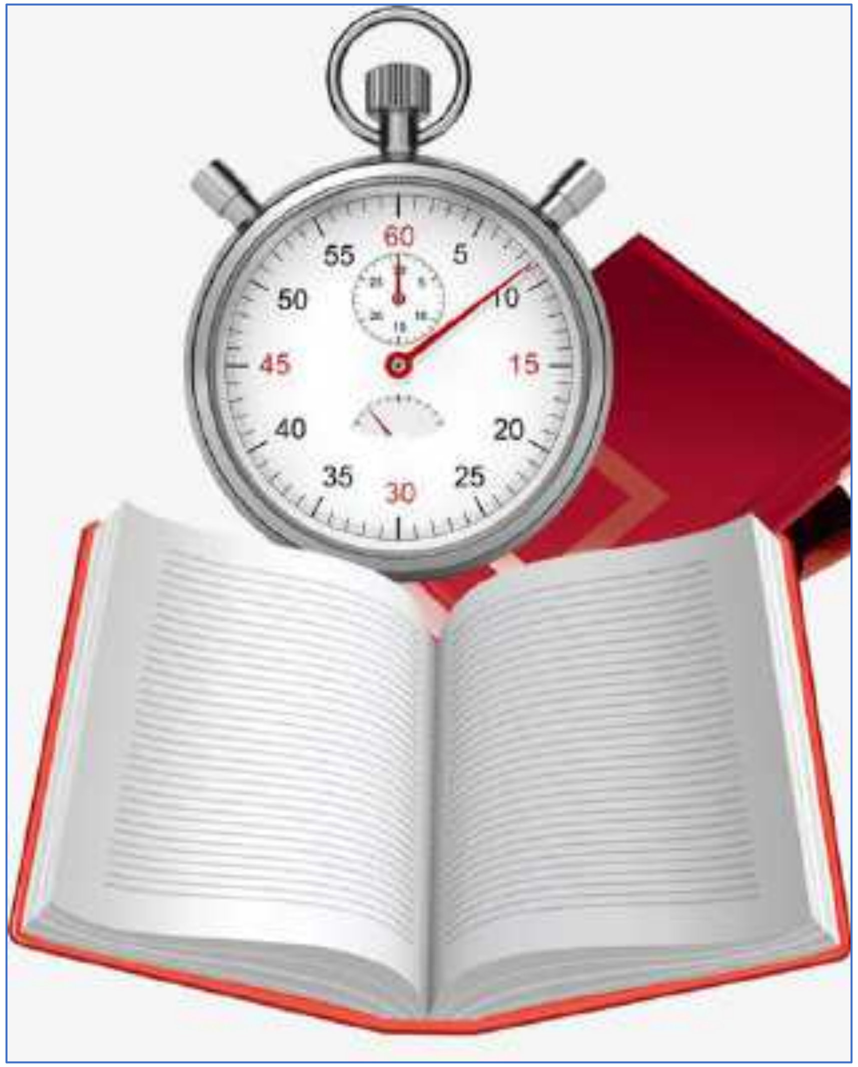
CS 103 -11 BP

Jimmy Liu 刘江

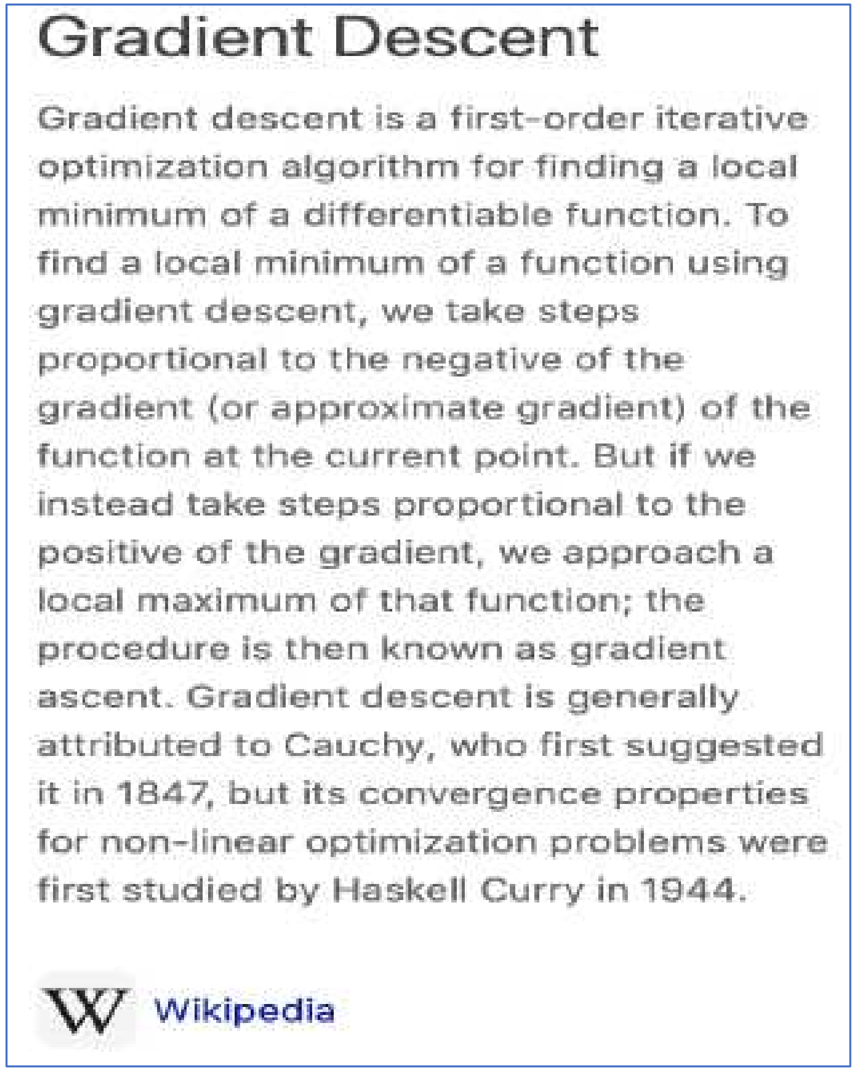
2020-11-27

Review



Med

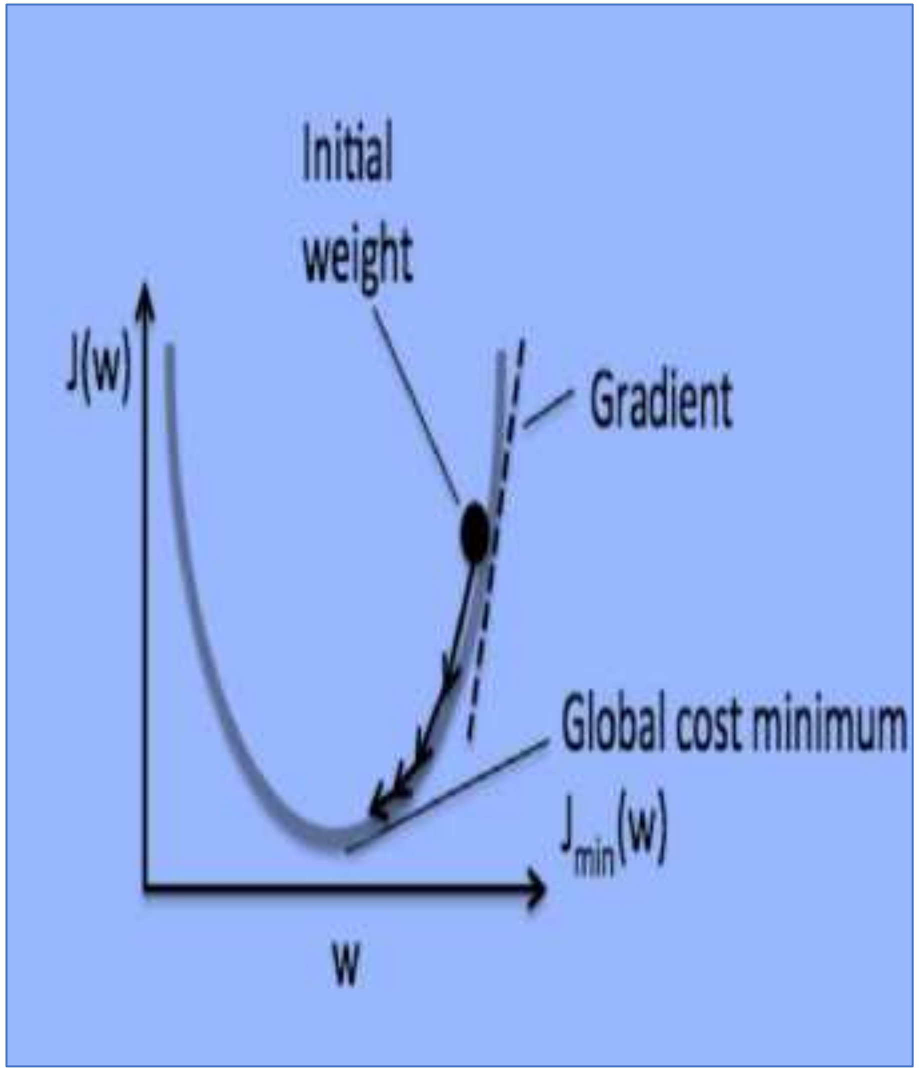
LMS Gradient Descent



Gradient Descent

Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. To find a local minimum of a function using gradient descent, we take steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point. But if we instead take steps proportional to the positive of the gradient, we approach a local maximum of that function; the procedure is then known as gradient ascent. Gradient descent is generally attributed to Cauchy, who first suggested it in 1847, but its convergence properties for non-linear optimization problems were first studied by Haskell Curry in 1944.

Wikipedia

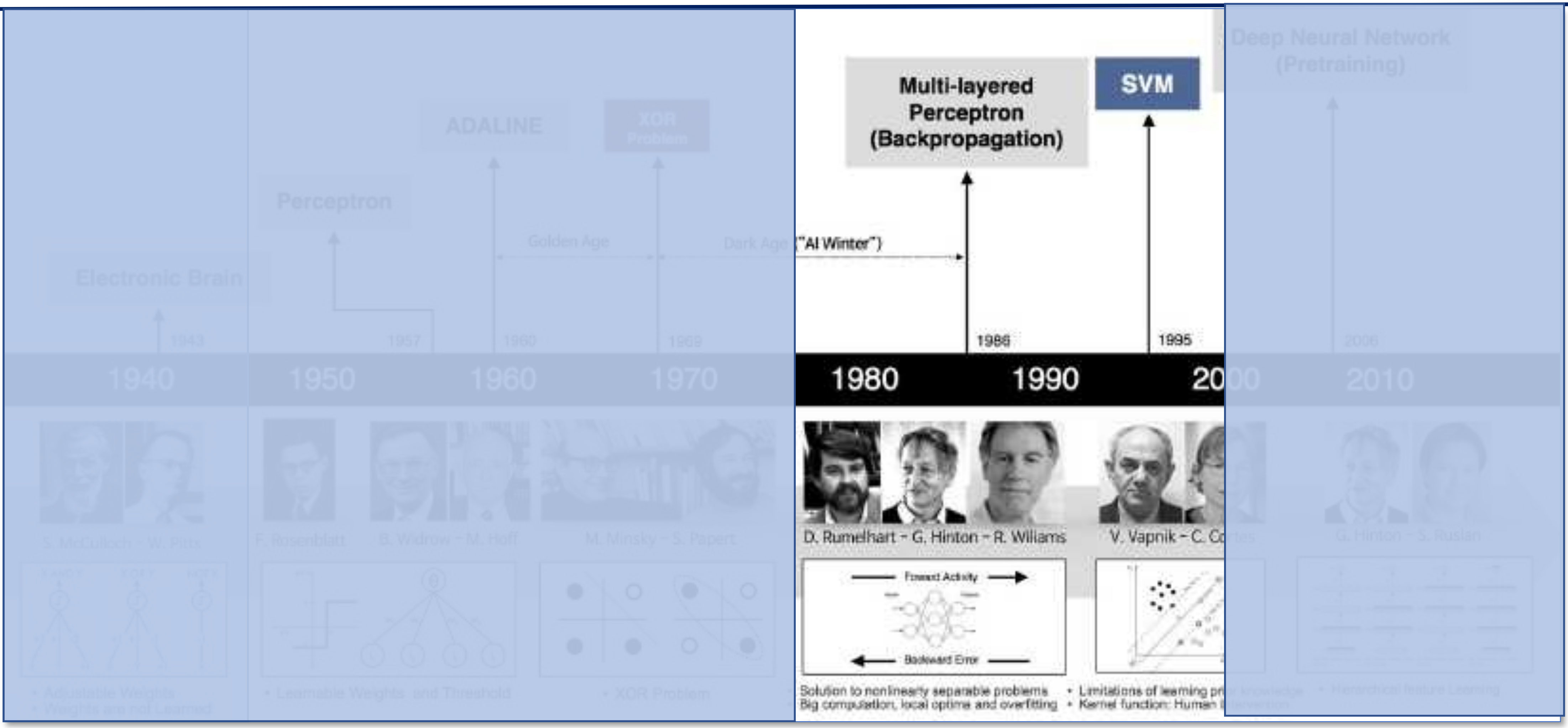


LMS Gradient Calculation = MADALINE learning

$$w_{t+1} = w_t - \alpha \frac{\partial J}{\partial w_t}$$

$$\begin{aligned}
&\frac{\partial J}{\partial w_j} \\
&= \frac{\partial}{\partial w_j} \frac{1}{2} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right)^2 \\
&= \frac{1}{2} \frac{\partial}{\partial w_j} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right)^2 \\
&= \boxed{} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right) \frac{\partial}{\partial w_j} \left(y^{(i)} - \phi(z)_A^{(i)}\right) \\
&= \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right) \frac{\partial}{\partial w_j} \left(y^{(i)} - \sum_i \left(w_j^{(i)} x_j^{(i)}\right)\right) \\
&= \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right)\left(-x_j^{(i)}\right) \\
&= -\sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right) x_j^{(i)}
\end{aligned}$$

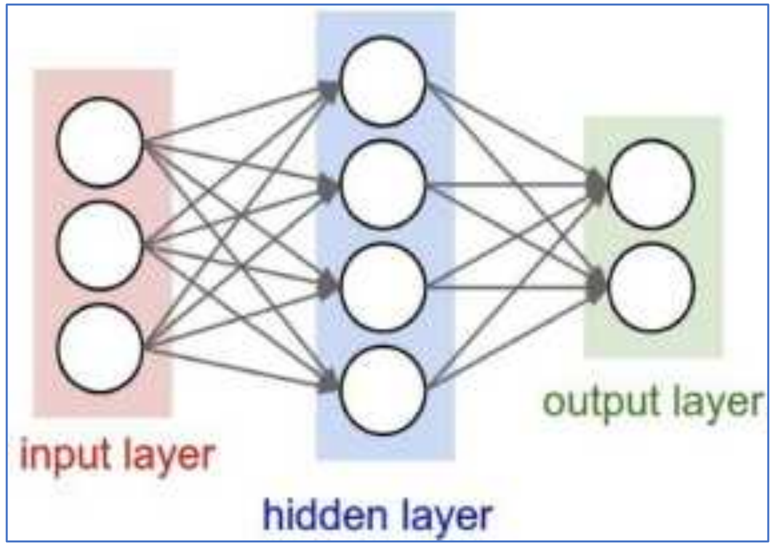
AI algorithm Developments – Machine Learning



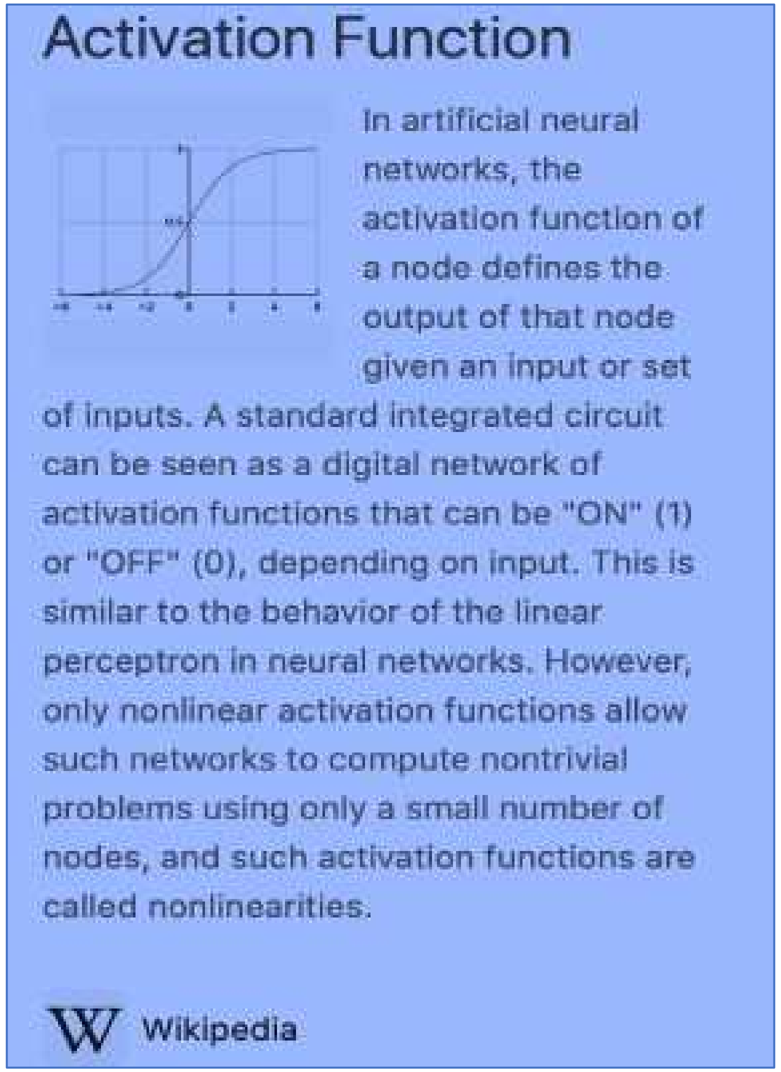
Electronic Brain
Perceptron
ADALINE
XOR Problem
Golden Age
Dark Age ("AI Winter")
Multi-layered Perceptron (Backpropagation)
SVM
Deep Neural Network (Pretraining)
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch - W. Pitts
F. Rosenblatt
B. Widrow - M. Hoff
M. Minsky - S. Papert
D. Rumelhart - G. Hinton - R. Williams
V. Vapnik - C. Cortes
G. Hinton - S. Ruslan
Forward Activity
Backward Error
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
Solution to nonlinearly separable problems
Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

