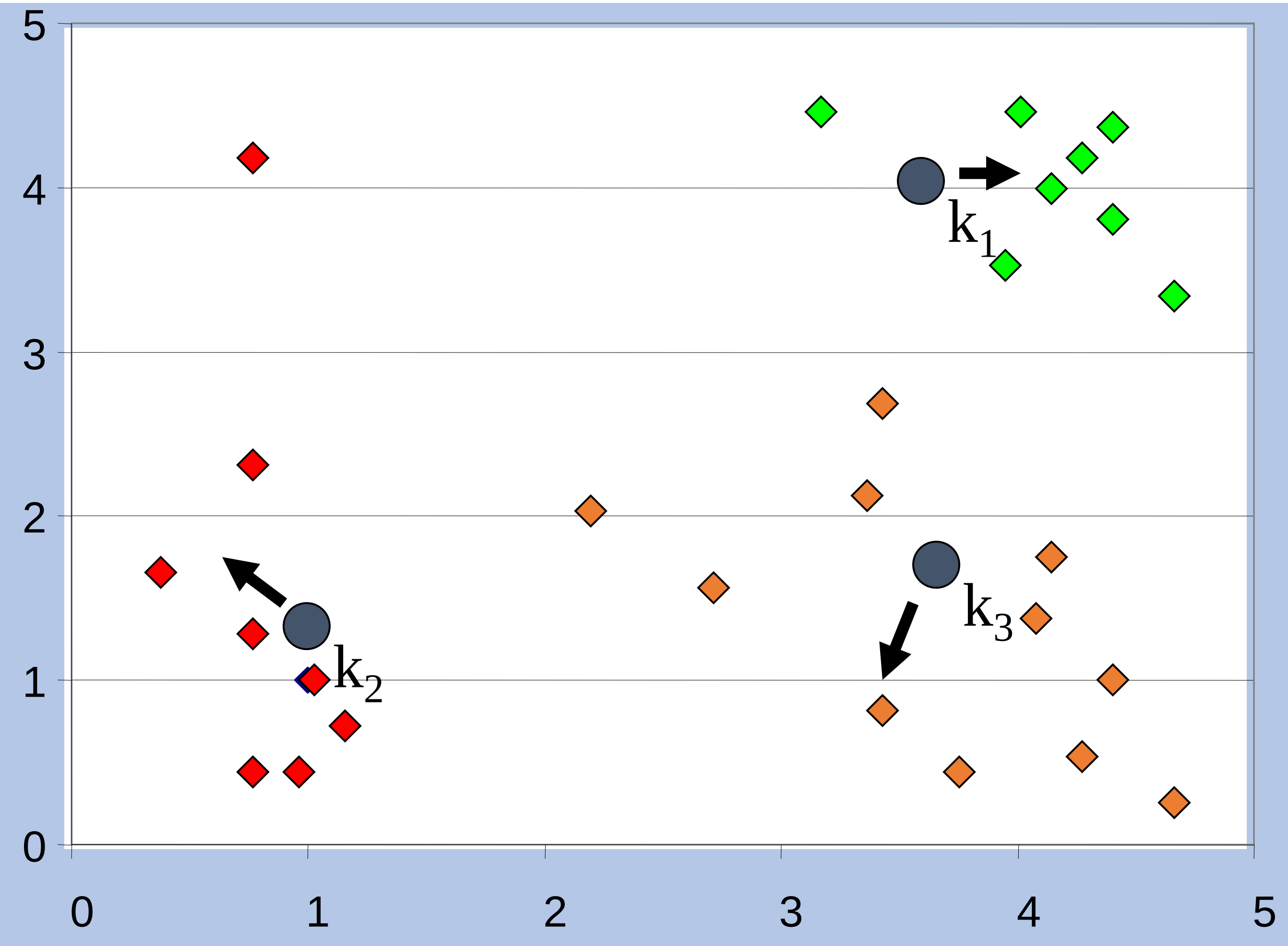
Feature: Position, Distance Metric: Euclidean Distance

