深度学习和神经网络基础

Machine Learning

Representation

Objective

Optimization

Traditional Machine Learning

- 人工特征工程+分类器

特征抽取
（Segmentation、PCA、Shape）

分类器
（SVM、NB、Maximum Entropy、CRF）

大数据下的机器学习

Deep Learning

- 自动学习多尺度的特征表示

Output
Output
Output
Mapping from features
Output
Mapping from features
Mapping from features
Additional layers of more abstract features
Hand-designed program
Hand-designed features
Features
Simple features
Input
Input
Input
Input
Rule-based systems
Classic machine learning
Deep learning
Representation learning

History

1958 Perceptron
1974 Backpropagation
Convolution Neural Networks for Handwritten Recognition
1998
Google Brain Project on 16k Cores
2012
awkward silence (AI Winter)
1969
Perceptron criticized
1995
SVM reigns
2006
Restricted Boltzmann Machine
2012
AlexNet wins ImageNet
IMAGENET

Deep Learning Big Guys

Yann LeCun
NYU & FaceBook

Geoff Hinton
Tornoto U & Google

Yoshua Bengio
Montreal U

Andrew Ng
Stanford & Baidu

Juergen Schmidhuber
Swiss AI Lab & NNAISENSE

Richard Socher
Founder, CEO, CTO of MetaMind

Deep Learning Wins on Speech

Deep Learning Wins on Image

- ICDAR 2009 French Connected Handwriting Competition based on data from the RIMES campaign
- ICDAR 2009 Handwritten Farsi/Arabic Character Recognition Competition
- ICDAR 2009 Arabic Connected Handwriting Competition of Univ. Braunschweig
- Online German Traffic Sign Recognition Contest
- ICDAR 2011 offline Chinese Handwriting Competition
- IJCNN 2011 on-site Traffic Sign Recognition Competition
- ISBI 2012 Brain Image Segmentation Challenge
- ICPR 2012 Contest on Mitosis Detection in Breast Cancer Histological Images
- MICCAI 2013 Grand Challenge on Mitosis Detection

手写体识别

- MNIST（手写体识别）任务
  - 60,000训练
  - 10,000测试
  - 28*28图像
- Convolutional NN(2013):99.79%

ImageNet

- 1,000类别，1,300,000image
  - 2010: 47% error (top-1) and 25% error for (top-5)
  - 2012: Less that 40% error (top-1) and less than 20% (top-5) with a very deep neural net (Krizhevsky et. al. 2012) AlexNet
  - 2014: GoogleNet
  - 2015: ResNet, Microsoft won 2015 LSVRC challenge
    - 152 layers (CNN), 19.38% error (top-1), 3.57% error (top-5) !!

网络加深
VGG16
VGG19
MSRANet
二者结合，
网络加深+收敛加速
ResNet
Inception
ResNet
增强卷积模块功能
NIN
GoogLeNet
Inception V3
Inception V4
Neocognitron
LeCun1989
LeNet
历史突破
AlexNet
Dropout
ReLU
GPU+Bigdata
从分类任务到检测任务
SPP-Net
R-CNN
Fast R-CNN
Faster R-CNN
增加新的功能单元
Inception V2
FCN
STNet
CNN+RNN/LSTM

GoogleNet

- 2014 ILSVRC winner
- 22层
- 错误率：6.7%

Res

- 2015 ILSVRC winner
- 152层
- 错误率：3.57%

152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10

Deep Learning Wins on NLP

关系分类

[叶莉]$_{e1}$ 1981 年 在 [上海]$_{e2}$ 一个 普通 工人 家庭 出生

出生地 (e1,e2)

Convolution Neural Network

Classifier	Feature Sets	F1
SVM	POS, stemming, syntactic patterns	60.1
SVM	word pair, words in between	72.5
SVM	POS, stemming, syntactic patterns, WordNet	74.8
MaxEnt	POS, morphological, noun compound, thesauri, Google n-grams, WordNet	77.6
SVM	POS, prefixes, morphological, WordNet, dependency parse, Levin classed, ProBank, FrameNet, NomLex-Plus, Google n-gram, paraphrases, TextRunner	82.2
RNN	- POS, NER, WordNet, syntactic tree	74.8 77.6
MVRNN	- POS, NER, WordNet, syntactic tree	79.1 82.4
Proposed	word pair, WordNet	**82.7**