3 Key Network Components of Neural Network

- Network Architecture
- Transfer Function
- Learning Rule



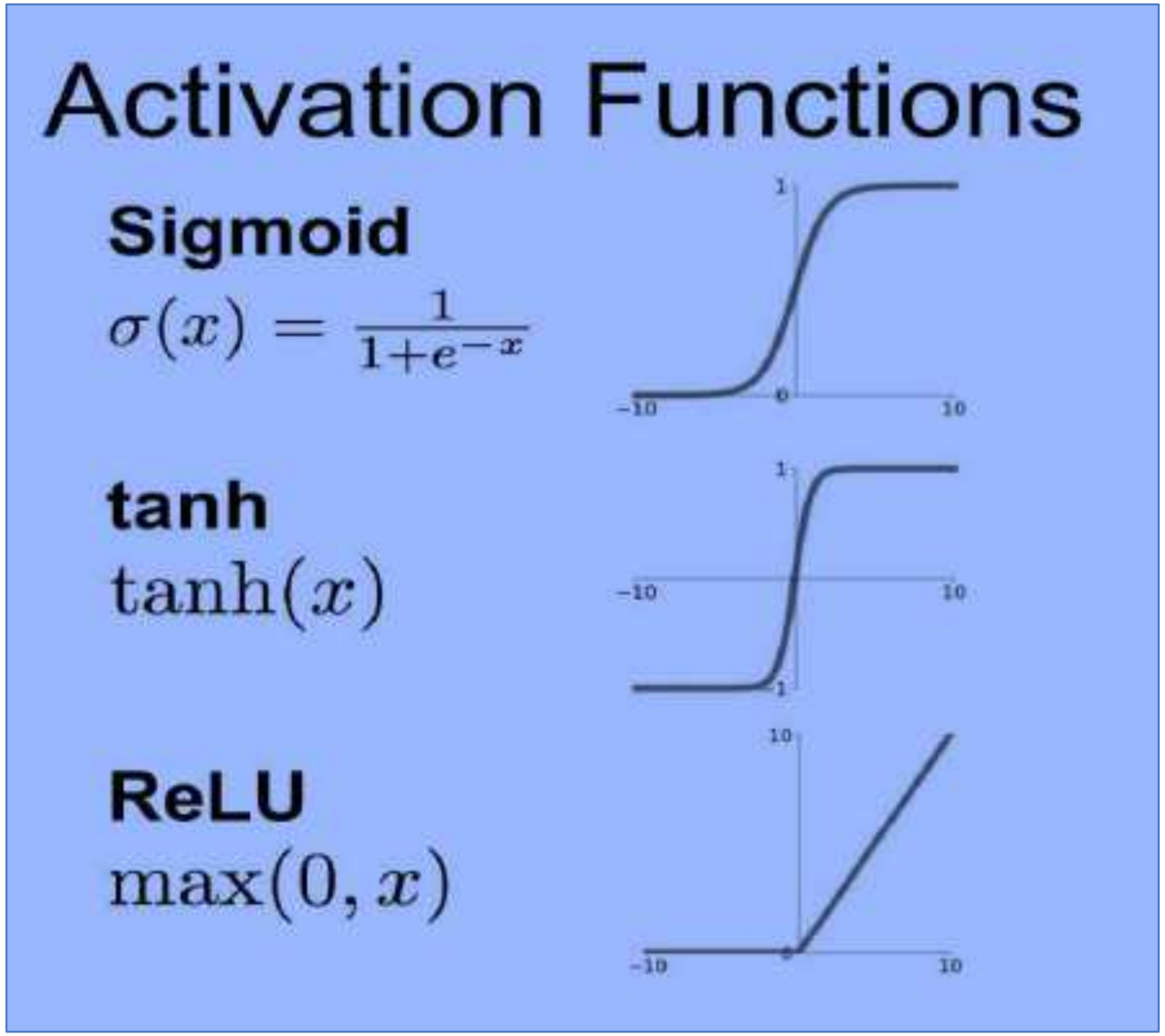
Activation (Transfer) Function Extension



Activation Function

In artificial neural networks, the activation function of a node defines the output of that node given an input or set of inputs. A standard integrated circuit can be seen as a digital network of activation functions that can be "ON" (1) or "OFF" (0), depending on input. This is similar to the behavior of the linear perceptron in neural networks. However, only nonlinear activation functions allow such networks to compute nontrivial problems using only a small number of nodes, and such activation functions are called nonlinearities.

Wikipedia



Activation Functions

Sigmoid

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

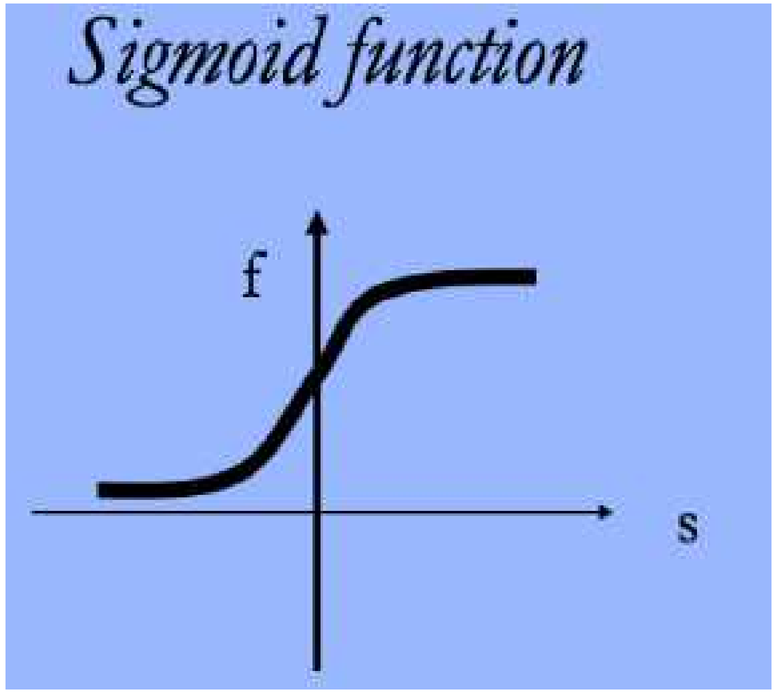
tanh

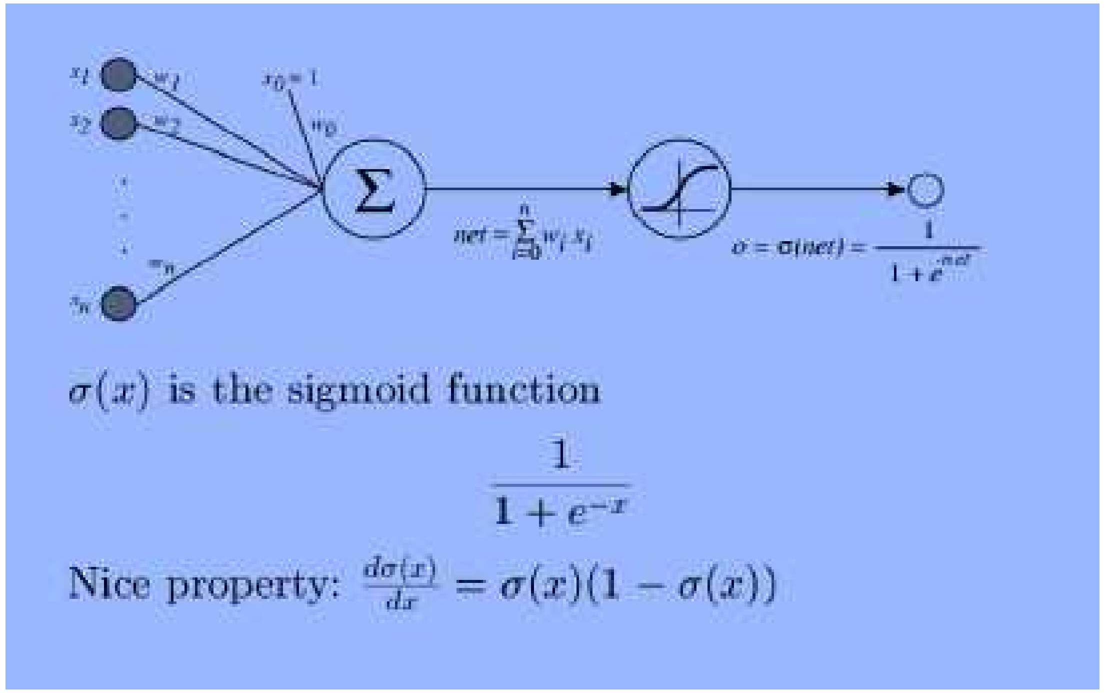
$\tanh(x)$

ReLU

$\max(0, x)$

Sigmoid Transfer Function




x_1
x_2
x_n
w_1
w_2
w_n
$x_0 = 1$
w_0
Σ
$net = \sum_{i=0}^{n} w_i x_i$
$o = \sigma(net) = \frac{1}{1+e^{-net}}$

$\sigma(x)$ is the sigmoid function

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

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

Sigmoid Unit

Learning Rule Extension – Back Propagation

In machine learning, backpropagation(backprop, BP) is a widely used algorithm in training feedforward neural networks for supervised learning. Generalizations of backpropagation exists for other artificial neural networks (ANNs), and for functions generally. These classes of algorithms are all referred to generically as "backpropagation". In fitting a neural network, backpropagation computes the gradient of the loss function with respect to the weights of the network for a single input–output example, and does so efficiently, unlike a naive direct computation of the gradient with respect to each weight individually. This efficiency makes it feasible to use gradient methods for training multilayer networks, updating weights to minimize loss; gradient descent, or variants such as stochastic gradient descent, are commonly used. The backpropagation algorithm works by computing the gradient of the loss function with respect to each weight by the chain rule, computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule.

Wikipedia

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 artifi cial intelligence in co vid-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

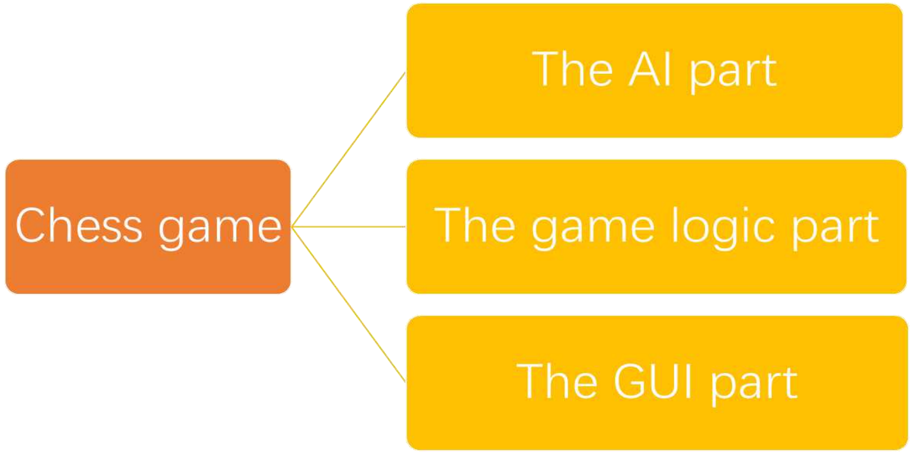
Group 1
Fighting the Landlord With AI

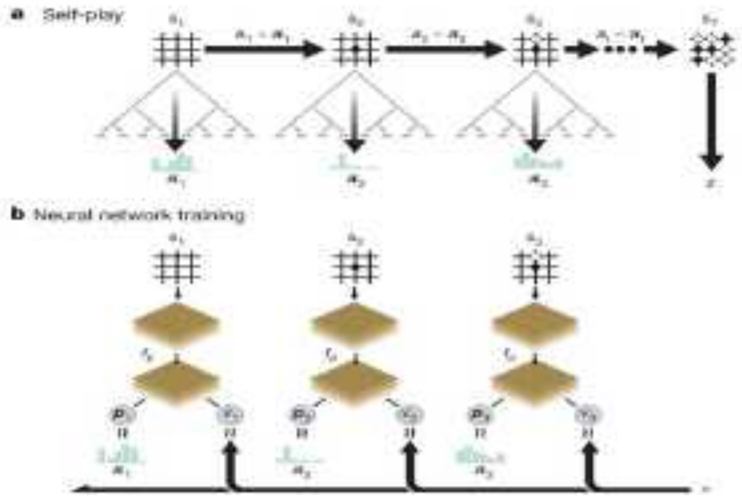
- Basic design
  - Implement the frame and basic game flow of the game··········FINISHED
  - Implement the rules of playing cards in detail··············ONGOING
  - Design the graphical interfaces·····························PLANNING
- Adding AI
  - Learn basic AI and algorithm································ONGOING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

- Find it complex to implement the rules in detail
- Teammates divided into to parts: One part working on basic logic and the other part focus on AI design

Group3： High Score Gamer

New Road Map





```
\documentclass[conference,compsoc]{IEEEtran}

% *** CITATION PACKAGES ***
%
\ifCLASSOPTIONcompsoc
  % IEEE Computer Society needs nocompress option
  % requires cite.sty v4.0 or later (November 2003)
  \usepackage[nocompress]{cite}
\else
  % normal IEEE
  \usepackage{cite}
\fi
% *** MATH PACKAGES ***
%
\usepackage{amsmath}
% \usepackage{algorithm}
% \usepackage{algorithmic}
\usepackage[ruled,vlined]{algorithm2e}
% \renewcommand{\algorithmicrequire}{\textbf{Input:}}
% \renewcommand{\algorithmicensure}{\textbf{Output:}}
\usepackage{array}
\usepackage{url}
\usepackage{float}

\hyphenation{op-tical net-works semi-conduc-tor}


\begin{document}

\title{Report on Introduction to Artificial Intelligence}


% author names and affiliations
% use a multiple column layout for up to three different
% affiliations
\author{\IEEEauthorblockN{Beichen Huang 11912012}
\IEEEauthorblockA{Department of Computer Science and Engineering\\
Sustech\\
11912012@mail.sustech.edu.cn}}


\maketitle
```

Group 4： AI application on diabetes

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

- 尝试了Deep Multi-Scale 3D Convolutional Neural Network (CNN) for MRI Gliomas Brain Tumor Classification中使用的Intensity Normalization预处理方法和随机Flip数据增强方法，没有取得好的效果。
- 实现了2篇专用于医学影像分类的论文中的网络，没有取得好的效果。
- 尝试了随机旋转，加高斯噪声等数据增强方法，没有取得好的效果。
- 修复了训练框架中的一个致命bug，但重新实验后没有取得好的效果。

后续计划：

- 目前已经大致完成Unet,ResNet,DenseNet,Attention这些方向的实验，计划再尝试实现autoencoder。
- 尝试调整训练集和验证集的比例，减少在验证集上推理时的偶然性。
- 一周后重新做实验获得完整的实验数据并制作PPT。

Group 7

AI Applications in Breast Cancer Imaging

常规乳腺X线检查
影像引导下的乳腺组织学活检
乳腺超声检查
常规乳腺MRI检查

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

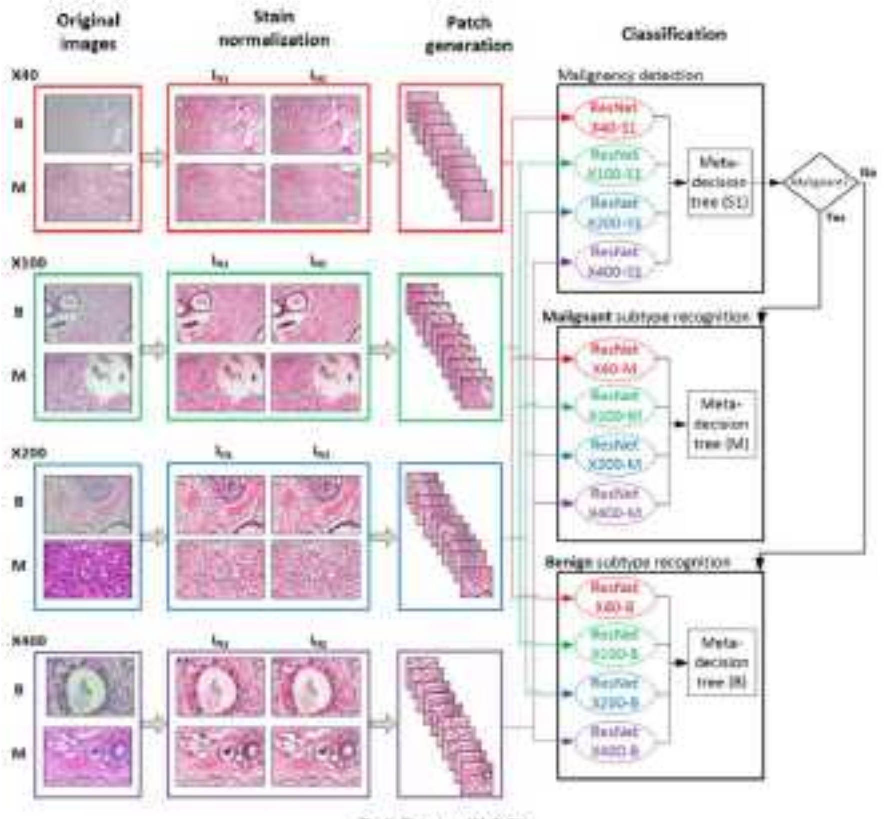
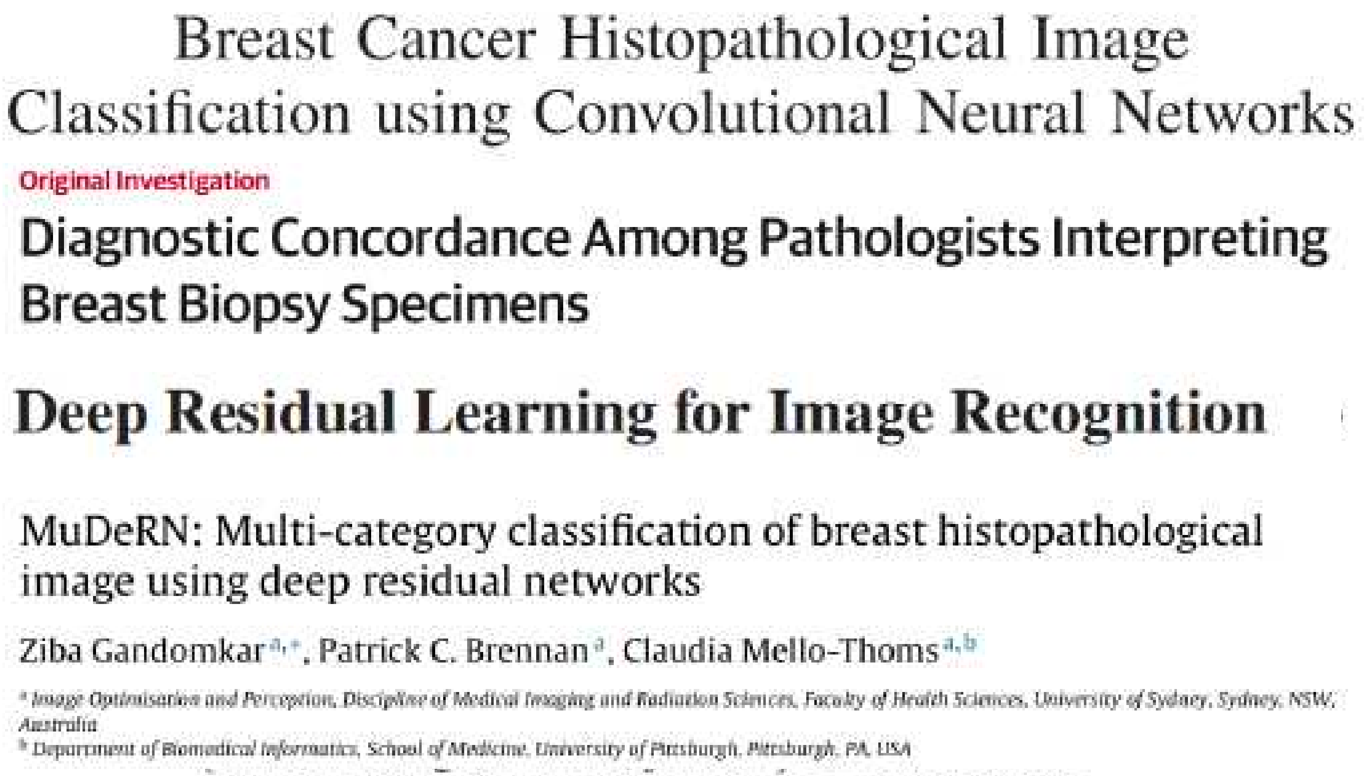


Fig. 1. The steps of MuDeRN.