1. K-means for Clustering : Step 3



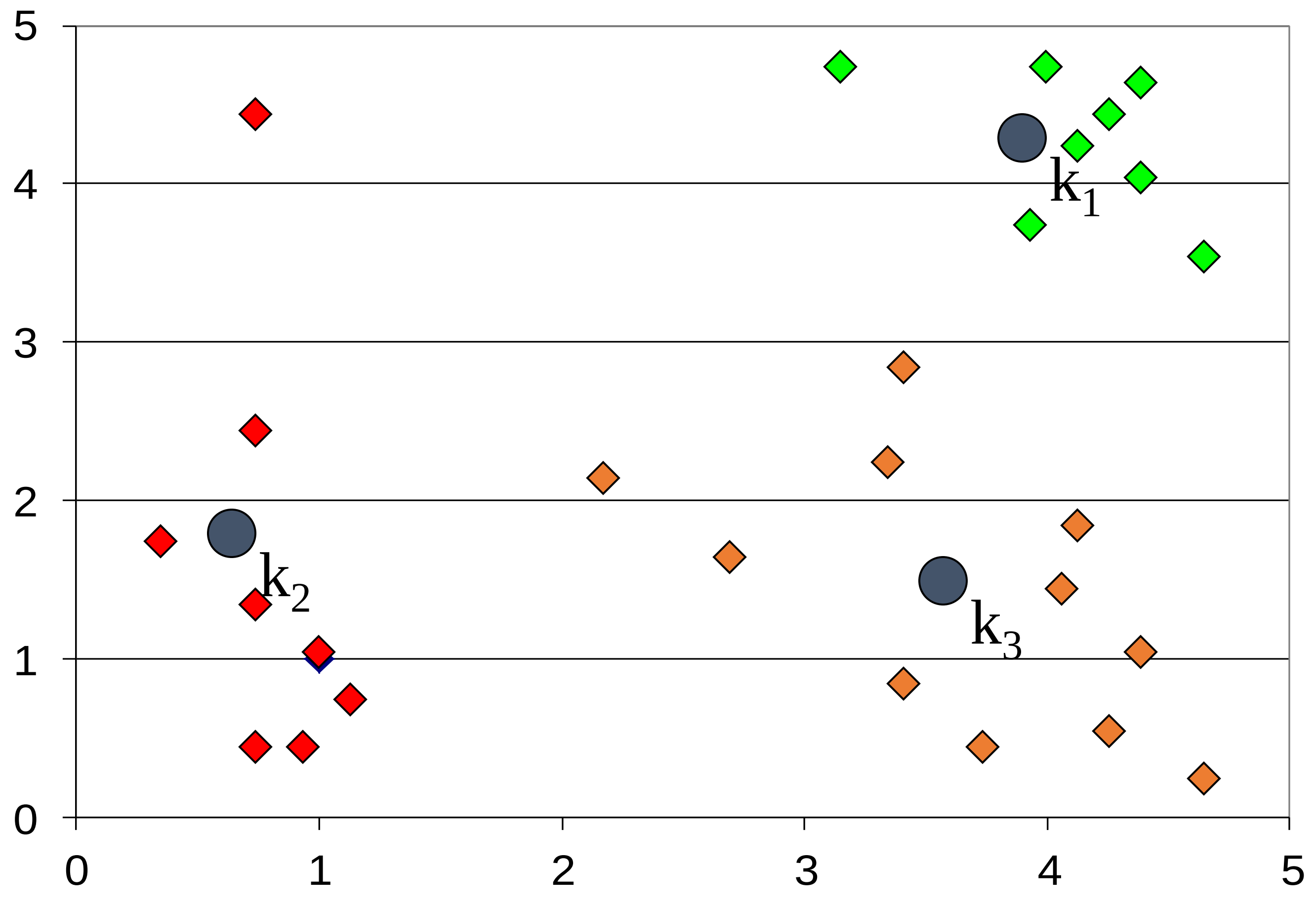
Feature: Position, Distance Metric: Euclidean Distance

