Deep Learning for Natural Language Processing (cs224d) Lecture Notes

Yihong Gu*

Contents

*gyh15@mails.tsinghua.edu.cn

1 NLP and DL: introduction

1.1 Natural Language Processing

Hierarchical Stages

首先，需要了解关于 Natural Language Processing 的几个层次：

- **Phonetic Analysis/OCR** 阶段：主要解决的是自然语言的输入端的问题，其中 **Phonetic** 为语音、**OCR** 为文本。
- **Morphological Analysis**: 主要解决词的合成的问题, 比如词的前缀和后缀的意义 (e.g. un-)。
- **Syntactic Analysis**: 主要解决句法合成的问题，研究词如何通过语法结构进行组合。
- **Semantic Interpretation**: 主要解决句子含义 (meaning) 的问题。
- **Discourse Processing**: 主要解决整篇文章的理解的问题。

Difficulty

传统的 NLP 的困难之处主要在以下两个方面，第一个方面是知识 (knowledge) 的表示，以下面句子为例

Jane hit June and then **she** [fell/ran]

显然，fell 和 ran 分别让 she 有了不同的指代 (Jane/June)，如果要让机器能够准确作出正确的指代关系的话，仅仅考虑单词的意思和语法结构是不够的，机器需要理解 hit 和 fell/ran 之间的逻辑关系，这需要进一步的先验知识。

第二个方面是模糊性 (ambiguity)，即多重含义。

1.2 Deep Learning

Representation Learning

我们考虑一个传统的 (预测类) 机器学习过程，我们把它分成两部分

$$\text{Machine Learning} = \text{Feature Engineering} + \text{Learning Algorithm}$$

在这样的过程中，我们拿到数据 (data)，先要人工设计数据的特征表示 (representation)，然后再使用特定的机器学习算法。通常而言，特征表示在整个过程的地位更高并且需要花费更多的精力。

而 Deep Learning 是什么？我们可以把它看成一种特征学习 (representation learning)，它能够从大量的数据中自动地学习特征。在具体实践中，Deep Learning 更加灵活，适用性也更强，一个适用于某个问题的 DL 模型可以直接运用于另外一个问题上。

Visualization

值得注意的是，可视化 (**visualization**) 在 DL 中非常重要，我们通常会 (使用 PCA 等方法) 把我们从 DL 中学到的 feature 投影到二维空间作可视化，我们可以从中发现一些非常有趣的性质：对于 NLP 问题来说，一个非常广泛的性质在于意思相近的单词/短语/句子在这样的欧几里得空间中也靠的更近。

DL+NLP

我们考虑 DL 在 NLP 上的应用，其核心在于把单词/短语/句子映射成 d 维空间中的向量，进而进行进一步的分析。

2 Word Vector

Word Vector 的意义在于把单词映射成 d 维空间中的向量。关于 word vector(以及之后的 NLP), 我们有一些相关的术语：

- **corpus**: 指用来作训练的文本全集。
- **token**: 文本中的某个单一的元素，可以是一个单词，也可以是一个标点，也可以是开始符 / 结束符。

传统的 NLP 中，关于单词有三种表示方法：

- one-hot: 单词的词向量 $\in \mathbb{R}^{|V|}$，其中 $|V|$ 为单词总数。
- taxonomy: 建立词与词之间的从属关系 (is-a)，比如 flower 从属于 plant(flower **is-a** plant)，相关的工作有 WordNet。
- synonym set: 建立同义词/近义词集合。

2.1 Statistical Based Method: SVD

Model

首先，我们通过文本构造 **Window based Co-occurence Matrix**，考虑构造这样一个矩阵 $X \in \mathbb{M}_{|V|\times|V|}$，其 **window size** 为 W，那么，如果单词 w_j 在单词 w_i 的大小为 W 的 window 中出现，$X_{i,j}$ 就累加 1，下面是一个例子

1. I enjoy flying.
2. I like NLP.
3. I like deep learning.

其对应的 co-occurence matrix X 为：

$$X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{array}{c} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{pmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array} \tag{1}$$