CNN: Zeng et al. Coling 2014

Classifier	Feature set	F_1
SVM	POS, WordNet, prefixes and other morphological features, depdency parse, Levin classes, PropBank, FanmeNet, NomLex-Plus, Google n-gram, paraphrases, TextRunner	82.2
RNN	Word embeddings Word embeddings, POS, NER, WordNet	74.8 77.6
MVRNN	Word embeddings Word embeddings, POS, NER, WordNet	79.1 82.4
CNN	Word embeddings Word embeddings, word position embeddings, WordNet	69.7 82.7
Chain CNN	Word embeddings, POS, NER, WordNet	82.7
FCM	Word embeddings Word embeddings, depedency parsing, NER	80.6 83.0
CR-CNN	Word embeddings Word embeddings, position embeddings	82.8† 82.7
	Word embeddings, position embeddings	**84.1**†
SDP-LSTM	Word embeddings Word embeddings, POS embeddings, WordNet embeddings, grammar relation embeddings	82.4 **83.7**

LSTM: Xu et al. EMNLP 2015

机器翻译

System	Training	MT06	MT02	MT03	MT04	MT05	MT08
MOSES	MERT	32.74	32.49	32.40	33.38	30.20	25.28
RNNSEARCH	MLE	30.70	35.13	33.73	34.58	31.76	23.57
	MRT	37.34	40.36	40.93	41.37	38.81	29.23

AlphaGo

AlphaGo

Deep Learning

Big Players

facebook

Microsoft

YAHOO!

Google

IBM

NVIDIA

Baidu 百度

目录

- 感知机 (SVM)
- 前馈神经网络

感知机 Perceptron

感知机 Perception

- Invented by Frank Rosenblatt

感知器是对生物神经细胞的简单数学模拟,是最简单的人工神经网络, 只有一个神经元。 感知器也可以看出是线性分类器的一个经典学习算法。

细胞体(Soma)中的神经细胞膜上有各种受体和离子通道,胞膜的受体可与相应的化学物质神经递质结合,引起离子通透性及膜内外电位差发生改变,产生相应的生理活动:兴奋或抑制。 细胞突起是由细胞体延伸出来的细长部分,又可分为**树突和轴突**。

- 树突(Dendrite)可以接受刺激并将兴奋传入细胞体。每个神经元可以有一或多个树突。
- 轴突 (Axons) 可以把兴奋从胞体传送到另一个神经元或其他组织。 每个神经元只有一个轴突。

抑制与兴奋

神经 细胞的状态取决于从其它的神经细胞收到的输入信号量,及突触的强度(抑制或加强)。当信号量总和超过了某个阈值时,细胞体就会兴奋,产生电脉冲。电脉冲沿着轴突并通过突触传递到其它神经元。

感知机

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

感知机

- 给定输入 $x=(x_1,x_2,x_3,\cdots x_n)$

$$y=f(x)=sign(w\bullet x+b)$$

$$sign(x)=\begin{cases}+1, & x\geq 0\\ -1, & x<0\end{cases}$$

$$y=f(x)=sign(w\bullet x)$$

$$x=(b,x_1,x_2,x_3,\cdots x_n)$$

$$w=\{1,w_1,w_2,w_3,\cdots w_n\}$$

感知器几何解释

$$w \cdot x + b = 0$$

感知机学习策略

- 损失函数
  - 1. 误分点的数目，与w, b 无关，因此不是连续可导的
  - 2. 误分点到超平面的距离

$$\frac{1}{||w||}|w \cdot x_e + b|$$

误分点 x_e: $-y_i(w \cdot x_i + b) > 0$

误分点到超平面的距离 $-\frac{1}{||w||}y_i(w \cdot x_\mathrm{i} + b)$

总误分点到超平面的距离 $-\frac{1}{||w||}\sum_{x_\mathrm{i} \in M} y_i(w \cdot x_\mathrm{i} + b)$

感知机学习策略

- 损失函数
  - M为误分点集合

$$L(w,b) = -\sum_{x_i \in M} y_i(w \cdot x_i + b)$$

优化

$$\min_{w,b} L(w,b) = -\sum_{x_i \in M} y_i(w \cdot x_i + b)$$

训练

- 梯度下降法

$$\frac{\partial L(w,b)}{\partial w} = -\sum_{x_{\mathrm{i}} \in M} y_i x_{\mathrm{i}}$$

$$\frac{\partial L(w,b)}{\partial b} = -\sum_{x_{\mathrm{i}} \in M} y_i$$

$$w \leftarrow w + \lambda \sum_{x_{\mathrm{i}} \in M} y_i x_{\mathrm{i}}$$

$$b \leftarrow b + \lambda \sum_{x_{\mathrm{i}} \in M} y_i$$

训练算法

输入: 训练集: $(\mathbf{x}_i, y_i), i = 1, \cdots, N$, 迭代次数: T
输出: $\mathbf{w}$

```
初始化: w_0 = 0;
k = 0;
for t = 1 ··· T do
    for i = 1 ··· N do
        选取一个样本 (x_i, y_i), if w^T (y_i x_i) < 0 then
            w_{k+1} = w_k + y_i x_i, ;
            k = k + 1;
        end
    end
end
return w_k;
```