1. K-means for Clustering : Step 4



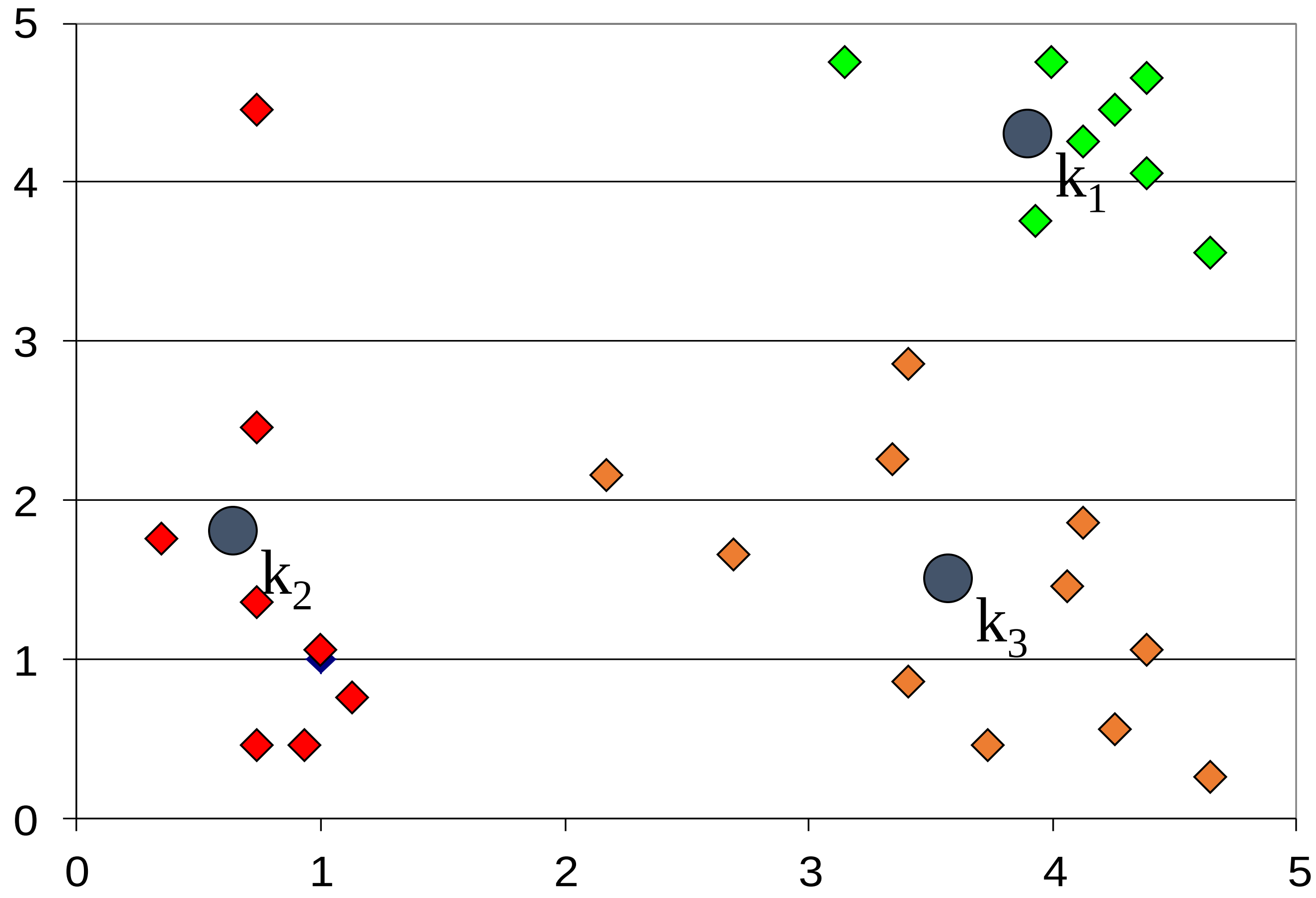
Feature: Position, Distance Metric: Euclidean Distance

1. K-means for Clustering : Step 5



Feature: Position, Distance Metric: Euclidean Distance

1. K-means for Clustering : Step 6



Feature: Position, Distance Metric: Euclidean Distance

Homework

- In a document classification applications, 7 text document objects from 3 categories of articles are represented by the points, which are:

Point	**1**	**2**	**3**	**4**	**5**	**6**	**7**
X	1	8	7	9	1	9	5
Y	6	3	1	8	2	3	6

- please use K-means algorithm to cluster the 7 document into three categories of articles (K=3)
- please specify which points belong to which cluster. For easy calculation, the initial three starting cluster center points are point 1, point 2 and point 3, also, please use L2 Distance to calculate the distance between two points
- The L2 distance between two points a=(x1, y1) and b=(x2, y2) is defined as:

$$d_{a,b} = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

Hint

		(1, 6)	(8, 3)	(7, 1)	
Point		Distance to centroid of cluster 1	Distance to centroid of cluster 2	Distance to centroid of cluster 3	Cluster
1	(1, 6)	0	7.62	7.81	**1**
2	(8, 3)				
3	(7, 1)				
4	(9, 8)				
5	(1, 2)				
6	(9, 3)				
7	(5, 6)				

CS 103 -12
Support Vector Machine and Machine Learning

Jimmy Liu 刘江

2020-12-04