Breast Cancer Histopathological Image Classification using Convolutional Neural Networks

Original Investigation
Diagnostic Concordance Among Pathologists Interpreting Breast Biopsy Specimens

Deep Residual Learning for Image Recognition

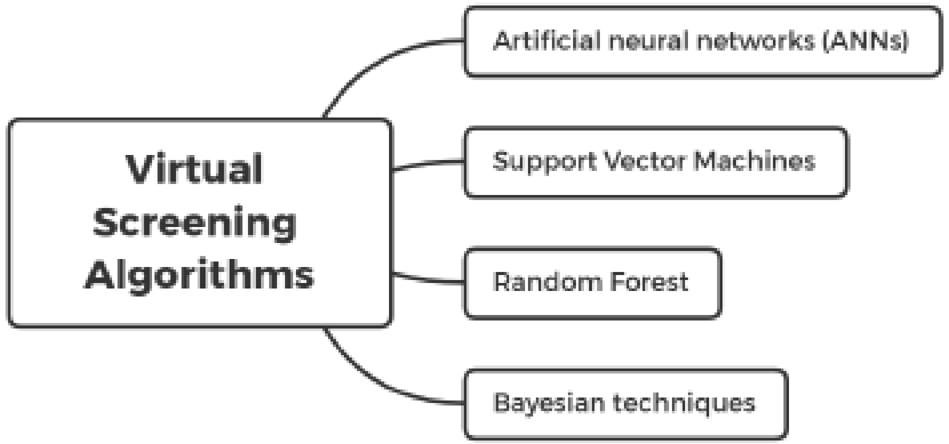
MuDeRN: Multi-category classification of breast histopathological image using deep residual networks

Ziba Gandomkar[a,*], Patrick C. Brennan[a], Claudia Mello-Thoms[a,b]

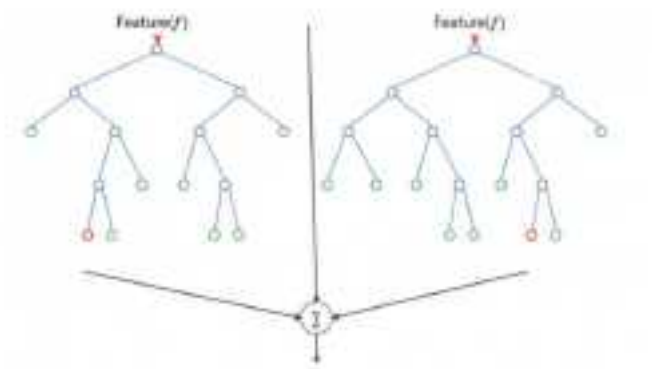
[a] Image Optimisation and Perception, Discipline of Medical Imaging and Radiation Sciences, Faculty of Health Sciences, University of Sydney, Sydney, NSW, Australia
[b] Department of Biomedical Informatics, School of Medicine, University of Pittsburgh, Pittsburgh, PA, USA

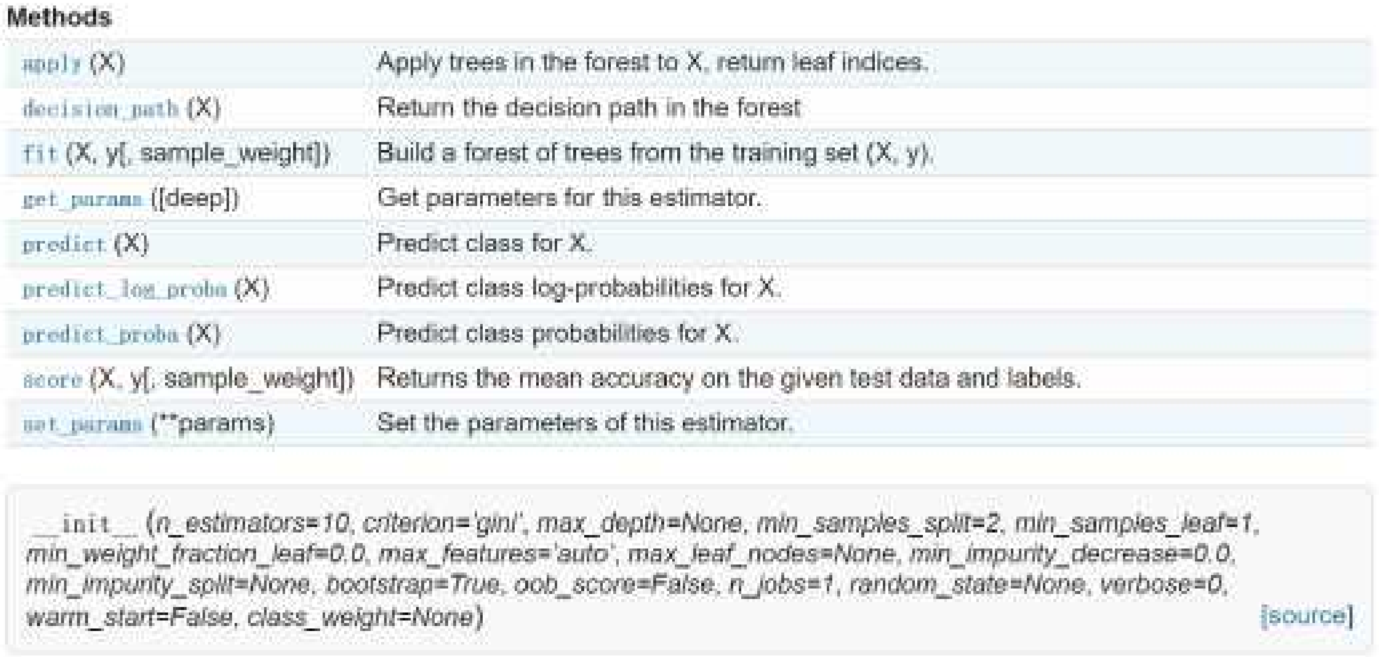
Group 8

Machine learning-based virtual screening algorithms in drug discovery ?


Virtual Screening Algorithms
Artificial neural networks (ANNs)
Support Vector Machines
Random Forest
Bayesian techniques

- In silico protein target prediction is a well-established computational technique that offers an alternative avenue to infer **target-ligand interactions** by utilizing known bioactivity information.
- **Random Forest**





Methods

`apply` (X)	Apply trees in the forest to X, return leaf indices.
`decision_path` (X)	Return the decision path in the forest
`fit` (X, y[, sample_weight])	Build a forest of trees from the training set (X, y).
`get_params` ([deep])	Get parameters for this estimator.
`predict` (X)	Predict class for X.
`predict_log_proba` (X)	Predict class log-probabilities for X.
`predict_proba` (X)	Predict class probabilities for X.
`score` (X, y[, sample_weight])	Returns the mean accuracy on the given test data and labels.
`set_params` (**params)	Set the parameters of this estimator.

```
__init__ (n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1,
min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0,
min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0,
warm_start=False, class_weight=None)                                                      [source]
```

3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.19.2 documentation (scikit-learn.org)

Group 9

上周工作汇报

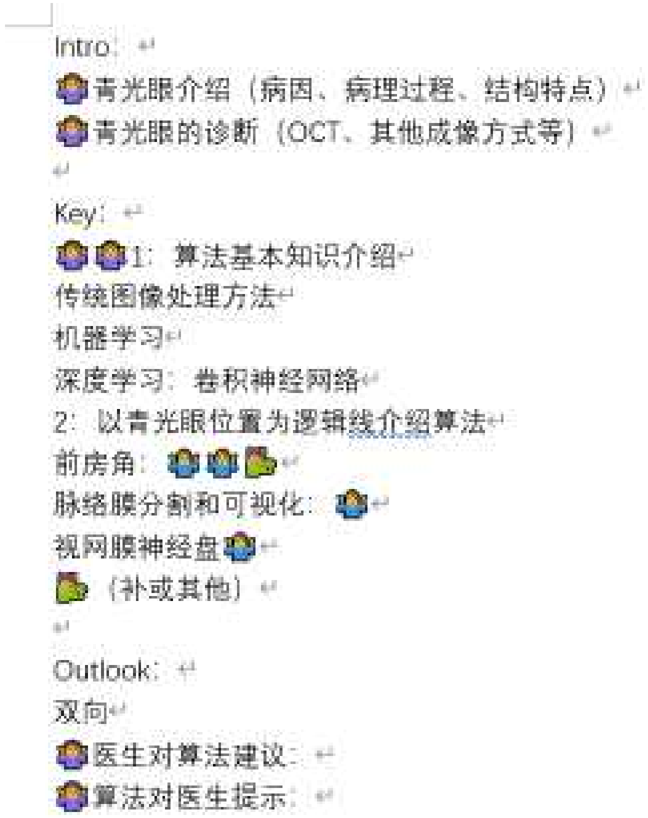
1. 11.20开会：一起填充和撰写文章内容
2. 完成了背景介绍和主体第一部分的内容

这周进度情况

1. 以青光眼的分类为逻辑线进行分类和分工：脉络膜分割、前房角、视神经乳头凹陷、虹膜和其他
2. 11.27开会：一起填充和撰写文章内容

下周任务安排

1. 继续阅读文献完成和补充算法介绍内容
2. 咨询和采访一些教授和医生完善Discussion部分



Intro：

🙆‍♀️青光眼介绍（病因、病理过程、结构特点）

🙆‍♀️青光眼的诊断（OCT、其他成像方式等）

Key：

🙆‍♀️🙆‍♀️1：算法基本知识介绍

传统图像处理方法

机器学习

深度学习：卷积神经网络

2：以青光眼位置为逻辑线介绍算法

前房角：🙆‍♂️🙆‍♂️🙋

脉络膜分割和可视化：🙆‍♂️

视网膜神经盘🙆‍♂️

🙋（补或其他）

Outlook：

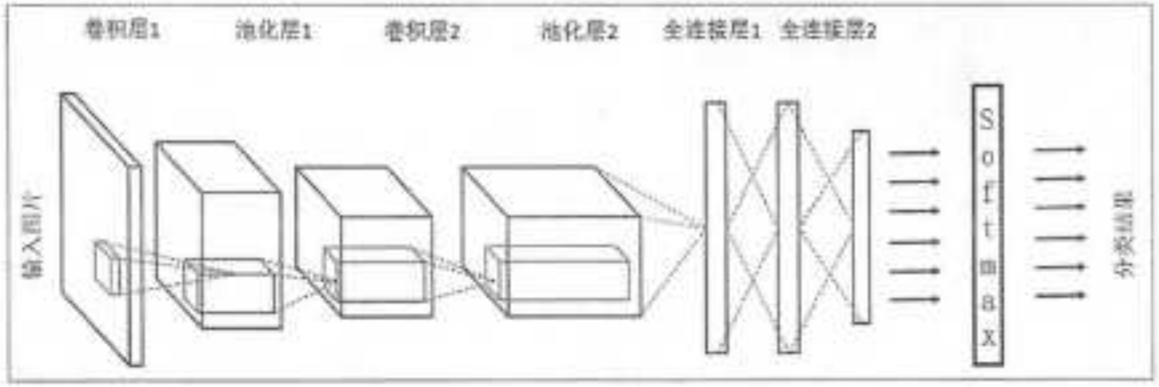
双向

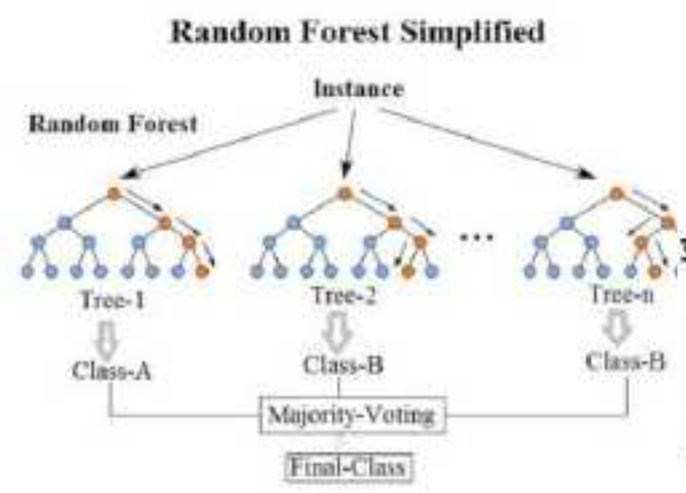
🙆‍♀️医生对算法建议：

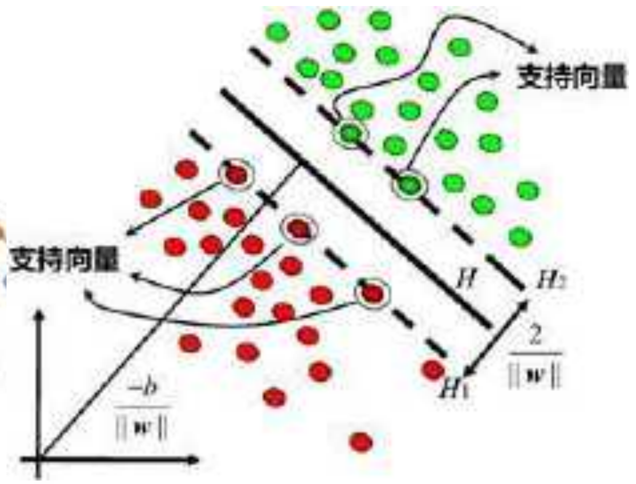
🙆‍♀️算法对医生提示：

Group 10

本周进展：文献收集和大致阅读完毕，制定了综述框架，开始撰写综述。







多层感知器（MLP）

卷积神经网络（CNN）

Algorithms for Cataract Detection

支持向量机（SVM）

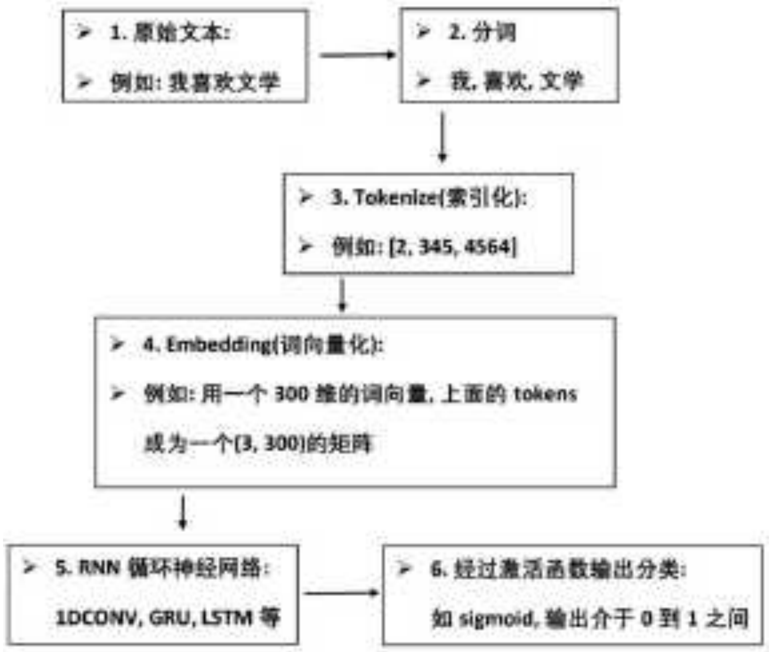
随机森林分类器（Random Forest）

Group 11

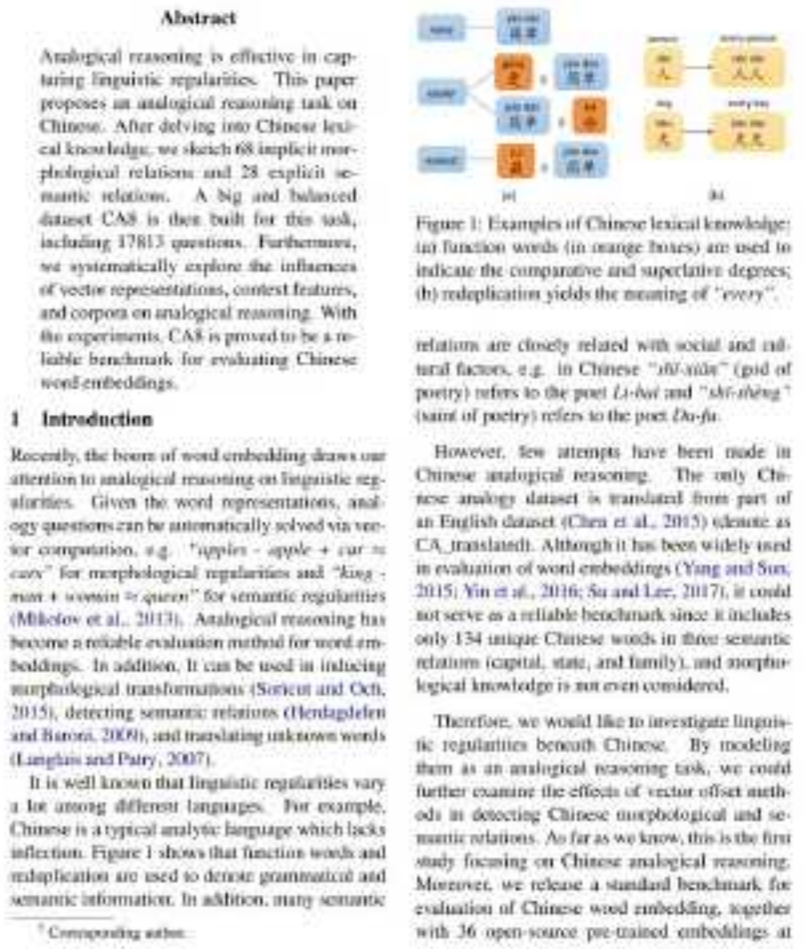
小组主题：图片文本识别与文本情感分析

上周进展：

1. 后期分工，文献收集，代码实现；
2. 文本情感分析模型初探；



➢ 1. 原始文本:
➢ 例如: 我喜欢文学
➢ 2. 分词
➢ 我, 喜欢, 文学
➢ 3. Tokenize(索引化):
➢ 例如: [2, 345, 4564]
➢ 4. Embedding(词向量化):
➢ 例如: 用一个 300 维的词向量, 上面的 tokens 成为一个(3, 300)的矩阵
➢ 5. RNN 循环神经网络:
1DCONV, GRU, LSTM 等
➢ 6. 经过激活函数输出分类:
如 sigmoid, 输出介于 0 到 1 之间



Abstract

Analogical reasoning is effective in capturing linguistic regularities. This paper proposes an analogical reasoning task on Chinese. After delving into Chinese lexical knowledge, we sketch 68 implicit morphological relations and 28 explicit semantic relations. A big and balanced dataset CA8 is then built for this task, including 17813 questions. Furthermore, we systematically explore the influences of vector representations, context features, and corpora on analogical reasoning. With the experiments, CA8 is proved to be a reliable benchmark for evaluating Chinese word embeddings.