二次目标函数

$$L(w,b)=\frac{1}{2}\sum_{x_{\mathrm{i}}\in M}(y_i-(w\cdot x_{\mathrm{i}}+b))^2 \qquad \widetilde{y_i}=w\cdot x_{\mathrm{i}}+b$$

$$\frac{\partial L(w,b)}{\partial w}=-\sum_{x_{\mathrm{i}}\in M}(y_i-\widetilde{y_i})x_{\mathrm{i}} \qquad \frac{\partial L(w,b)}{\partial b}=-\sum_{x_{\mathrm{i}}\in M}(y_i-\widetilde{y_i})$$

$$w\leftarrow w+\lambda\sum_{x_{\mathrm{i}}\in M}(y_i-\widetilde{y_i})x_{\mathrm{i}} \qquad b\leftarrow b+\lambda\sum_{x_{\mathrm{i}}\in M}(y_i-\widetilde{y_i})$$

图示

size
1
domestication
size
2
domestication
size
3
domestication
size
4
domestication

感知器和SVM

Margins

线性可分支持向量机

- 假设特征空间上的训练数据集：

$$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$$

$$x_i\in X=R^n,\ y_i\in Y=\{+1,-1\},\ i=1,2,\cdots,N$$

点到超平面的距离

$$g(x) = w \cdot x + b$$

距离： $|w \cdot x + b|$

$$y(w \cdot x + b)$$

函数间隔和几何间隔

- 函数间隔
  - 样本点的函数间隔

$$\hat{\gamma}_i = y_i(w \cdot x_i + b)$$

  - 训练数据集的函数间隔

$$\hat{\gamma} = \min_{i=1,\cdots,N} \hat{\gamma}_i$$

函数间隔和几何间隔

- 几何间隔

$$\gamma_i = y_i\left(\frac{w}{\|w\|}\cdot x_i + \frac{b}{\|w\|}\right)$$

函数间隔和几何间隔

- 几何间隔
  - 对于给定的训练数据集T和超平面(w, b)

$$\gamma_i = y_i\left(\frac{w}{\|w\|}\cdot x_i + \frac{b}{\|w\|}\right)$$

  - 训练数据集的几何间隔

$$\gamma = \min_{i=1,\cdots,N} \gamma_i$$

即

$$\gamma_i = \frac{\hat{\gamma}_i}{\|w\|} \qquad \gamma = \frac{\hat{\gamma}}{\|w\|}$$

间隔最大化

- 最大间隔分类超平面

$$\max_{w,b} \quad \gamma$$

$$\text{s.t.} \quad y_i\left(\frac{w}{\|w\|}\cdot x_i+\frac{b}{\|w\|}\right)\geq\gamma, \quad i=1,2,\cdots,N$$

- 根据几何间隔和函数间隔的关系

$$\max_{w,b}\frac{\hat{\gamma}}{||w||}$$

$$s.t. \quad y_i(w\cdot x_i+b)\geq\hat{\gamma}, \qquad i=1,2,\ldots,N$$

- 可以取 $\hat{\gamma}=1$

- 最大化 $\frac{1}{\|w\|}$ 和最小化 $\frac{1}{2}\|w\|^2$ 等价

间隔最大化

- 线性可分支持向量机学习的最优化问题

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2$$
$$\text{s.t.} \quad y_i(w\cdot x_i+b)-1\geq 0,\quad i=1,2,\cdots,N$$

- 凸二次规划(Convex Quadratic Programming)

支持向量和Margins（边界）

- 在线性可分情况下，训练数据集的样本点中与分离超平面距离最近的样本点的实例称为支持向量(Support Vector)；
- 支持向量是使约束条件式等号成立的点，即

$$y_i(w \cdot x_i + b) - 1 = 0$$

- 正例： $H_1 : w \cdot x + b = 1$
- 负例： $H_2 : w \cdot x + b = -1$

线性支持向量机与软间隔最大化

- 训练数据中有一些特异点（outlier），不能满足函数间隔大于等于1的约束条件。
- 解决方法：对每个样本点 (x_i, y_i) 引进一个松弛变量 $\xi_i \geq 0$
- 使得函数间隔加上松弛变量大于等于1