注意到这里的矩阵 X 为对称矩阵，我们对其作 SVD，令 $X = \mathcal{U}\mathrm{S}\mathcal{V}$，其中 $\mathcal{U}, \mathcal{V} \in \mathbb{M}_{|V|\times|V|}$ 并且是正定矩阵，$\mathrm{S} \in \mathbb{M}_{|V|\times|V|}$ 并且是对角矩阵，记 $\sigma_i = S_{i,i}$，取 $\mathcal{U}$ 的前 K 列 $\mathcal{U}^{[,1:K]}$ 作为最后的 K 维 word vector，这时候第 i 行的向量 $\mathrm{u}_i = \mathcal{U}_i^{[,1:K]}$ 即为单词 w_i 的 word vector。

Drawbacks and Extensions

这样的模型主要有以下几个问题：

- 由矩阵本身引起的问题：矩阵太大并且非常稀疏、加入新单词后拓展非常耗时间。
- 关于 co-occurence 的问题：需要解决部分词出现太频繁所导致的问题 (比如 the etc.)。

我们考虑这两类的问题的解决方案，首先是 co-occurence 的问题，我们可以用以下的方案来解决：

1. 令 $X^*_{i,j} = min(X_{i,j}, t)$，其中 $t \sim 100$

2. 使用 pearson correlation，同时让负值取 0 从而代替原来的简单的计数。
3. 使用 ramp window：给 window 的不同位置加权，一个比较自然的想法是距离更近的位置的权值更高。

我们无法很好的解决第一个问题，这就引入了下面的2.2的内容。

2.2 Iteration Based Methods: word2vec

我们在本节的模型中都会运用到以下记号：

- w_i 为单词集 (vocabulary) 中的第 i 个单词
- n 为单词空间的维数
- $\mathcal{V} \in \mathbb{R}^{n\times|V|}$ 为 input word matrix，其中 v_i 是 $\mathcal{V}$ 的第 i 列，表示单词 w_i 的 input vector
- $\mathcal{U} \in \mathbb{R}^{n\times|V|}$output word matrix，其中 u_i 是 $\mathcal{U}$ 的第 i 列，表示单词 $w_i$output vector
- window size 为 m。

CBOW Model

在 CBOW(Continuous Bag of Word) 模型中，我们考虑一个问题：使用周围的单词 (context) 预测中间的单词 (center word)，我们考虑使用以下的预测模型结构：

假设我们需要预测的单词是 $w^{(c)}$，其周围的单词为 $w^{(c-m)},\cdots,w^{(c-1)},w^{(c+1)},\cdots,w^{(c+m)}$，为了方便表示，我们使用这些单词的 one-hot 表示形式 $\mathrm{x}^{(c-m)},\cdots,\mathrm{x}^{(c-1)},\mathrm{x}^{(c+1)},\cdots,\mathrm{x}^{(c+m)}$，则相应的单词 $w^{(c+i)}$ 所对应的 input vector 为

$$\mathrm{v}^{(c+i)} = \mathcal{V}\mathrm{x}^{(c+i)} \tag{2}$$

我们对这 $2m$ 个向量取平均，得到一个 n 维向量 $\hat{\mathrm{v}}$

$$\hat{\mathrm{v}} = \frac{1}{2m}(\mathrm{v}^{(c-m)} + \cdots + \mathrm{v}^{(c-1)} + \mathrm{v}^{(c+1)} + \cdots + \mathrm{v}^{(c+m)}) \tag{3}$$

接下来我们得到了一个得分向量 (score vector)z，这里 $\mathrm{z} \in \mathbb{R}^{|V|}$ 其第 i 个元素表示 $\hat{\mathrm{v}}$ 和第 i 个单词的输出向量的相似程度，这里的相似程度用点乘的大小来度量)：

$$\mathrm{z} = \mathcal{U}\hat{\mathrm{v}} \tag{4}$$

最后我们把得分向量作一个 softmax 变换得到中心的单词的概率分布

$$\hat{\mathrm{y}} = \mathrm{softmax}(\mathrm{z}) \tag{5}$$

我们定义损失函数 (统计/决策论意义上的)H 为：

$$H(\mathrm{y},\hat{\mathrm{y}}) = -\sum_{y=1}^{|V|} y_i \log(\hat{y}_i) \tag{6}$$