1 Introduction

Recently, the boom of word embedding draws our attention to analogical reasoning on linguistic regularities. Given the word representations, analogy questions can be automatically solved via vector computation, e.g. "*apples - apple + car ≈ cars*" for morphological regularities and "*king - man + woman ≈ queen*" for semantic regularities (Mikolov et al., 2013). Analogical reasoning has become a reliable evaluation method for word embeddings. In addition, It can be used in inducing morphological transformations (Soricut and Och, 2015), detecting semantic relations (Herdagdelen and Baroni, 2009), and translating unknown words (Langlais and Patry, 2007).

It is well known that linguistic regularities vary a lot among different languages. For example, Chinese is a typical analytic language which lacks inflection. Figure 1 shows that function words and reduplication are used to denote grammatical and semantic information. In addition, many semantic

Figure 1: Examples of Chinese lexical knowledge: (a) function words (in orange boxes) are used to indicate the comparative and superlative degrees; (b) reduplication yields the meaning of "*every*".

relations are closely related with social and cultural factors, e.g. in Chinese "*shī-xiān*" (god of poetry) refers to the poet *Li-bai* and "*shī-shèng*" (saint of poetry) refers to the poet *Du-fu*.

However, few attempts have been made in Chinese analogical reasoning. The only Chinese analogy dataset is translated from part of an English dataset (Chen et al., 2015) (denote as CA_translated). Although it has been widely used in evaluation of word embeddings (Yang and Sun, 2015; Yin et al., 2016; Su and Lee, 2017), it could not serve as a reliable benchmark since it includes only 134 unique Chinese words in three semantic relations (capital, state, and family), and morphological knowledge is not even considered.

Therefore, we would like to investigate linguistic regularities beneath Chinese. By modeling them as an analogical reasoning task, we could further examine the effects of vector offset methods in detecting Chinese morphological and semantic relations. As far as we know, this is the first study focusing on Chinese analogical reasoning. Moreover, we release a standard benchmark for evaluation of Chinese word embedding, together with 36 open-source pre-trained embeddings at

† Corresponding author.

第12组项目汇报 2020/11/27

- 项目内容：手势识别及应用
- 组员：张骥霄 杜鹏辉 车文心 张静远
- 本周进展：
  - ➢确立明确的应用方向，进一步细分任务
  - ➢完成数据集的整合，优化了数据处理部分
  - ➢正在完成3D-CNN模型的构建，预计未来两周将逐步对不同模型进行搭建和训练。

第13组

AI in Lab

Abstract:	Xi
Introduction:	Sun
Main:	Done!
Conclusion:	This week's job
Discussion:	Yu

We had built a data pool and allocated jobs for weeks after

We gonna draw conclusions individually first and collectively after

CONTENT

第14组

1. 重新规划分工以及时间安排。学习使用Texworks进行文本编辑。

2. 完成Introduction部分和大纲，整理完参考文献。

第15组

人工智能在无障碍设施领域中的使用调查小组进度报告

本周我们完成的内容有以下部分:

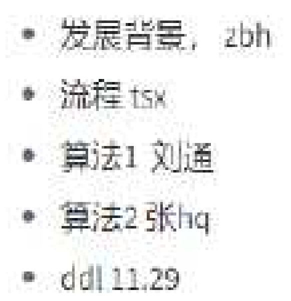
论文的简介以及引言部分，产品的介绍。

其中一个小组完成了智能眼镜辅助辅助系统的简介及，对卷积神经网络相关算法的研读和分析。正在寻找相关资料加深理解相关算法。

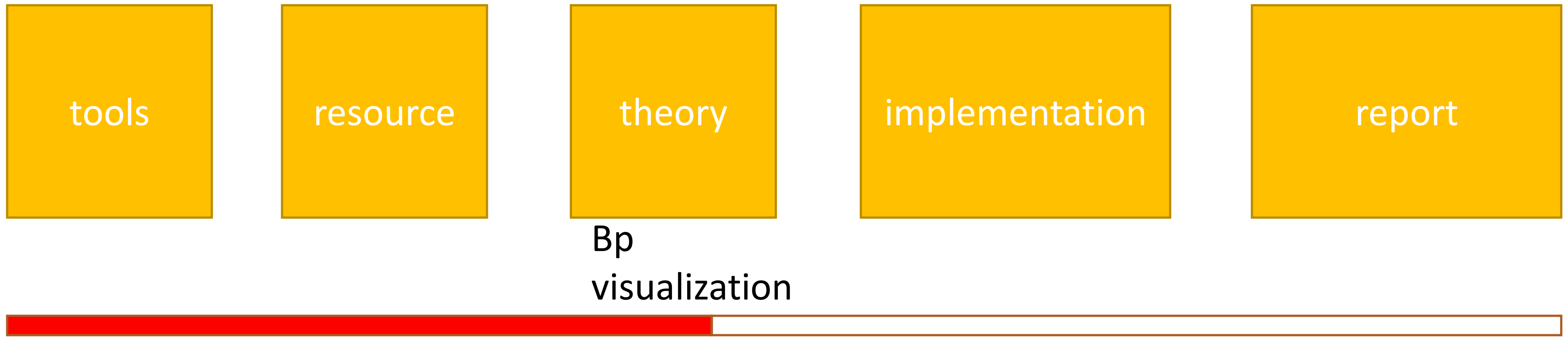
另外一个小组完成了基于人工智能技术的盲人导盲系统的分析，以及基于毫米波雷达和声呐的盲人辅助系统的硬件描述和功能实现的论文部分，接下来准备进行VFH算法以及相关算法的分析。

第十一周记录 第16组

- 进展顺利，11.29完成论文



- 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

- 自然语言处理
  - 分析对象
    - 词汇级
      - 语法分析
        - 中文分词
        - 词性标注
        - 命名实体识别
        - 新词发现
      - 语义表示
        - 语义表示
        - 语义消歧
      - 语义关系
        - 语义关系建模
        - 语义关系抽取
        - 语义关系计算
    - 句子级
      - 语句变换
        - 近义词替换
        - 语义归一化
        - 省略/纠错
      - 语句解析
        - 句法结构分析
        - 依存关系分析
      - 语句表示
        - 语义表示
        - 文本分类
      - 语句生成
        - 规则模板
        - 知识图谱
        - 机器翻译
    - 段落/篇章级
  - 分析内容
    - 词法分析
      - 中文分词
      - 词性标注
      - 命名实体识别
      - 新词发现
    - 语法分析
      - 句子级
        - 句法结构分析
        - 依存关系分析
      - 段落/篇章级
    - 语义分析
      - 词汇级
        - 语义表示
        - 语义消歧
      - 句子级
        - 语义表示
        - 文本分类
        - 意图识别
        - 情感分析
      - 段落/篇章级
    - 语用分析
      - 内容分析
        - 语境分析
        - 句意理解
      - 内容生成
        - 规则匹配
        - 知识推理
        - 机器翻译

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:

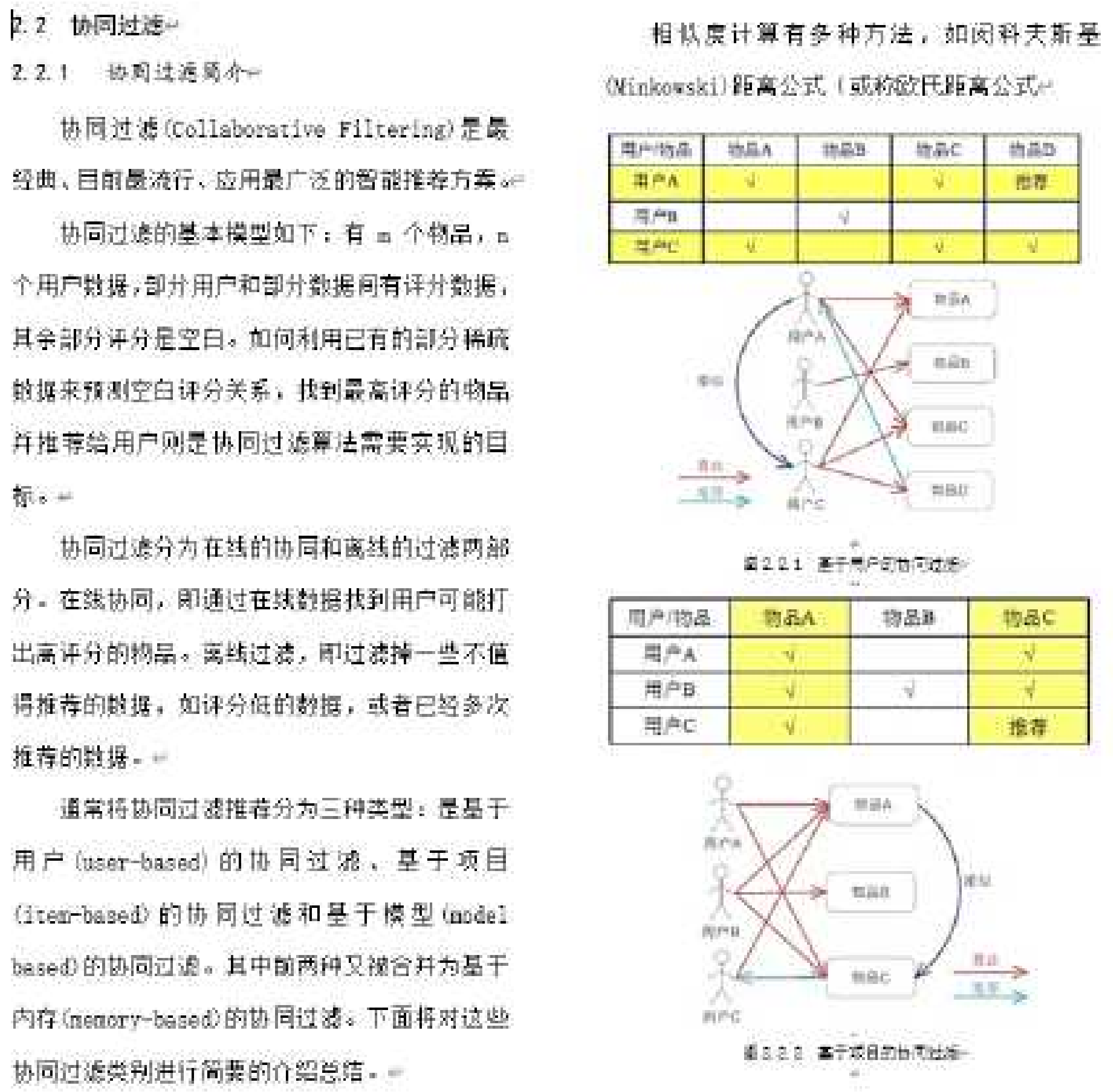
个性化推荐系统的背景和发展历程----进行中

基于内容的推荐算法----进行中

基于协同过滤的推荐算法----已完成

基于图网络的推荐算法----进行中

基于知识图谱的推荐算法----进行中



2.2 协同过滤

2.2.1 协同过滤简介

协同过滤(Collaborative Filtering)是最经典、目前最流行、应用最广泛的智能推荐方案。

协同过滤的基本模型如下：有 m 个物品，n 个用户数据，部分用户和部分数据间有评分数据，其余部分评分是空白。如何利用已有的部分稀疏数据来预测空白评分关系，找到最高评分的物品并推荐给用户则是协同过滤算法需要实现的目标。

协同过滤分为在线的协同和离线的过滤两部分。在线协同，即通过在线数据找到用户可能打出高评分的物品。离线过滤，即过滤掉一些不值得推荐的数据，如评分低的数据，或者已经多次推荐的数据。

通常将协同过滤推荐分为三种类型：是基于用户(user-based)的协同过滤，基于项目(item-based)的协同过滤和基于模型(model based)的协同过滤。其中前两种又被合并为基于内存(memory-based)的协同过滤。下面将对这些协同过滤类别进行简要的介绍总结。

相似度计算有多种方法，如闵科夫斯基(Minkowski)距离公式(或称欧氏距离公式

用户/物品	物品A	物品B	物品C	物品D
用户A	√		√	推荐
用户B		√		
用户C	√		√	√

图2.2.1 基于用户的协同过滤

用户/物品	物品A	物品B	物品C
用户A	√		√
用户B	√	√	√
用户C	√		推荐

图2.2.2 基于项目的协同过滤

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
    - 完成了数据的收集与标记（周末/平日/天气）
    - 了解Tensorflow的时间序列预测，查找相关的文档

- 论文综述部分
  - 查找不同的公交到站时间预测与规划方法

- 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

Group 20：给线稿上色的AI的算法研究

小组成员：韩晗（组长）、刘思语、赵晓蕾、陈松斌

1.
Use GAN to learn Image Colorization

- Discriminator trains the basic judgment ability through a small data set.
- Generator paints according to the line draft.
- Discriminator determines if the image is real.

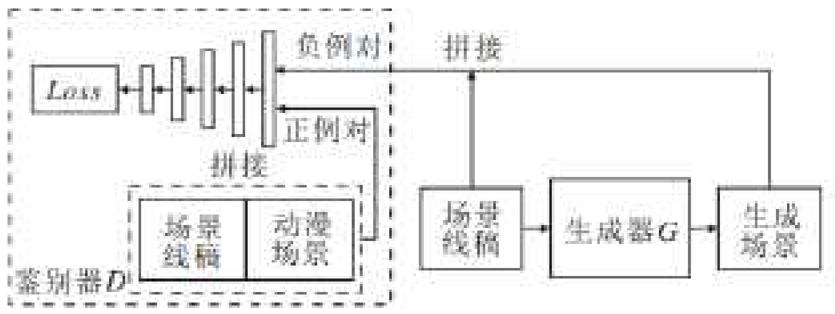
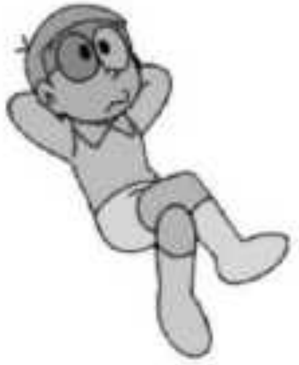


图 2　本文算法的网络总体结构
Fig. 2　Network structure of the proposed algorithm

2.
However, the semantic information in the original line draft is easily lost in the convolution. (like this →)



3.
Guided by the obvious information of the shallow layer of the scene line draft, and then integrated into the deep layer.
To enhance the ability on the important features such as color and position related to the line draft.

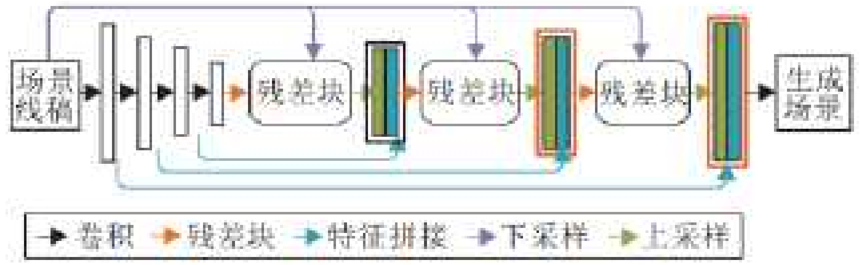
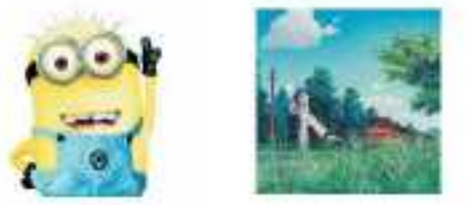
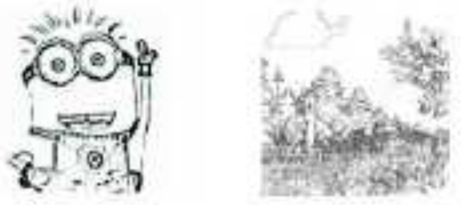


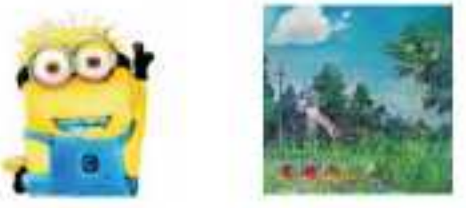
图 6　DIEU-Net 生成器的网络结构
Fig. 6　Generator structure of DIEU-Net