约束条件变为： $y_i(w \cdot x_i + b) \geq 1 - \xi_i$

目标函数变为： $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i$

$C > 0$ 为惩罚参数

核函数

- 如果能用 R^n 中的一个超曲面将正负例正确分开，则称这个问题为非线性可分问题。

核函数

- 进行一个非线性变换，将非线性问题变换为线性问题，通过解变换后的线性问题的方法求解原来的非线性问题。

- 原空间： $\mathrm{X} \subset R^2, x = \left(x^{(1)}, x^{(2)}\right) \in \mathrm{X}$

- 新空间： $\mathrm{Z} \subset R^2, z = \left(z^{(1)}, z^{(2)}\right) \in \mathrm{Z}$ $\quad z = \phi(x) = \left(\left(x^{(1)}\right)^2, \left(x^{(2)}\right)^2\right)^T$

$$w_1\left(x^{(1)}\right)^2 + w_2\left(x^{(2)}\right)^2 + b = 0 \Rightarrow w_1 z^{(1)} + w_2 z^{(2)} + b = 0$$

- 用线性分类方法求解非线性分类问题分为两步：
  - 首先使用一个变换将原空间的数据映射到新空间；
  - 然后在新空间里用线性分类学习方法从训练数据中学习分类模型。

核函数

- 核函数定义：

设X是输入空间(欧氏空间 R^n 的子集或离散集合)，又设H为特征空间(希尔伯特空间)，如果存在一个从X到H的映射

$$\phi(x): \mathrm{X} \to \mathrm{H}$$

使得对所有 $x, z \in \mathrm{X}$

函数 $K(x,z)$ 满足条件 $K(x,z) = \phi(x) \cdot \phi(z)$

则称 $K(x,z)$ 为核函数，$\phi(x)$ 为映射函数，

式中 $\phi(x) \cdot \phi(z)$ 为 $\phi(x)$ 和 $\phi(z)$ 的内积

非线性支持向量机与核函数

- 假设输入空间是R^2，核函数是$K(x,z)=(x\cdot z)^2$，试找出其相关的特征空间H和映射$\phi(x): R^2 \to \mathrm{H}$

取特征空间 $\mathrm{H}=R^3$

记 $x=\left(x^{(1)},x^{(2)}\right)^T$ ， $z=\left(z^{(1)},z^{(2)}\right)^T$

可以取： $(x\cdot z)^2=\left(x^{(1)}z^{(1)}+x^{(2)}z^{(2)}\right)^2=\left(x^{(1)}z^{(1)}\right)^2+2x^{(1)}z^{(1)}x^{(2)}z^{(2)}+\left(x^{(2)}z^{(2)}\right)^2$

容易验证： $\phi(x)=\left(\left(x^{(1)}\right)^2,\sqrt{2}x^{(1)}x^{(2)},\left(x^{(2)}\right)^2\right)^T$

$$\phi(x)\cdot\phi(z)=(x\cdot z)^2=K(x,z)$$

常用核函数

1. 多项式核函数（Polynomial Kernel Function)

$$K(x,z)=(x\bullet z+1)^P$$

2. 高斯核函数（Gaussian Kernel Function)

$$K(x,z)=\exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$$

Toolkit

- SVM light : Joachims
  - http://svmlight.joachims.org
- LIBSVM:
  - http://www.csie.ntu.edu.tw/~cjlin/libsvm

The Problems of Perception

- In 1969, Minsky and Papert: XOR function

Non-linear Hidden Layer

$$\boldsymbol{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$

$$\boldsymbol{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix},$$

$$\boldsymbol{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix},$$

$$b = 0.$$

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = \boldsymbol{w}^\top \max\{0, \boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{c}\} + b$$