实际上，如果让 y 为 center word 的 one-hot 向量，其中 $y_k = 1$，那么，$H(\mathrm{y},\hat{\mathrm{y}})$ 可以简化为

$$H(\mathrm{y},\hat{\mathrm{y}}) = -y_k \log(\hat{y}_k) \tag{7}$$

我们写出最后的 object function 的化简形式

$$
\begin{aligned}
J &= -\log \mathbb{P}(\mathrm{u_c}|w^{(c-m)},\cdots,w^{(c-1)},w^{(c+1)},\cdots,w^{(c+m)}) && (8)\\
&= -\log \frac{\exp(\mathrm{u}_c^T\hat{\mathrm{v}})}{\sum_{j=1}^{|V|}\exp(\mathrm{u}_j^T\hat{\mathrm{v}})} && (9)\\
&= -\mathrm{u}_c^T\hat{\mathrm{v}}+\log\sum_{j=1}^{|V|}\exp(\mathrm{u}_j^T\hat{\mathrm{v}}) && (10)
\end{aligned}
$$

其中 u_c 为 center word 的 output vector。

结合上面的表达式，我们需要对这两个 vector 有一个直观的认识，实际上，我们的优化目标是让每个 center word 周围的单词的 input vector 的平均值尽量接近其 output vector。

Skip-Gram Model

Skip-Gram Model 的方向正好和 CBOW 相反，在 Skip-Gram 中，我们给定中间的 center word，需要预测周围的 context，为了简化描述，我们直接给出概率形式的表达：式 [11] 给出了给定中间单词 w_c，单词 w_o 在其 window 中出现的概率；式 [12] 给出了损失函数的形式，这里我们使用了近一步的独立性假设 $\mathbb{P}(w_{c-m},\cdots,w_{c-1},c_{c+1},\cdots,w_{c+m}|w_c)=\prod_{j=-m,j\neq 0}^{m}\mathbb{P}(w_{c+j}|w_c)$(虽然这非常不符合实际)；我们把 [11] 代入 [12] 并且进一步化简得到式 [13]

$$
\begin{aligned}
\mathbb{P}(w_o|w_c) &= \frac{\exp(\mathrm{u}_o^T\mathrm{v}_c)}{\sum_{k=1}^{|V|}\exp(\mathrm{u}_k^T\mathrm{v}_c)} && (11)\\
J &= -\log\prod_{j=-m,j\neq 0}^{m}\mathbb{P}(w_{c+j}|w_c) && (12)\\
J &= -\sum_{j=-m,j\neq 0}^{m}\mathrm{u}_{c+j}^T\mathrm{v}_c+2m\log\sum_{k=1}^{|V|}\exp(\mathrm{u}_k^T\mathrm{v}_c) && (13)
\end{aligned}
$$

我们给出梯度的表达式：

$$
\begin{aligned}
\frac{\partial J}{\partial \mathrm{v}_c} &= -\sum_{j=-m,j\neq 0}^{m}\mathrm{u}_{c+j}^T+2m\sum_{x=1}^{|V|}\frac{\exp(\mathrm{u}_x^T\mathrm{v}_c)}{\sum_{k=1}^{|V|}\exp(\mathrm{u}_k^T\mathrm{v}_c)}\mathrm{u}_x^T && (14)\\
\frac{\partial J}{\partial \mathrm{u}_x} &= m\frac{\exp(\mathrm{u}_x^T\mathrm{v}_c)}{\sum_{k=1}^{|V|}\exp(\mathrm{u}_k^T\mathrm{v}_c)}\mathrm{v}_c^T-y_x\mathrm{v}_c^T && (15)
\end{aligned}
$$

但是，最后我们的形式仅考虑一对 $\mathbb{P}(o|c)$，最后的 object function 的结果如式 [16]，梯度的表达式如式 [17] 和 [18]

$$
\begin{aligned}
J &= -\mathrm{u}_{c+j}^T\mathrm{v}_c+\log\sum_{k=1}^{|V|}\exp(\mathrm{u}_k^T\mathrm{v}_c) && (16)\\
\frac{\partial J}{\partial \mathrm{v}_c} &= -\mathrm{u}_{c+j}^T+\sum_{x=1}^{|V|}\frac{\exp(\mathrm{u}_x^T\mathrm{v}_c)}{\sum_{k=1}^{|V|}\exp(\mathrm{u}_k^T\mathrm{v}_c)}\mathrm{u}_x^T && (17)\\
\frac{\partial J}{\partial \mathrm{u}_x} &= \frac{\exp(\mathrm{u}_x^T\mathrm{v}_c)}{\sum_{k=1}^{|V|}\exp(\mathrm{u}_k^T\mathrm{v}_c)}\mathrm{v}_c^T-y_x\mathrm{v}_c^T && (18)
\end{aligned}
$$