(a) 原图
(a) Original images



(b) 输入线稿
(b) Input sketches

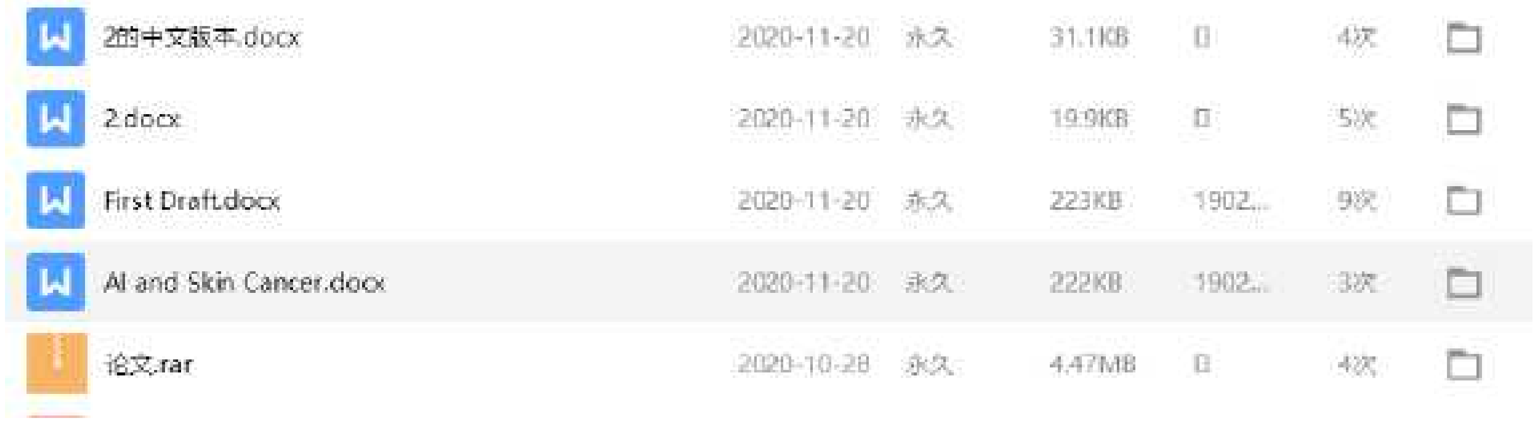


(c) 生成效果
(c) Generation results

Group21： AI and Skin Caner

小组成员： 李修治 刘宇欣 沈睿琦

目前的进度：
进度就是论文初稿已经完成， 目前在互相修改其他两人的论文， 预计下次上课之前完成论文二稿和PPT制作

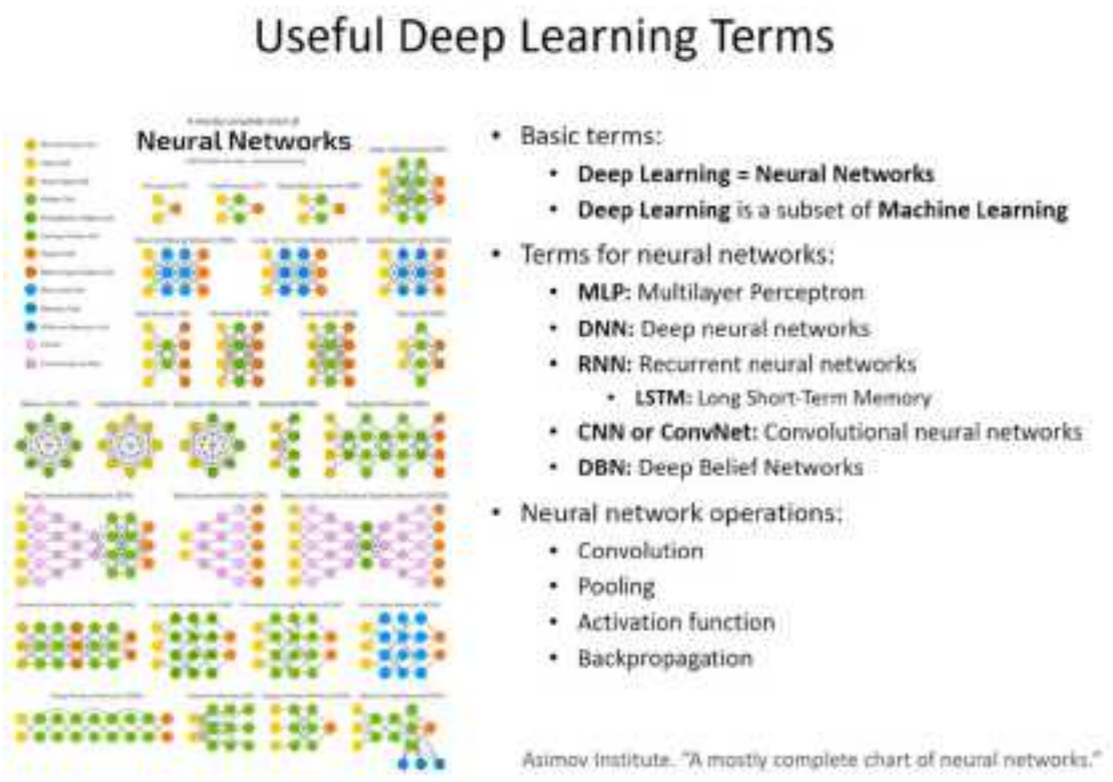


2的中文版本.docx	2020-11-20	永久	31.1KB	0	4次
2.docx	2020-11-20	永久	19.9KB	0	5次
First Draft.docx	2020-11-20	永久	223KB	1902...	9次
AI and Skin Cancer.docx	2020-11-20	永久	222KB	1902...	3次
论文.rar	2020-10-28	永久	4.47MB	0	4次

Group 22：深度学习在自动驾驶中的应用综述

小组成员：王晓轩

1.已完成所有相关资料的收集；
2.报告框架已经基本确定；
3.报告内容有待完善。
文章摘要：
目前，随着计算机视觉和机器学习相关理论的快速发展，基于人工智能技术的自动驾驶研究是学术界研究的热点，为传统的汽车产业的发展带来了新的契机。本文综述了深度学习相关理论和研究方法在自动驾驶中的进展，结合实例对深度学习在自动驾驶中的应用做了详细的介绍，并预测了未来的研究趋势。



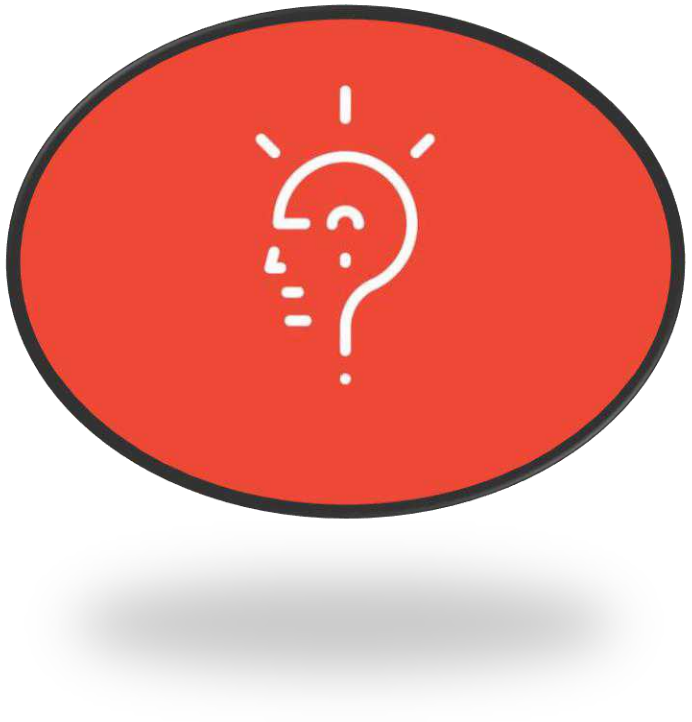
Useful Deep Learning Terms

Neural Networks

- Basic terms:
  - **Deep Learning = Neural Networks**
  - **Deep Learning** is a subset of **Machine Learning**
- Terms for neural networks:
  - **MLP:** Multilayer Perceptron
  - **DNN:** Deep neural networks
  - **RNN:** Recurrent neural networks
    - **LSTM:** Long Short-Term Memory
  - **CNN or ConvNet:** Convolutional neural networks
  - **DBN:** Deep Belief Networks
- Neural network operations:
  - Convolution
  - Pooling
  - Activation function
  - Backpropagation

Asimov Institute. "A mostly complete chart of neural networks."

Any Question?



Machine Language

3 5

Hidden Layers for NN

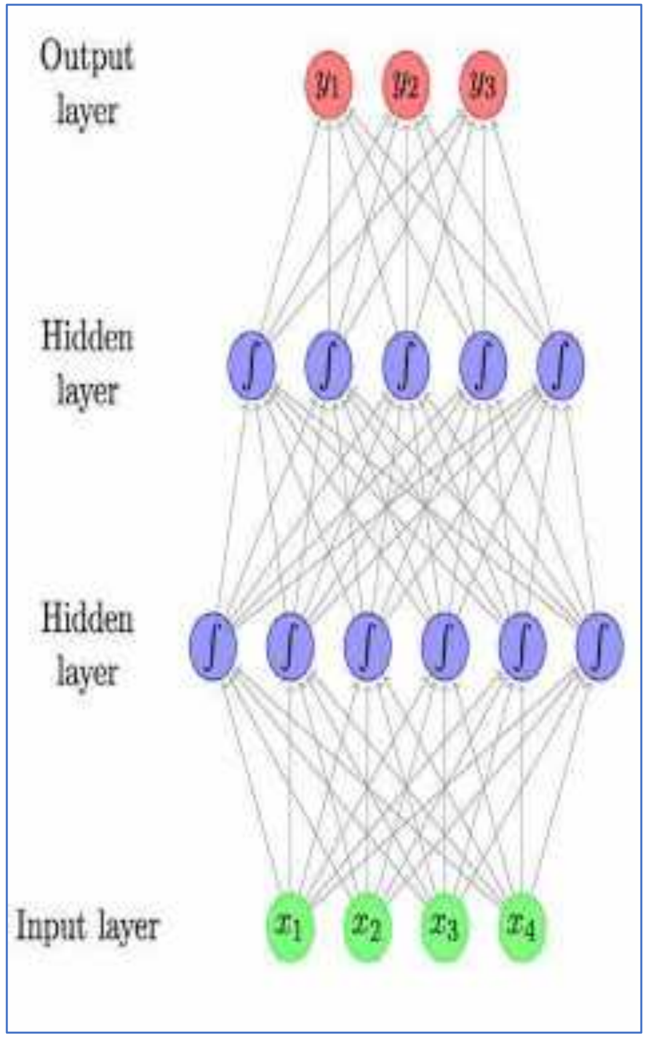
Theoretical answer:

- A neural network with 1 hidden layer is a **universal function approximator**
- Cybenko (1989): For any continuous function g(**x**), there exists a 1-hidden-layer neural net $h_\theta(\mathbf{x})$ s.t. $| h_\theta(\mathbf{x}) - g(\mathbf{x}) | < \epsilon$ for all **x**, assuming sigmoid activation functions

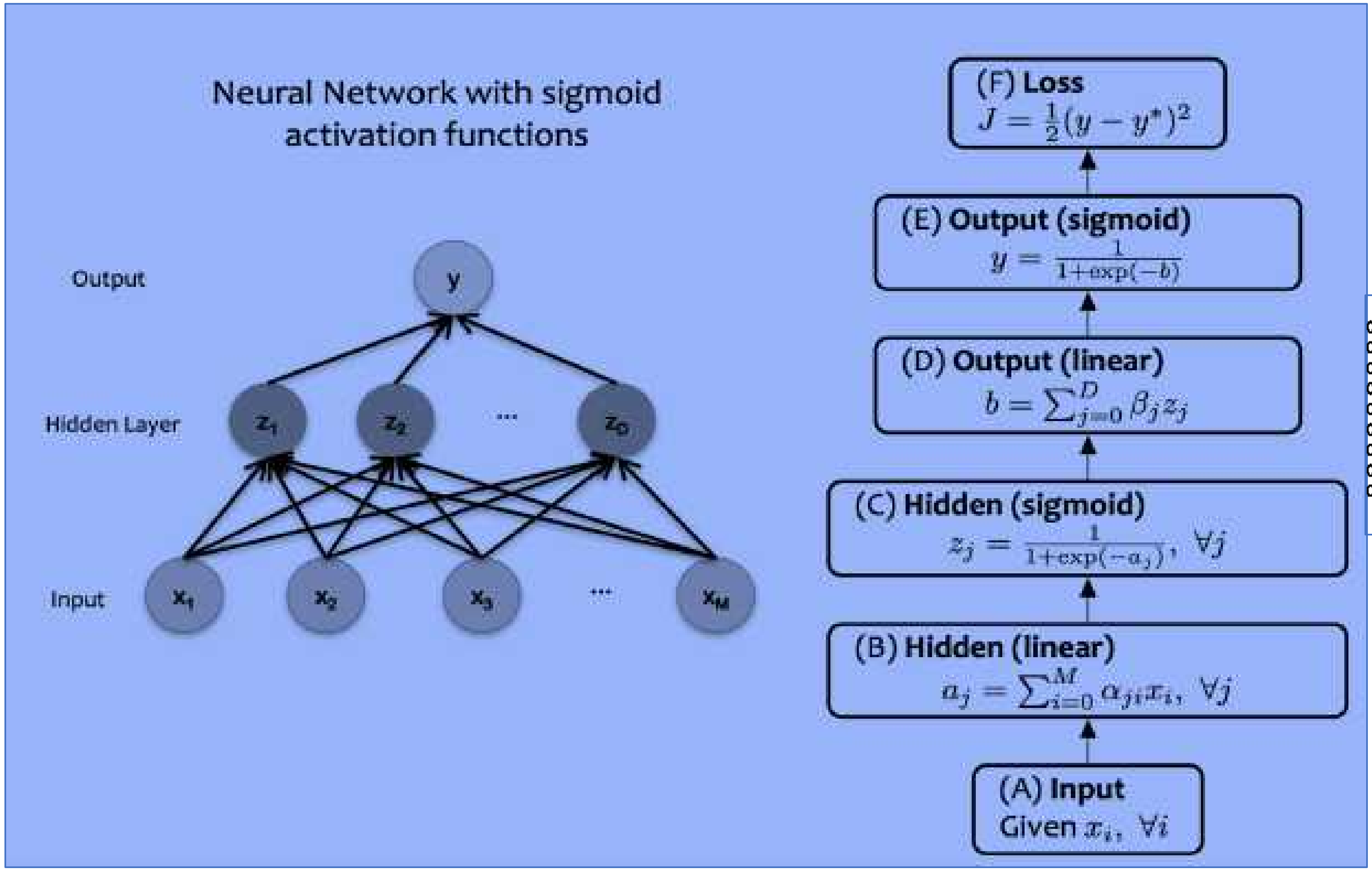
Empirical answer:

- Before 2006: "Deep networks (e.g. 3 or more hidden layers) are too hard to train"
- After 2006: "Deep networks are easier to train than shallow networks (e.g. 2 or fewer layers) for many problems"

Big caveat: You need to know and use the right tricks.



NN with Sigmoid Hidden Layer



(F) **Loss**
$J = \frac{1}{2}(y - y^*)^2$

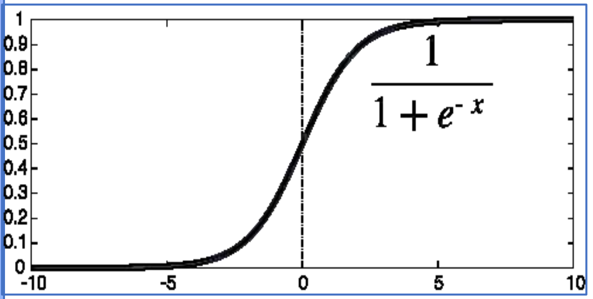
(E) **Output (sigmoid)**
$y = \frac{1}{1+\exp(-b)}$

(D) **Output (linear)**
$b = \sum_{j=0}^{D} \beta_j z_j$

(C) **Hidden (sigmoid)**
$z_j = \frac{1}{1+\exp(-a_j)}, \ \forall j$

(B) **Hidden (linear)**
$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \ \forall j$

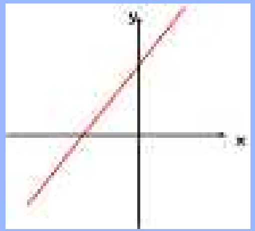
(A) **Input**
Given $x_i, \ \forall i$

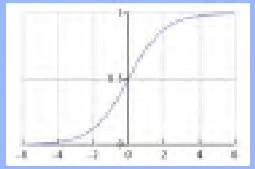


Loss (Objective) Functions

Objective Functions for NNs

1. Quadratic Loss:
   - the same objective as Linear Regression
   - i.e. mean squared error
2. Cross-Entropy:
   - the same objective as Logistic Regression
   - i.e. negative log likelihood
   - This requires probabilities, so we add an additional "softmax" layer at the end of our network





	Forward	Backward
Quadratic	$J = \frac{1}{2}(y - y^*)^2$	$\frac{dJ}{dy} = y - y^*$
Cross Entropy	$J = y^* \log(y) + (1 - y^*) \log(1 - y)$	$\frac{dJ}{dy} = y^* \frac{1}{y} + (1 - y^*) \frac{1}{y - 1}$

Total-Sum-Squared-Error (TSSE)

$$TSSE = \frac{1}{2} \sum_{patterns} \sum_{outputs} (desired - actual)^2$$

Root-Mean-Squared-Error (RMSE)

$$RMSE = \sqrt{\frac{2 * TSSE}{\# patterns * \# outputs}}$$

Mean Squared Error (MSE)

$$Loss = \frac{1}{n} \sum_{i=1}^{n} (f_i - y_i)^2$$

Mean Absolute Error (MAE)

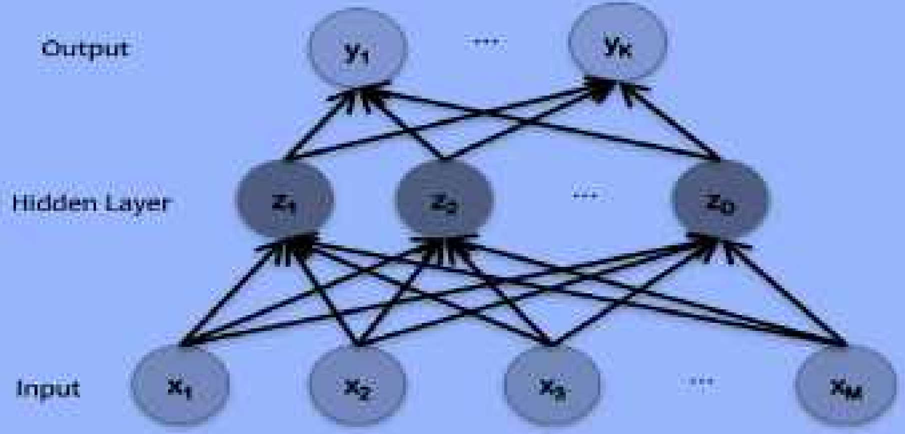
$$Loss = \frac{1}{n} \sum_{i=1}^{n} |f_i - y_i|$$

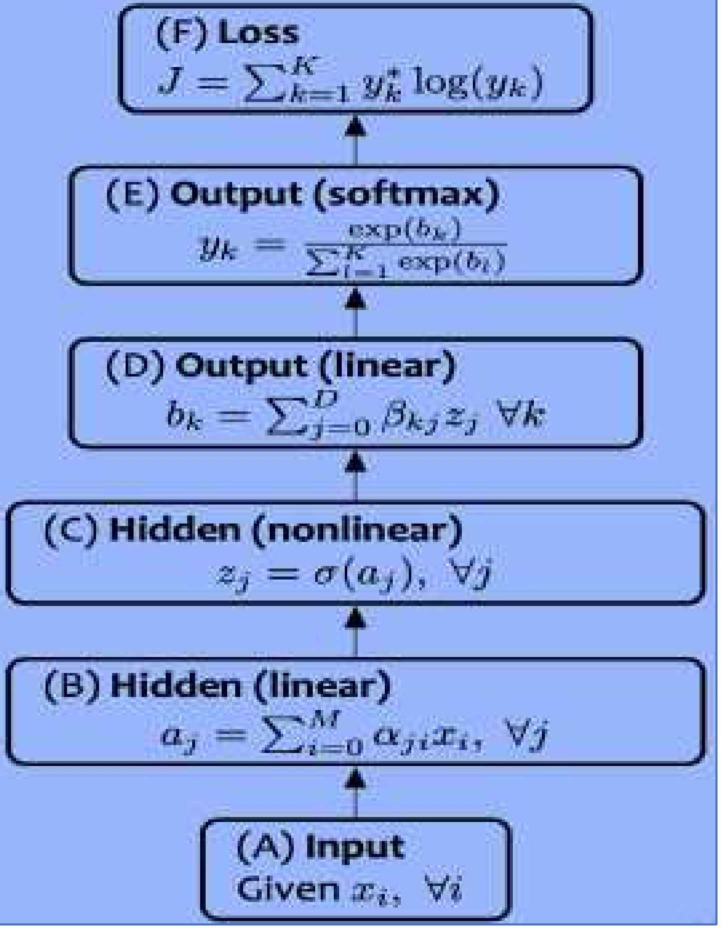
Multiple Classification with Softmax (Probability)