Example

- [0,0]
  - $\max(0, \left(\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}0\\-1\end{bmatrix}\right) = \begin{bmatrix}0\\0\end{bmatrix}$
  - $\begin{bmatrix}1\\-2\end{bmatrix}^{\mathrm{T}} \times \begin{bmatrix}0\\0\end{bmatrix} + 0 = 0$
- [1,0]
  - $\max(0, \left(\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}0\\-1\end{bmatrix}\right) = \begin{bmatrix}1\\0\end{bmatrix}$
  - $\begin{bmatrix}1\\-2\end{bmatrix}^{\mathrm{T}} \times \begin{bmatrix}1\\0\end{bmatrix} + 0 = 1$
- [0,1]
  - $\max(0, \left(\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}0\\-1\end{bmatrix}\right) = \begin{bmatrix}1\\0\end{bmatrix}$
  - $\begin{bmatrix}1\\-2\end{bmatrix}^{\mathrm{T}} \times \begin{bmatrix}1\\0\end{bmatrix} + 0 = 1$
- [1,1]
  - $\max(0, \left(\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}0\\-1\end{bmatrix}\right) = \begin{bmatrix}2\\1\end{bmatrix}$
  - $\begin{bmatrix}1\\-2\end{bmatrix}^{\mathrm{T}} \times \begin{bmatrix}2\\1\end{bmatrix} + 0 = 0$

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = \boldsymbol{w}^\top \max\{0, \boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{c}\} + b$$

$$\boldsymbol{W} = \begin{bmatrix}1 & 1\\1 & 1\end{bmatrix},$$

$$\boldsymbol{c} = \begin{bmatrix}0\\-1\end{bmatrix},$$

$$\boldsymbol{w} = \begin{bmatrix}1\\-2\end{bmatrix},$$

$$b = 0.$$

Non-linear Neurons

$$z = w \cdot x + b$$

$$y = f(z)$$

Sigmoid

Sigmoid Function

$$y = \sigma(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + \exp(-\sum_j w_j\, x_j - b)}$$

Sigmoid函数导数

$$y = \frac{1}{1+e^{-z}}$$

$$\frac{\partial y}{\partial z} = -\frac{1}{(1+e^{-z})^2}\frac{\partial e^{-z}}{\partial z} = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}}\frac{e^{-z}}{1+e^{-z}} = y(1-y)$$

$$\frac{e^{-z}}{1+e^{-z}} = \frac{1+e^{-z}-1}{1+e^{-z}} = \frac{1+e^{-z}}{1+e^{-z}} - \frac{1}{1+e^{-z}} = 1-y$$

Training

$$E = \frac{1}{2} \sum_{output} (t - y)^2$$

$$y = \frac{1}{1+e^{-z}}$$

$$z = \sum_i w_i x_i$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial w_i}$$

$$\frac{\partial E}{\partial y} = -(t - y)$$

$$\frac{\partial y}{\partial z} = y(1 - y)$$

$$\frac{\partial z}{\partial w_i} = x_i$$

$$\frac{\partial E}{\partial w_i} = -(t - y)\, y(1 - y) x_i$$

Other Activation Functions

Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)[7]		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[8]		$f(x) = \begin{cases} \alpha x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$

rectifier(x) = max(0, x)

rectifier 函数被认为有生物上的解释性。神经科学家发现神经元具有单侧抑制、宽兴奋边界、稀疏激活性等特性。采用 rectifier 函数的单元也叫作正线性单元(rectified linear unit, ReLU)

前馈神经网络

前馈神经网络 Feed Forward Neural Networks

(a)
(c)
(b)
(d)

前馈计算

- L表示神经网络的层数
- n^l表示第l层神经元的个数
- $f_l(z)$表示第l层的激活函数
- w^l表示第l层的权重
- b^l表示第l层神经元的偏置
- z^l表示第l层神经元的状态
- y^l表示第l层神经元的输出

$$\mathrm{z}^l = \mathrm{w}^l \cdot \mathrm{y}^{l-1} + \mathrm{b}^l$$

$$\mathrm{y}^l = f_l(\mathrm{z}^l)$$

$$\mathrm{z}^l = \mathrm{w}^l \cdot f_l(\mathrm{z}^{l-1}) + \mathrm{b}^l$$

前馈计算

$$\mathrm{x} \rightarrow \mathrm{z}^1 \rightarrow f_1(\mathrm{z}^1) \rightarrow \mathrm{z}^2 \rightarrow f_2(\mathrm{z}^2) \rightarrow \ldots\ldots \rightarrow \mathrm{z}^l \rightarrow f_l(\mathrm{z}^l) \rightarrow \ldots\ldots \rightarrow \mathrm{z}^L \rightarrow f_L(\mathrm{z}^L) \rightarrow y$$

训练：梯度下降法

$$E = \frac{1}{2}\sum_{output}(t_j - y_j)^2$$

$$w_{ij} \leftarrow w_{ij} - \lambda\frac{\partial E}{\partial w_{ij}}$$

$$E = \frac{1}{2}\sum_{output}(t_j - y_j)^2$$

训练：Error Back Propagating

$$y = \frac{1}{1+e^{-z}} \qquad z = \sum_i w_i y_i$$

$$E = \frac{1}{2} \sum_{output} (t_j - y_j^L)^2$$

$$\frac{\partial E}{\partial w_{ij}^l} = \frac{\partial z_j^l}{\partial w_{ij}^l} \frac{\partial E}{\partial z_j^l} = \frac{\partial E}{\partial z_j^l} y_i^{l-1}$$

$$\frac{\partial E}{\partial z_j^l} = \frac{\partial y_j^l}{\partial z_j^l} \frac{\partial E}{\partial y_j^l} = \frac{\partial E}{\partial y_j^l} y_j^l (1 - y_j^l)$$

$$\frac{\partial E}{\partial y_j^l} = \sum_k \frac{\partial z_k^{l+1}}{\partial y_j^l} \frac{\partial E}{\partial z_k^{l+1}} = \sum_k w_{jk}^{l+1} \frac{\partial E}{\partial z_k^{l+1}}$$

反向错误传播

$\frac{\partial E}{\partial z_1^4}$
$\frac{\partial E}{\partial y_1^4} = \sum_j w_{1j}^4 \frac{\partial E}{\partial z_1^5}$
$\frac{\partial E}{\partial z_1^5}$
$\frac{\partial E}{\partial y_1^5}$
$\frac{\partial E}{\partial z_2^4}$
$\frac{\partial E}{\partial y_2^4} = \sum_j w_{2j}^4 \frac{\partial E}{\partial z_2^5}$
E
$\frac{\partial E}{\partial z_3^4}$
$\frac{\partial E}{\partial y_3^4} = \sum_j w_{3j}^4 \frac{\partial E}{\partial z_3^5}$
$\frac{\partial E}{\partial z_2^5}$
$\frac{\partial E}{\partial y_2^5}$
1
2
3
4
5

反向传播

- 在第l层神经元上的误差：$\delta^l = \frac{\partial E}{\partial z^l}$
- 各层神经元上的误差
  - $\delta^L = \frac{\partial E}{\partial y^L} \odot f'(z^L)$
  - $\delta^l = \left(w^{l+1}\delta^{l+1}\right) \odot f'\left(z^l\right)$

  - $\frac{\partial E}{\partial b^l} = \delta^l$
  - $\frac{\partial E}{\partial w^l} = y^{l-1}\delta^l$

Back Propagation Algorithm

- Input x
- Feed Forward
  - For each layer $l = 2,3,\ldots,L$, compute $z^l = w^l y^{l-1}$ and $y^l = f(z^l)$
- Output Error
  - Compute $\delta^L = \nabla_y E \odot f'(z^l)$
- Back Propagating
  - For each $l = L-1, L-2, \ldots, 2$ compute $\delta^l = ((w^{l+1})\delta^{l+1}) \odot f'(z^l)$
- Output
  - The gradient of the cost function is given by $\frac{\partial E}{\partial w^l} = y^{l-1}\delta^l$

深度学习主要做什么？
调参数！
调参的秘诀在哪里？
碰运气！
调不出来怎么办？
换个模型继续调！
AI梦工厂

交叉熵损失函数

- 二次损失函数

$$E = \frac{1}{2}\sum_{output}(t_j - y_j)^2 \qquad y = \sigma(z) = \frac{1}{1+e^{-z}} \qquad z = \sum_i w_i x_i$$

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial w} = (t - y)\frac{\partial}{\partial z}\frac{1}{1 + e^{-z}}x$$

交叉熵损失函数

- 快速更新的损失函数

$$y = \sigma(z) = \frac{1}{1+e^{-z}} \qquad z = \sum_i w_i x_i$$

$$E = -\frac{1}{\mathrm{n}} \sum_x t ln(y) + (1-t)\ln(1-y)$$

$$x = 1,\ \mathrm{y} = 0$$

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial w}$$

$$-\frac{1}{\mathrm{n}} \sum_x t$$

$$= -\frac{1}{\mathrm{n}} \sum_x \left(\frac{t}{\sigma(z)} - \frac{1-t}{1-\sigma(z)}\right) \frac{\partial \sigma(z)}{\partial z} x$$

$$= -\frac{1}{\mathrm{n}} \sum_x \left(\frac{\frac{\partial \sigma(z)}{\partial z} x}{\sigma(z)\left(1-\sigma(z)\right)}\right)(t - \sigma(z))$$