Multi-Class Output

Softmax:

$$y_k = \frac{\exp(b_k)}{\sum_{l=1}^{K} \exp(b_l)}$$





(F) Loss
$J = \sum_{k=1}^{K} y_k^* \log(y_k)$

(E) Output (softmax)
$y_k = \frac{\exp(b_k)}{\sum_{l=1}^{K} \exp(b_l)}$

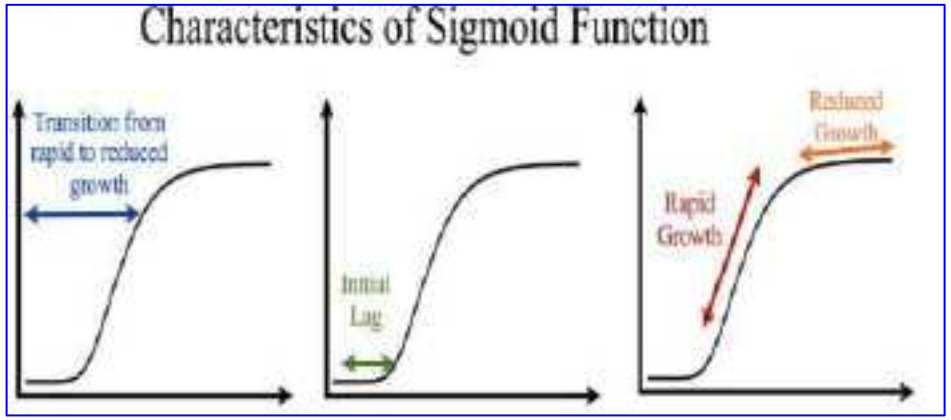
(D) Output (linear)
$b_k = \sum_{j=0}^{D} \beta_{kj} z_j \ \forall k$

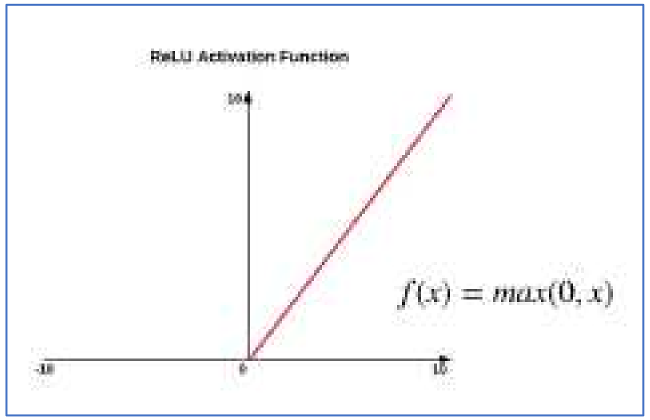
(C) Hidden (nonlinear)
$z_j = \sigma(a_j), \ \forall j$

(B) Hidden (linear)
$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \ \forall j$

(A) Input
Given $x_i, \ \forall i$

Sigmoid and ReLU


Characteristics of Sigmoid Function
Transition from rapid to reduced growth
Initial Lag
Rapid Growth
Reduced Growth



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

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

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

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

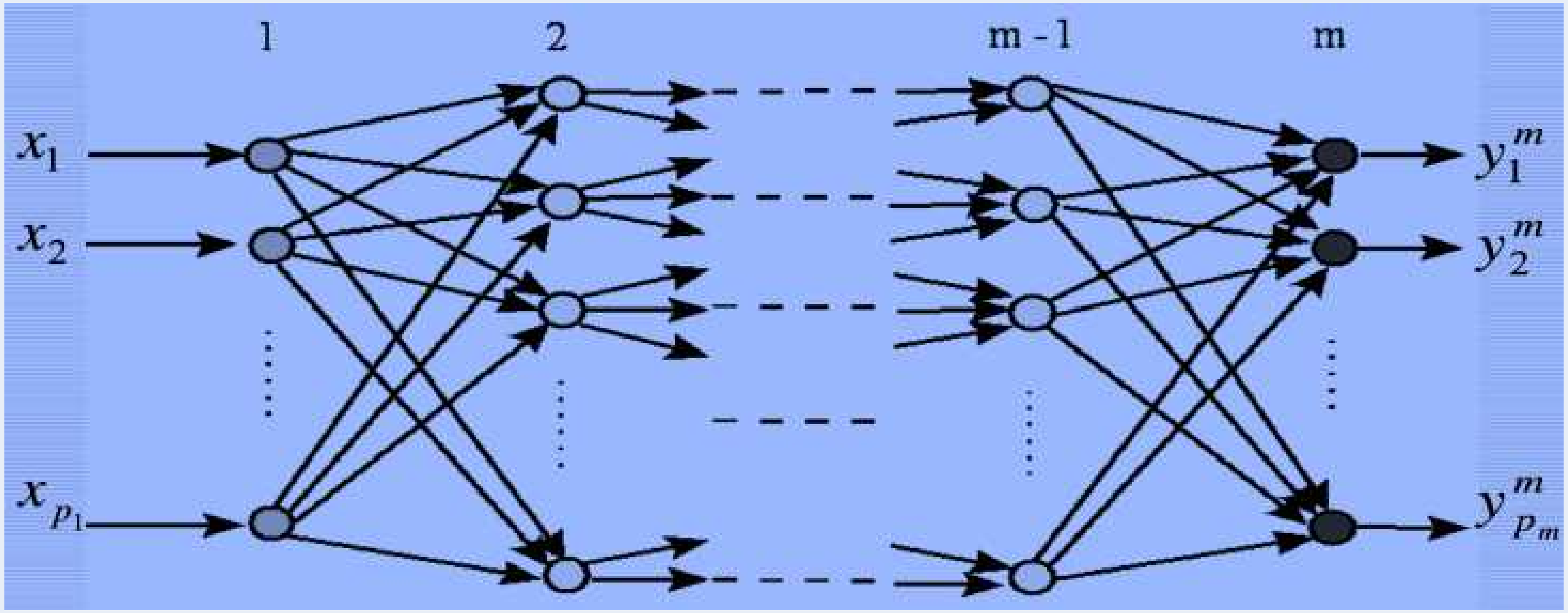
- 6 t=t+1, repeat 2-5 until the error rate for the N samples reaches the set target or stops decreasing .

BP N Training Samples, p1 Input x

$$x_i=[x_{i1},x_{i2},\ldots,x_{ip1}]^T$$

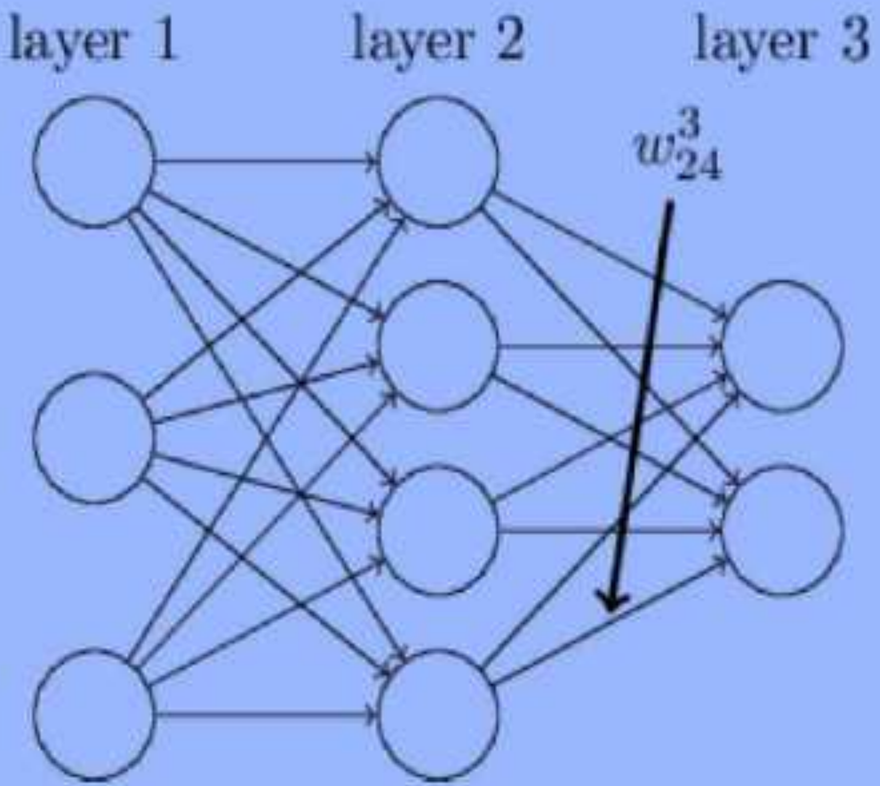
$$i=1,2,\ldots,N$$

Structure for BP : m Layers



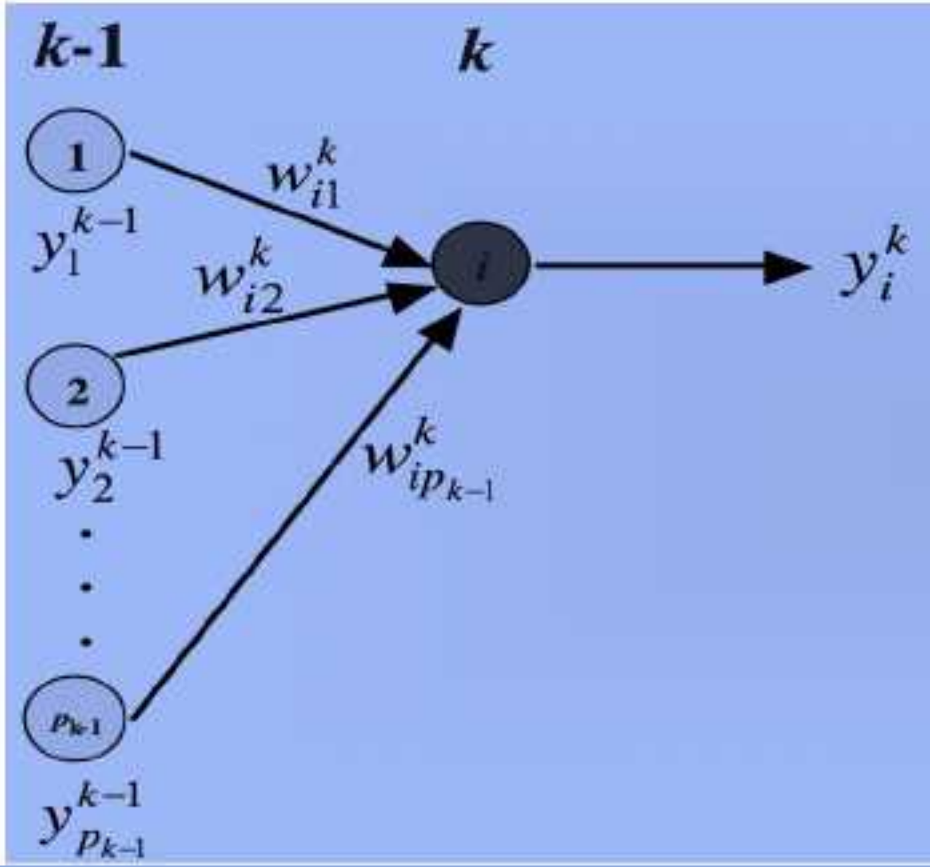
$X = [x_1 \quad x_2 \quad \Lambda \quad x_{p_1}]^T$ $\qquad$ $Y = [y_1^m \quad y_2^m \quad \Lambda \quad y_{p_m}^m]^T$

Weight W_{jk}



w^l_{jk} is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer

Layer k Node i in BP



$$u_i^k = \sum_{j=1}^{p_{k-1}} w_{ij}^k y_j^{k-1} - \theta_i = \sum_{j=0}^{p_{k-1}} w_{ij}^k y_j^{k-1}$$

$$(y_0^{k-1} = \theta_i, w_{i0}^k = -1)$$

$$y_i^k = f(u_i^k) = \frac{1}{1+e^{-u_i^k}}$$

$$i = 1,2,\ldots,p_k$$

$$k = 1,2,\ldots,m$$

Kth layer, we have P_K inputs,

u is the SUM before transfer,

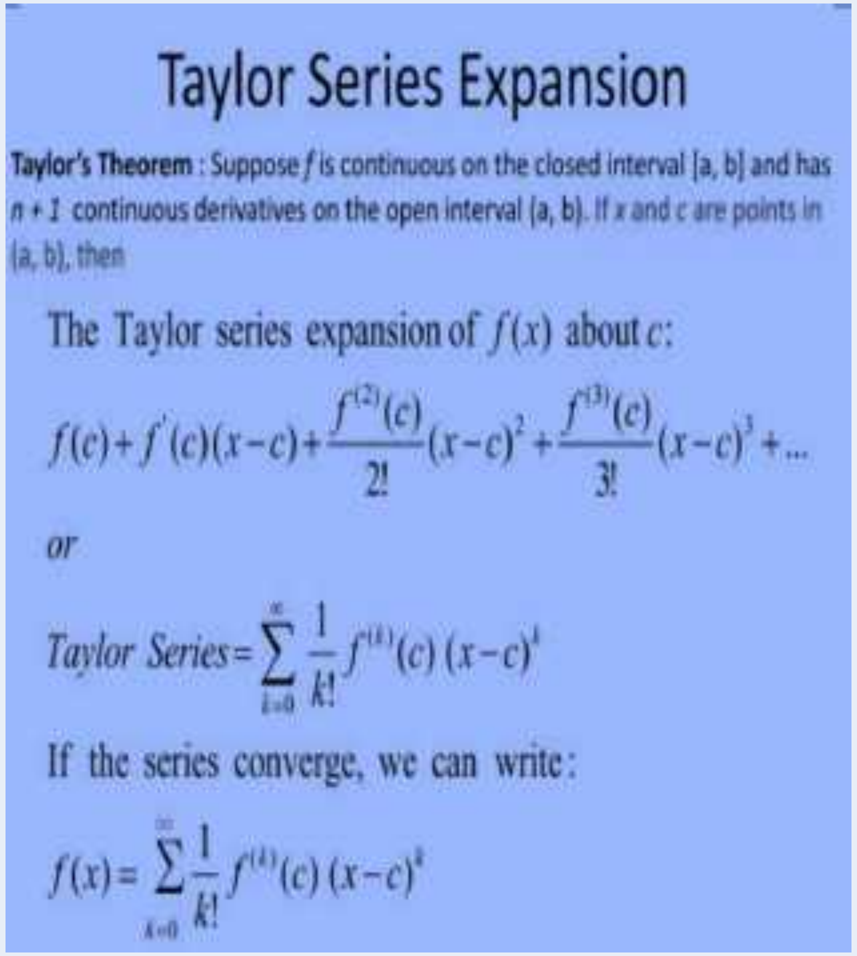
BP error function J and node function u, y

$$J = \frac{1}{2}\sum_{j=1}^{p_m}(y_j^m - y_j)^2$$

$$u_i^k = \sum_j w_{ij}^{k-1} y_j^{k-1} \qquad i = 1,2,\ldots,p_k$$

$$y_i^k = f_k(u_i^k) \qquad k = 1,2,\ldots,m$$

Weight Update: Taylor 1st Expansion



Taylor Series Expansion

Taylor's Theorem : Suppose f is continuous on the closed interval [a, b] and has $n+1$ continuous derivatives on the open interval (a, b). If x and c are points in (a, b), then

The Taylor series expansion of $f(x)$ about c:

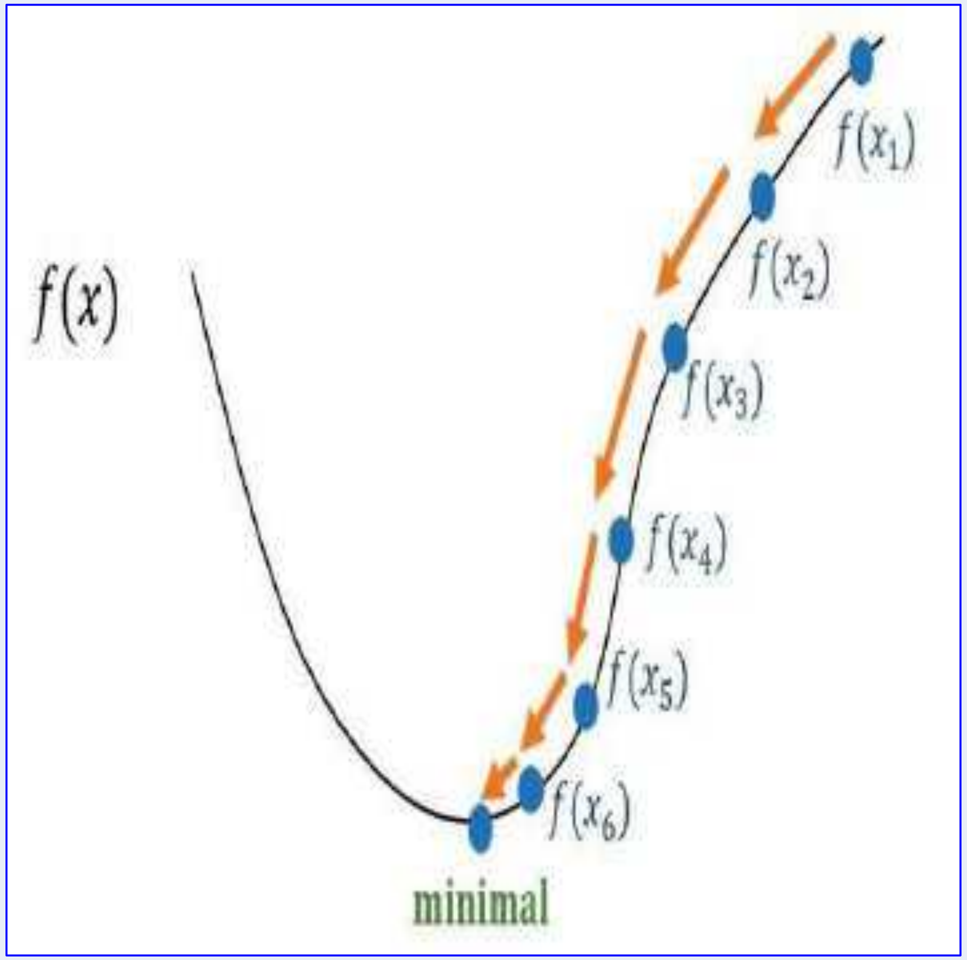
$$f(c)+f'(c)(x-c)+\frac{f^{(2)}(c)}{2!}(x-c)^2+\frac{f^{(3)}(c)}{3!}(x-c)^3+\ldots$$

or

$$\text{Taylor Series}=\sum_{k=0}^{\infty}\frac{1}{k!}f^{(k)}(c)\,(x-c)^k$$

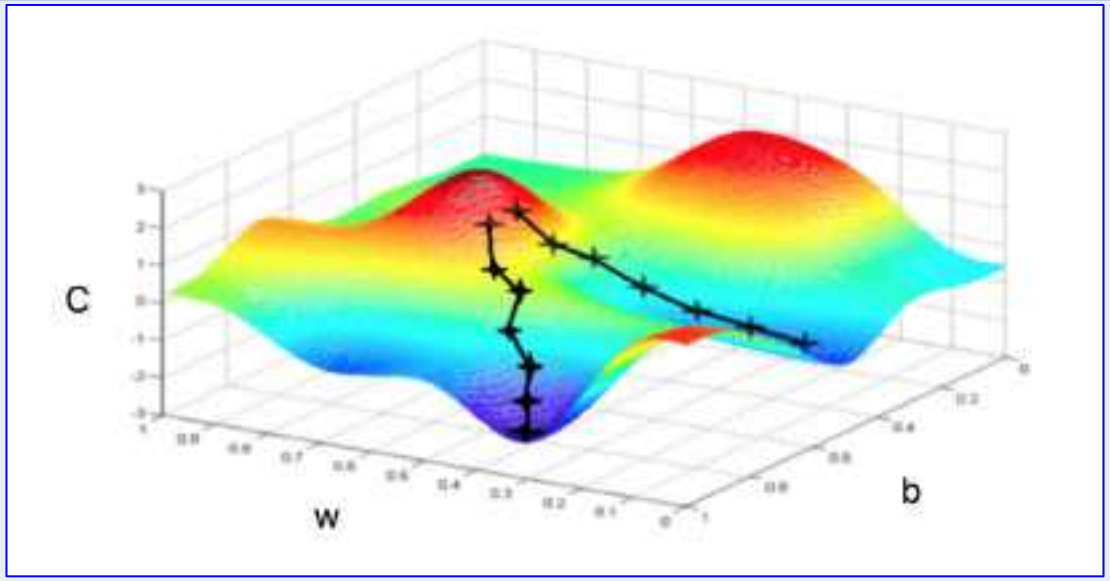
If the series converge, we can write:

$$f(x)=\sum_{k=0}^{\infty}\frac{1}{k!}f^{(k)}(c)\,(x-c)^k$$



BP Learning Rule: Weight Delta Calculation

$$\Delta w_{ij}^{k-1} = -\varepsilon \frac{\partial J}{\partial w_{ij}^{k-1}} \qquad j = 1, 2, \ldots, p_{k-1}$$