$$= -\frac{1}{\mathrm{n}} \sum_x x(t - \sigma(z))$$

梯度检验

- 前向传播和后向传播计算梯度时容易出现一些小错误而不被发现，这时候就需要进行梯度检验来检验我们计算的梯度是否是我们所需要的。

$$\frac{\mathrm{dJ}(\theta)}{\mathrm{d}\theta} \approx \frac{J(\theta+\varepsilon)-J(\theta-\varepsilon)}{2\varepsilon}$$

$$\frac{\partial}{\partial\theta_1}J(\theta) \approx \frac{J(\theta_1+\varepsilon,\theta_2,\theta_3,\cdots,\theta_n)-J(\theta_1-\varepsilon,\theta_2,\theta_3,\cdots,\theta_n)}{2\varepsilon}$$

$$\frac{\partial}{\partial\theta_2}J(\theta) \approx \frac{J(\theta_1,\theta_2+\varepsilon,\theta_3,\cdots,\theta_n)-J(\theta_1,\theta_2-\varepsilon,\theta_3,\cdots,\theta_n)}{2\varepsilon}$$

$$\vdots$$

$$\frac{\partial}{\partial\theta_n}J(\theta) \approx \frac{J(\theta_1,\theta_2,\theta_3,\cdots,\theta_n+\varepsilon)-J(\theta_1,\theta_2,\theta_3,\cdots,\theta_n-\varepsilon)}{2\varepsilon}$$

SGD

- Gradient Descent
  - $\theta \leftarrow \theta + \lambda \nabla_\theta L$
- Stochastic Gradient Descent(SGD)
  - Online GD (=1), Minibatch SGD (>1)
  - 每次迭代随机选择一个样本计算梯度，并更新参数

$$w_{t+1} \leftarrow w_t - \eta \frac{\partial R(w_t, x_t, y_t)}{\partial w}$$

Mini-batch SGD

批量梯度下降和随机梯度下降的折中

每次（随机）选取m个样本进行参数的更新

Algorithm 8.1 Stochastic gradient descent (SGD) update at training iteration k

Require: Learning rate ϵ_k.

Require: Initial parameter $\boldsymbol{\theta}$

while stopping criterion not met **do**

Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.

Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$

Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \hat{\boldsymbol{g}}$

end while

梯度消失问题

- 在神经网络中误差反向传播的迭代公式为

$$\delta^l = (w^{l+1}\delta^{l+1}) \odot f'(z^l)$$

- 误差从输出层反向传播时,在每一层都要乘以该层的激
- Sigmod函数、ReLU函数、tanh函数

(a) logistic 函数的导数　(b) tanh 函数的导数

sigmod $\frac{\partial y}{\partial z} = y(1-y) \in [0,0.25]$

tanh $\frac{\partial y}{\partial z} = 1 - y^2 \in [0,1]$

Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)[7]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[8]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

Supervised Pre-training

一些超参数

- 层数
- 每层神经元的个数
- 激活函数

权重初始化

- 权重不能全部设为0
  - 每个神经元都有着相同的输出，在反向传播时具有相同的梯度
- 小随机数初始化
  - w = np.random.randn(1000)
  - 问题：一个神经网络的层中的权重值很小，那么在反向传播的时候就会计算出非常小的梯度（因为梯度与权重值是成比例的）。这就会很大程度上减小反向传播中的“梯度信号”

$$Var(z) = \sum_{i=1}^{n} Var(w_i x_i) \qquad z = \sum_{i=1}^{n} w_i x_i$$

$$Var(z) = \sum_{i=1}^{n} Var(x_i) Var(w_i), if\ E(w_i) = E(x_i) = 0$$

$$Var(z) = nVar(w)Var(x) \qquad Var(w) = 1/n$$

权重初始化：校准方差

- 解决方案：校准方差，除以1/sqrt(n)
  - `w = np.random.randn(n) / np.sqrt(n)`
  - ReLU:`w = np.random.randn(n) * np.sqrt(2/n)`

- b的初始化
  - 通常设为0
  - 有些设为如0.01的小值
  - 这样做能让所有的ReLU单元一开始就激活，这样就能保存并传播一些梯度

数据预处理

- 原始的训练数据中,每一维特征的来源以及度量单位不同,会造成这些特征值的分布范围往往差异很大。
  - 标准归一化:将每一个维特征都处理为符合标准正态分布(均值为 0,标准差为 1)
  - 缩放归一化:通过缩放将特征取值范围归一到 [0, 1] 或 [−1, 1]
  - 白化:使用 PCA (Principal Component Analysis) 去除掉各个成分之间的相关性

Early Stop

- 利用开发集
  - learn model on training data
  - valuate_score_of_current_model = evaluate on validation data
  - if valuate_score_of_current_model > best_validation_score:
    - best_validation_score = valuate_score_of_current_model
    - save current model
    - patient = 0
  - else: patient += 1
  - if patient > max_patient: – break
- 第二轮训练
  - 找到最佳训练轮数
  - 在训练集和开发集上再训练，当在开发集上达到一个较小值则停止

Bagging

- 给定训练集$D = \{x, y\}$
- 随机采样N个子集合D_i
- 在D_i上训练N个模型
- 对于训练数据分别采用每个模型得到结果
- 对于结果，利用投票（多种投票策略）输出结果

Dropout

(a) Standard Neural Net (b) After applying dropout.

Dropout vs. Bagging

- 假设输入$\{v_1, v_2\}$
  - 第一轮：Dropout{0, 1}，输入为$\{0, v_2\}$
  - 第二轮：Dropout{1, 0}，输入为$\{v_1, 0\}$

- Dropout像一种Bagging
  - 根据不同的数据子集可以训练得到一个模型
- Dropout
  - 所有模型通过一次训练得到，并且共享参数

正则化 Regularization

Which output value should
you predict for this test input?

例子

- L1正则项
  - $L = L + \alpha|w|$

- L2正则项
  - 目标函数：$L = L + \alpha\frac{1}{2}||w||^2$
  - 梯度：$\nabla L = \nabla L + \alpha\, w$
  - 权重更新：$w = w\text{-}\lambda(\nabla L + \alpha\, w) = (1 - \lambda\alpha)w - \lambda\nabla L$

数据增强

- 不过拟合的另一个途径是数据增强，即自动生成更多的训练数据
  - 增加正、负样本
  - 增加伪噪声数据

学习率

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}}$$

加速

- 自适应变长加速
  - AdaGard
  - RMSprop
  - Adam
  - AdaDelta

Algorithm 8.8 The Adadelta algorithm

Require: Decay rate ρ, constant ϵ
Require: Initial parameter $\boldsymbol{\theta}$
  Initialize accumulation variables $\boldsymbol{r} = \mathbf{0}$, $\boldsymbol{s} = \mathbf{0}$,
  while Stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$.
    Set $\boldsymbol{g} = \mathbf{0}$
    for $i = 1$ to m **do**
      Compute gradient: $\boldsymbol{g} \leftarrow \boldsymbol{g} + \nabla_{\boldsymbol{\theta}} L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    end for
    Accumulate gradient: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1 - \rho) \boldsymbol{g}^2$
    Compute update: $\Delta\boldsymbol{\theta} = -\frac{\sqrt{\boldsymbol{s}+\epsilon}}{\sqrt{\boldsymbol{r}+\epsilon}} \boldsymbol{g}$ % (operations applied element-wise)
    Accumulate update: $\boldsymbol{s} \leftarrow \rho \boldsymbol{s} + (1 - \rho) [\Delta\boldsymbol{\theta}]^2$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
  end while

步骤

- 设计网络结构
- 设定每层节点数
- 算出梯度
- 写代码
- 利用数据调参

常用的深度学习工具包

- Theano
  - Theano:蒙特利尔大学的 Python工具包，用来高效地定义、优化和执行多维数组数据对应数学表达式。Theano可以透明的使用 GPUs 和高效的符号微分。
- TensorFlow
  - TensorFlow2:谷歌开发的 Python 工具包，Linux环境，可以在任意具备 CPU 或者 GPU 的设备上运行
- Torch
  - 和Theano类似，Facebook 开发 Lua 工具包,和 Theano 类似。
  - 自动求导
  - 无缝 CPU 和 GPU 切换
- Caffe（ Convolutional Architecture for Fast Feature Embedding ）
  - 一个计算 CNN 相关算法的框架。Caffe 是用 C++ 和 Python 实现的。
  - 所要实现的网络结构可以在配置文件中指定,不需要编码。
  - 目前在 GPU 上实现 CNN 最快,在单个 Tesla K20 机器上,每天处 理 20million 图片。
  - 无缝 CPU 和 GPU 切换
  - 主要用于计算机视觉
- Keras
  - 一个极简和高度模块化的神经网络库
  - CNN和RNN
  - 无缝 CPU 和 GPU 切换
  - 支持任意链接

Taking Home Message

- 用ReLU作为激活函数
- 分类时用交叉熵作为损失函数
- SGD+mini-batch
- 数据预处理（归一化）
- 动态学习率（动态调整步长）
- 用L1或者L2正则化（跳过前几轮）
- Dropout
- 层数深（分层训练 ）
- 数据增强

谢谢