General: Update the weight of jth node at level k-1 to ith node at level k

$$\frac{\partial J}{\partial w_{ij}^{k-1}} = \frac{\partial J}{\partial u_i^k}\frac{\partial u_i^k}{\partial w_{ij}^{k-1}} = \frac{\partial J}{\partial u_i^k}\frac{\partial}{\partial w_{ij}^{k-1}}\left(\sum_j w_{ij}^{k-1} y_j^{k-1}\right) = \frac{\partial J}{\partial u_i^k} y_j^{k-1}$$

For all kth level calculation, for ith node at layer k, we define d^k

$$d_i^k = \frac{\partial J}{\partial u_i^k} = \frac{\partial J}{\partial y_i^k}\frac{\partial y_i^k}{\partial u_i^k} = \frac{\partial J}{\partial y_i^k} f_k'(u_i^k)$$

d Calculation for Output Layer Nodes

$$\frac{\partial J}{\partial y_i^k} = \frac{\partial J}{\partial y_i^m} = y_i^m - y_i$$

$$\boxed{d_i^m = (y_i^m - y_i) f_m'(u_i^m)}$$

d Calculation for Hidden Layer Nodes

$$\frac{\partial J}{\partial y_i^k} = \sum_l \frac{\partial J}{\partial u_l^{k+1}} \frac{\partial u_l^{k+1}}{\partial y_i^k} = \sum_l d_l^{k+1} w_{li}^k$$

$$d_i^k = \frac{\partial J}{\partial y_i^k} f_k'(u_i^k) = f_k'(u_i^k) \sum_l d_l^{k+1} w_{li}^k$$

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

Weight Delta Calculation Summary: for Both Output and Hidden Layers

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

$$d_i^{m} = (y_i^{m} - y_i) f'_m (u_i^{m})$$

$$d_i^{k} = f'_k (u_i^{k}) \sum_l d_l^{k+1} w_{li}^{k}$$

Delta when Transfer Function is Sigmoid

$$y_i^k = \frac{1}{1+e^{-u_i^k}}$$

$\sigma(x)$ is the sigmoid function

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

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

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

$$d_i^m = (y_i^m - y_i) f_m'(u_i^m)$$

$$d_i^k = f_k'(u_i^k) \sum_l d_l^{k+1} w_{li}^k$$

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

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

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

Recall: 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

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

Recall: 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 .

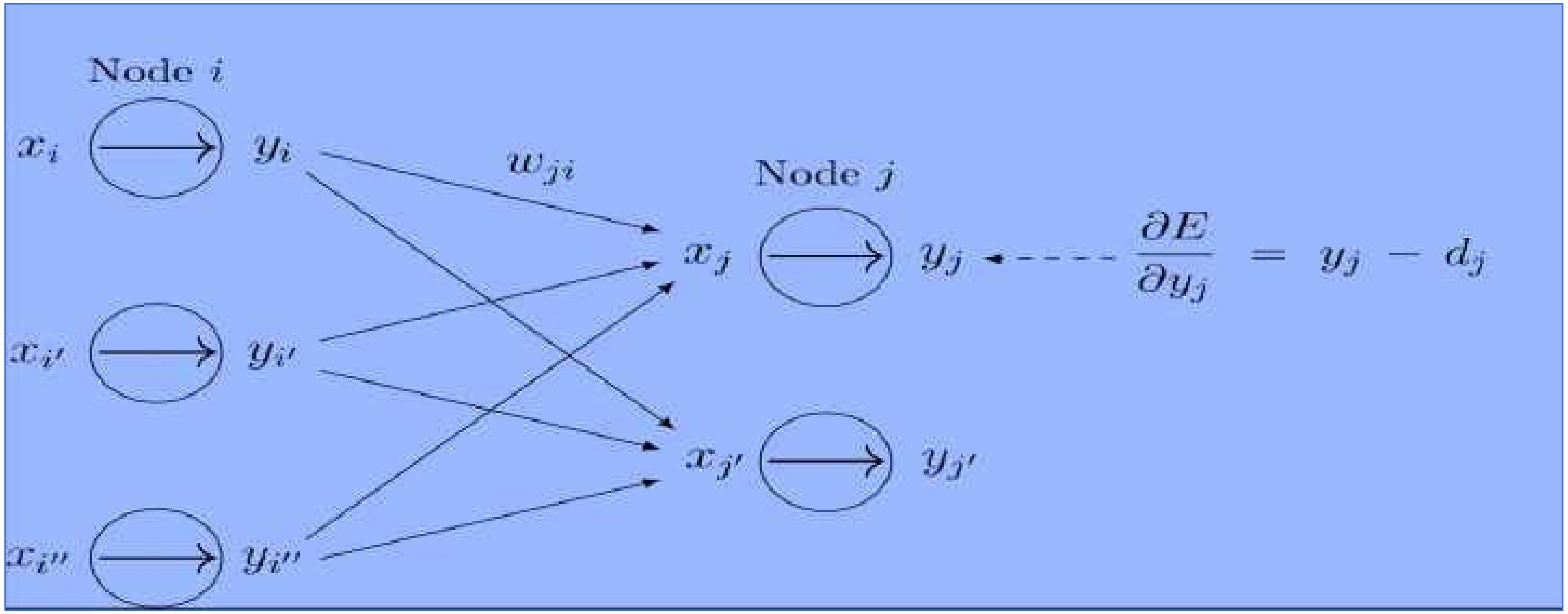
BP Algorithm Code Flowchart

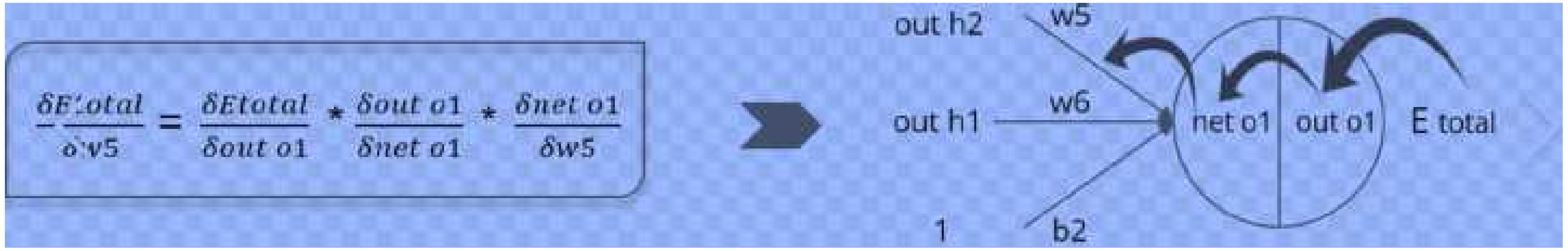
- Initialize weights (typically random)
- Keep doing epochs
  - For each example in training set do
    - forward pass to compute
      - O = neural-net-output(network, example)
      - miss = (T-O) at each output unit
    - backward pass to calculate deltas (d) to weights
    - update all weights
  - end
- until tuning set error stops improving

Forward pass

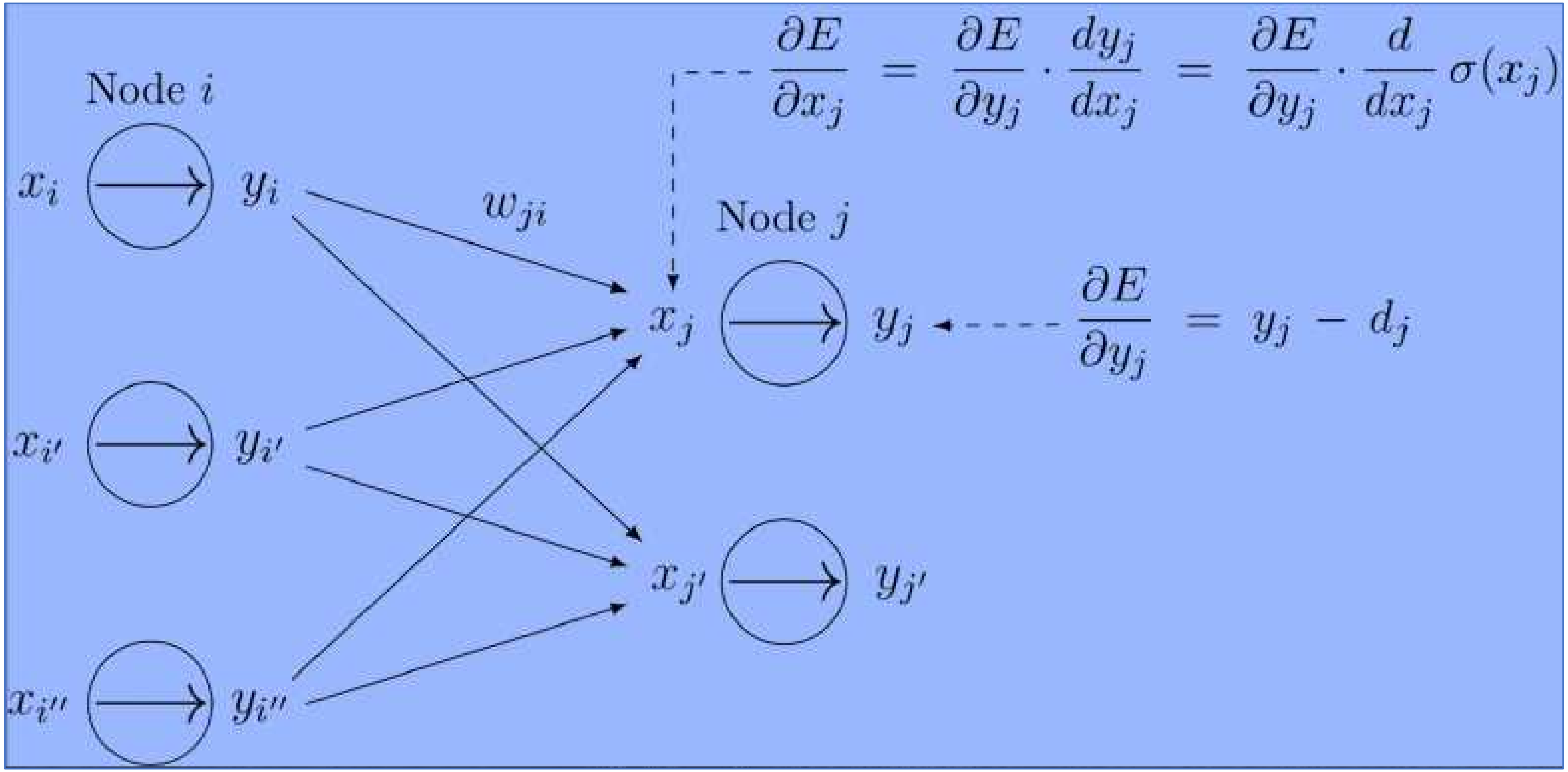
Backward pass

BP Chain Rule Flow




$\frac{\delta Etotal}{\delta w5} = \frac{\delta Etotal}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w5}$
out h2
w5
out h1
w6
1
b2
net o1
out o1
E total

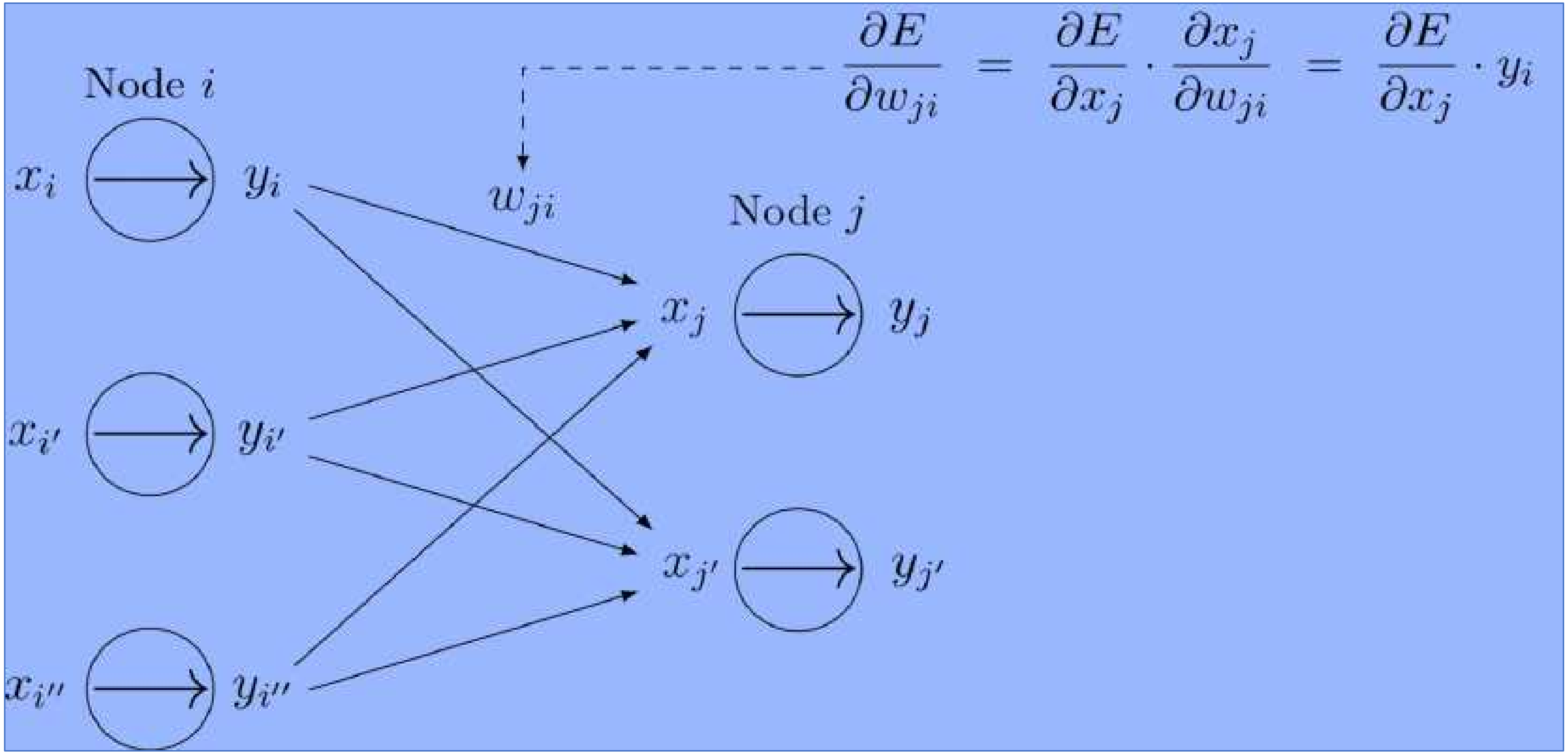
BP Chain Rule Flow



$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{dx_j} = \frac{\partial E}{\partial y_j} \cdot \frac{d}{dx_j}\sigma(x_j)$$

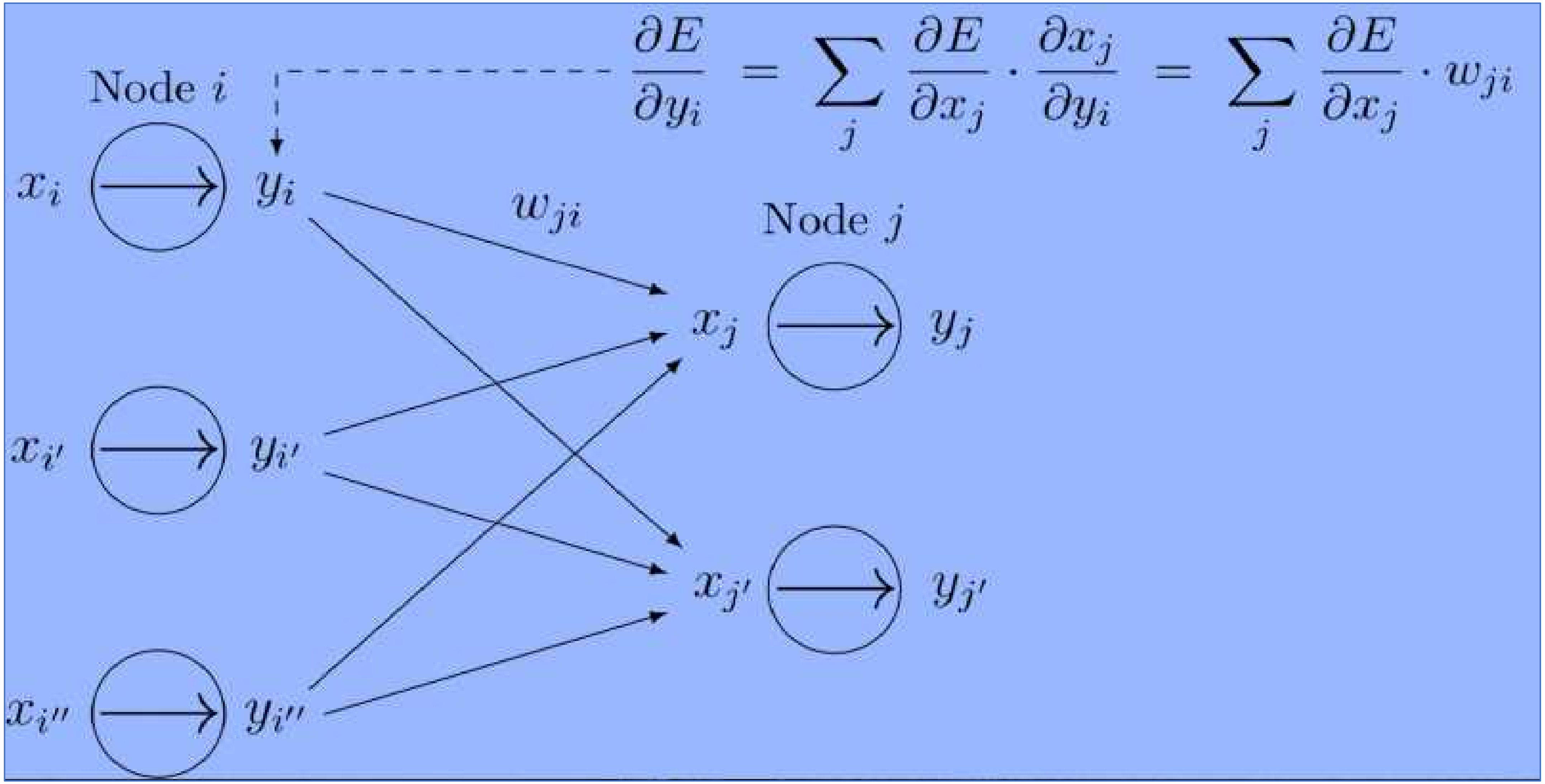
$$\frac{\partial E}{\partial y_j} = y_j - d_j$$

BP Chain Rule Flow



$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ji}} = \frac{\partial E}{\partial x_j} \cdot y_i$$

BP Chain Rule Flow



$$\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial x_j} \cdot \frac{\partial x_j}{\partial y_i} = \sum_j \frac{\partial E}{\partial x_j} \cdot w_{ji}$$

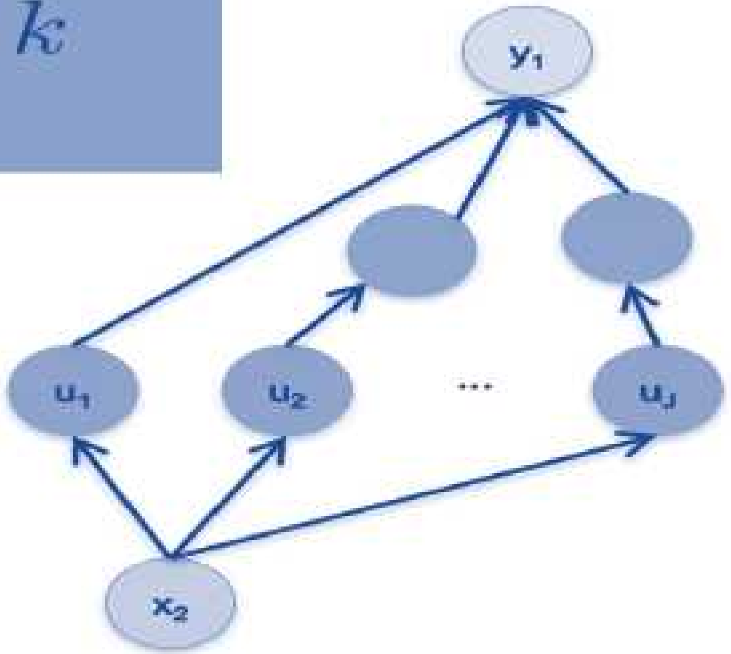
Chain Rule in General

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

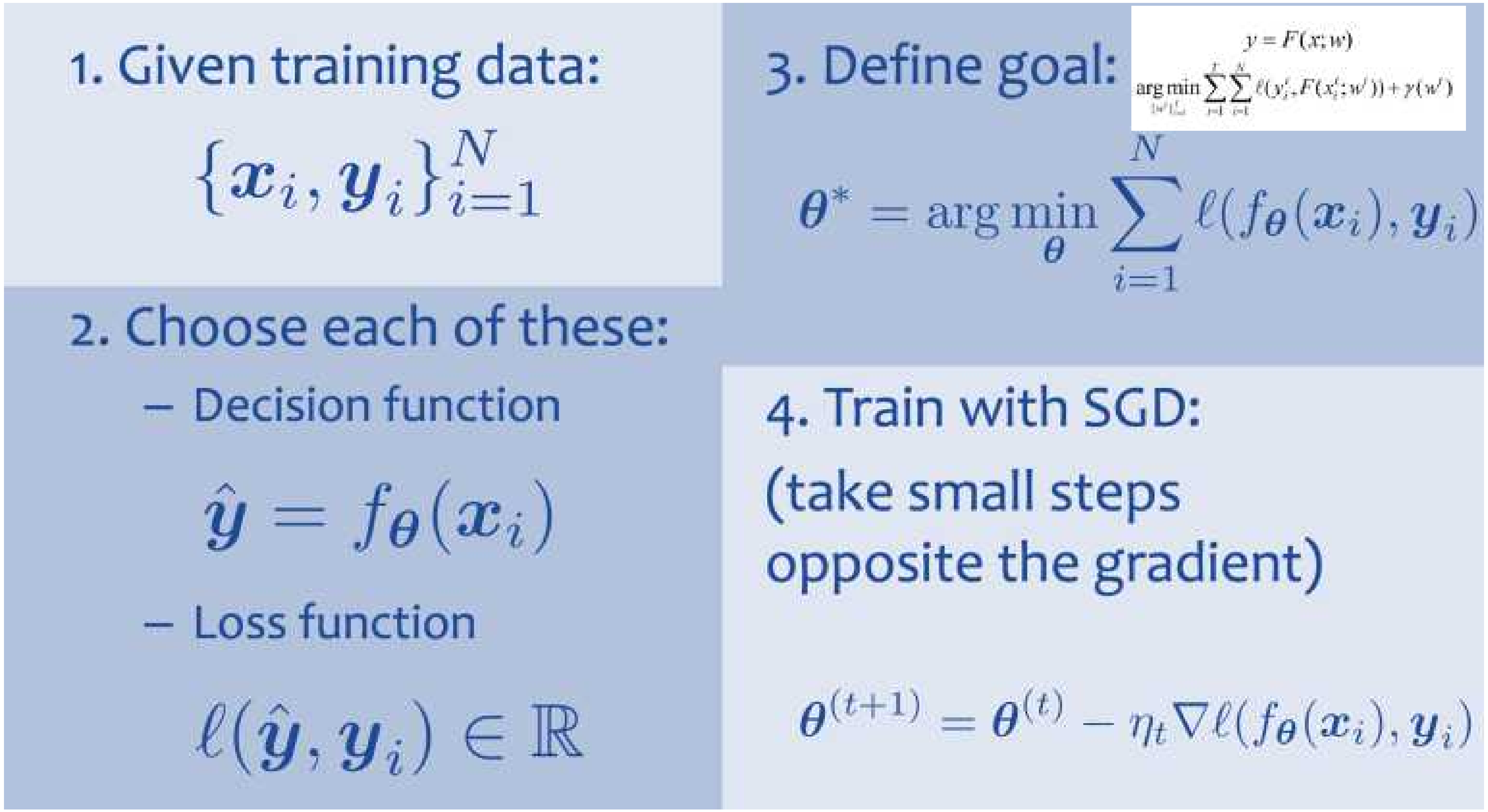
Chain Rule:

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

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:

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

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

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

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

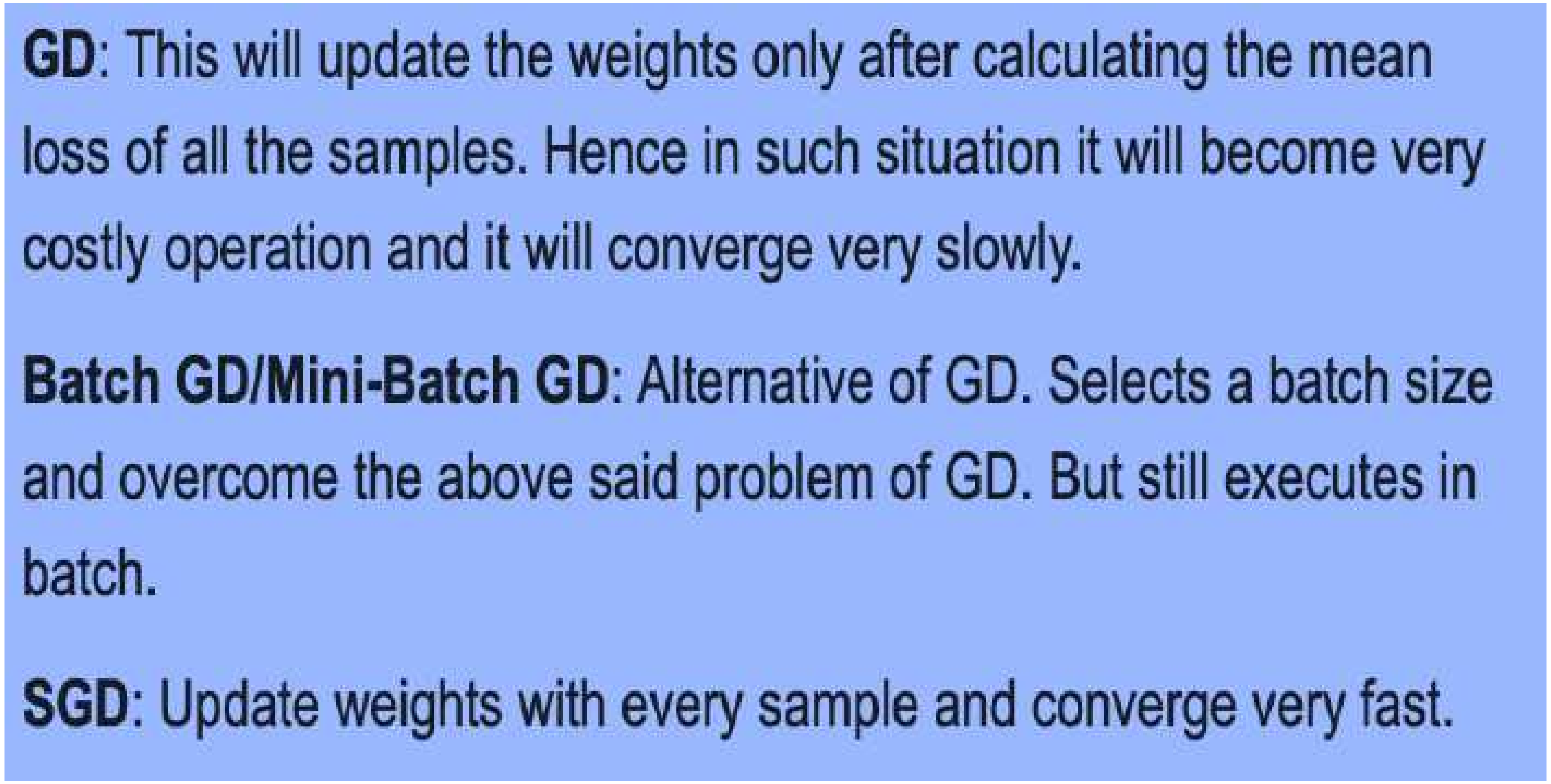
There are three core differences for the updating rule between BP and Perceptron :

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

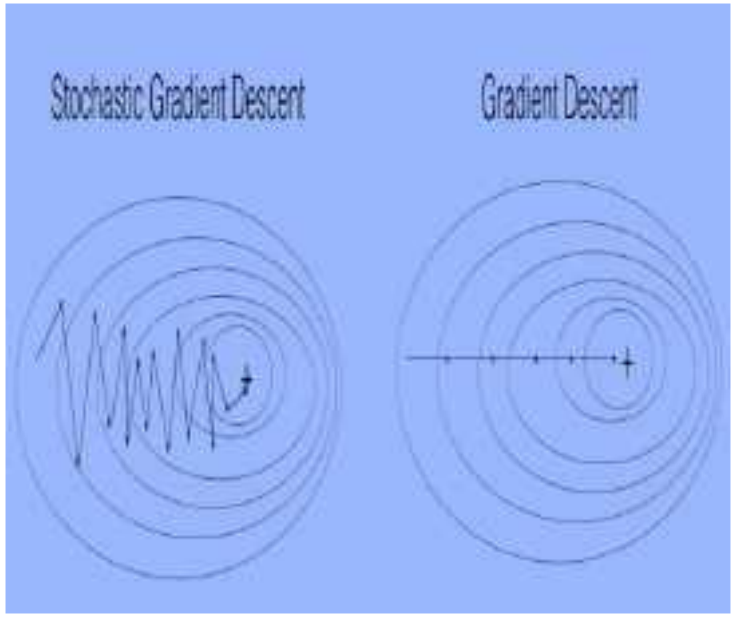


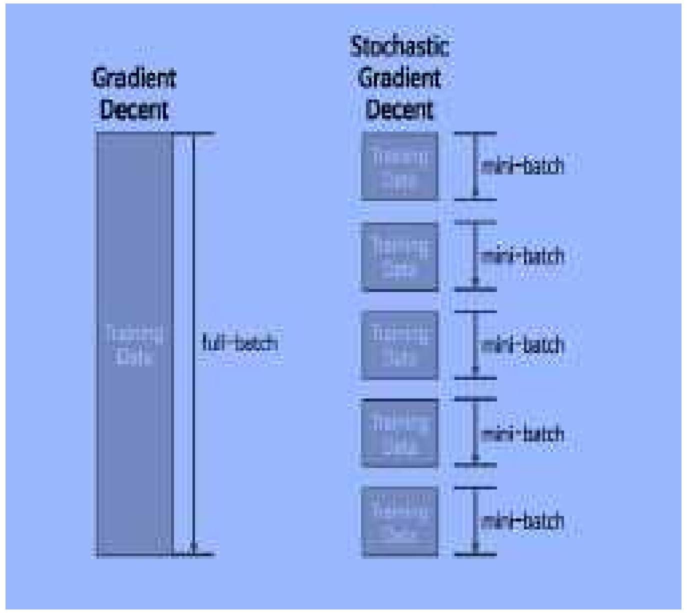
GD: This will update the weights only after calculating the mean loss of all the samples. Hence in such situation it will become very costly operation and it will converge very slowly.

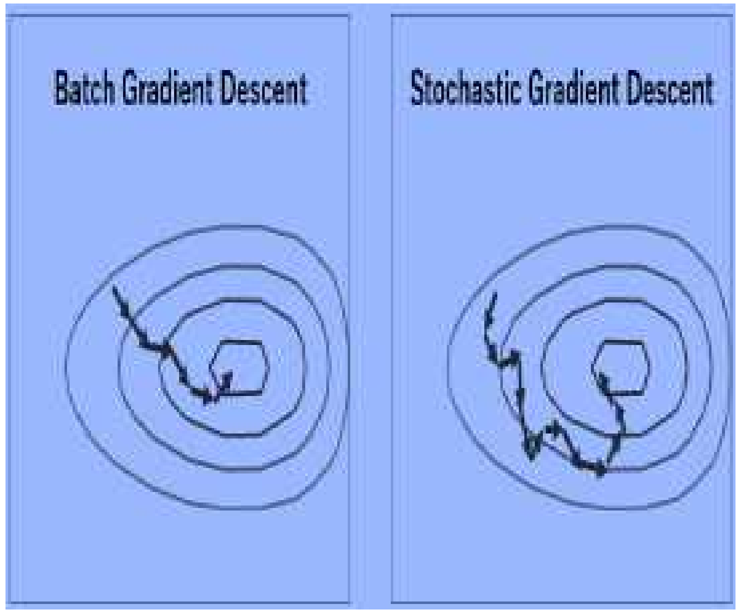
Batch GD/Mini-Batch GD: Alternative of GD. Selects a batch size and overcome the above said problem of GD. But still executes in batch.

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

Gradient Descent Vs Stochastic Gradient Descent

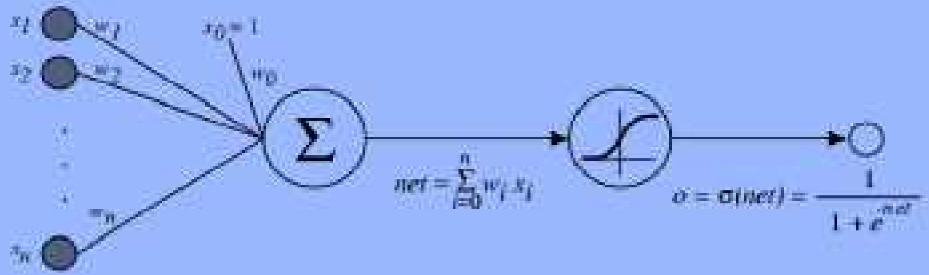






Any Question?

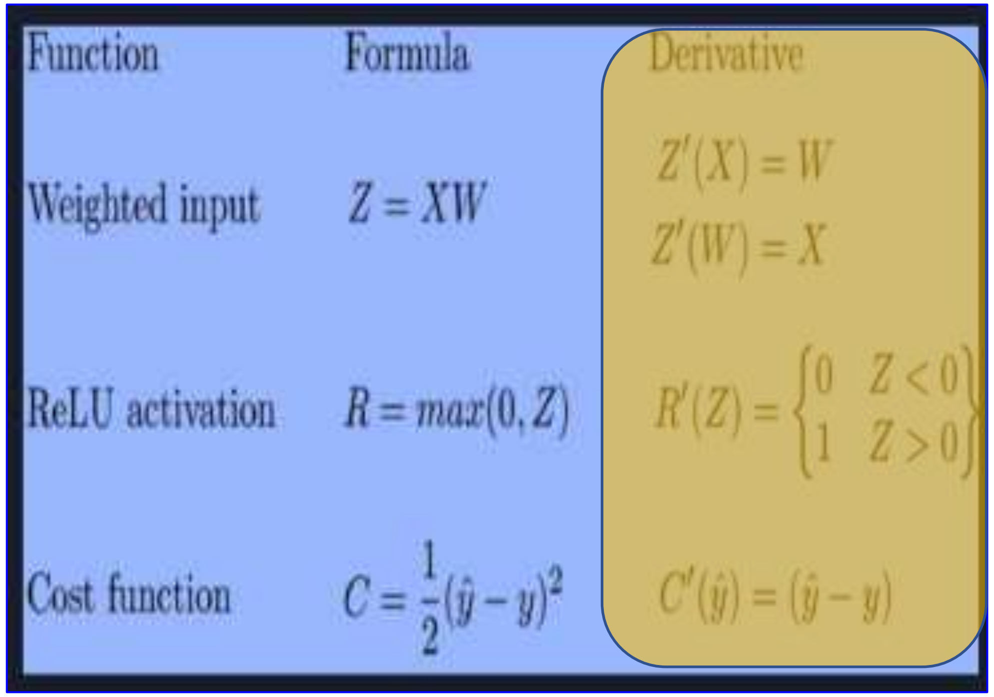
Homework: Prove the derivative formula (in yellow box) of Sigmoid and ReLU Transfer Functions


x_1
x_2
x_n
w_1
w_2
w_n
$x_0 = 1$
w_0
Σ
$net = \sum_{i=0}^{n} w_i x_i$
$o = \sigma(net) = \frac{1}{1+e^{-net}}$

$\sigma(x)$ is the sigmoid function

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

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



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

CS 103 -11
BP

Jimmy Liu 刘江

2020-11-27