Negative Sampling

考虑到以上的模型在实际计算中运算量非常大，所以我们考虑另外一种近似方法，这种近似方法的阐述也分成两个部分：第一个部分是 object function，第二个部分是 gradient。

首先，我们考虑 Skip-Gram Model 的 Negative Sampling 的近似优化方法：

我们使用 (x, c) 来分别表示 $(\text{context}, \text{center})$，同时记 $D = \{(x, c)|w_x = w^{(c+j)}, -m \leq j \leq m, j \neq 0\}$，那么，我们可以得到另外一种 $\mathcal{P}(w_o|w_c)$ 的表达式 [19]

$$\mathbb{P}\big((o, c) \in D\big) = \sigma(\mathrm{u}_o^T \mathrm{v}_c) = \frac{1}{1 + \exp(-\mathrm{u}_o^T \mathrm{v}_c)} \tag{19}$$

同时使用独立性条件，我们写出 object function 的表达式 [20]，将 [19] 的结果代入，得到最终的表达式 [21]

$$J = -\log\Big\{ \prod_{(o,c)\in D} \mathbb{P}\big((o, c) \in D\big) \prod_{(x,c)} \mathbb{P}\big((w, c) \in \bar{D}\big)\Big\} \tag{20}$$

$$J = -\sum_{(o,c)\in D} \log(\sigma(\mathrm{u}_o^T \mathrm{v}_c)) - \sum_{(w,c)\in \bar{D}} \log(\sigma(-\mathrm{u}_w^T \mathrm{v}_c)) \tag{21}$$

我们把 $\bar{D}$ 称为 nagative corpus，在具体求解的时候，我们在 $\bar{D}$ 中随机抽 K 个组成 (一对的)object function [22]，其梯度为 [23]，[24] 和 [25]，其中每个单词被抽到的概率服从分布 [26]：

$$J = -\log(\sigma(\mathrm{u}_{c+j}^T \mathrm{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\bar{\mathrm{u}}_k^T \mathrm{v}_c)) \tag{22}$$

$$\frac{\partial J}{\partial \mathrm{v}_c} = -\big(1 - \sigma(\mathrm{u}_{c+j}^T \mathrm{v}_c)\big)\mathrm{u}_{c+j}^T + \sum_{k=1}^{K} \big(1 - \sigma(-\bar{\mathrm{u}}_k^T \mathrm{v}_c)\big)\bar{\mathrm{u}}_k^T \tag{23}$$

$$\frac{\partial J}{\partial \mathrm{u}_{c+j}} = -\big(1 - \sigma(\mathrm{u}_{c+j}^T \mathrm{v}_c)\big)\mathrm{v}_c^T \tag{24}$$

$$\frac{\partial J}{\partial \bar{\mathrm{u}}_k} = \big(1 - \sigma(-\bar{\mathrm{u}}_k^T \mathrm{v}_c)\big)\mathrm{v}_c^T \tag{25}$$

$$\mathbb{P}(w) \propto \big(\text{frequency of word } w\big)^{3/4} \tag{26}$$

同样，我们考虑 CBOW Model 的 Negative Sampling 的近似优化方法：

沿用之前的记号，类似的，我们可以得出 object function：

$$J = -\log(\sigma(\mathrm{u}_c^T \hat{\mathrm{v}})) - \sum_{w\neq c} \log(\sigma(-\mathrm{u}_w^T \hat{\mathrm{v}})) \tag{27}$$

同样，我们可以得出 (一对的)object function [28]，梯度 [29]，[30] 和 [31]。

$$J = -\log(\sigma(\mathrm{u}_c^T \hat{\mathrm{v}})) - \sum_{k=1}^{K} \log(\sigma(-\bar{\mathrm{u}}_k^T \hat{\mathrm{v}})) \tag{28}$$

$$\frac{\partial J}{\partial \mathrm{v}_{c+j}} = \frac{1}{2m}\Big\{ -\big(1-\sigma(\mathrm{u}_c^T \hat{\mathrm{v}})\big)\mathrm{u}_c^T + \sum_{k=1}^{K}\big(1-\sigma(-\bar{\mathrm{u}}_k^T \hat{\mathrm{v}})\big)\bar{\mathrm{u}}_k^T \Big\} \tag{29}$$

$$\frac{\partial J}{\partial \mathrm{u}_c} = -\big(1-\sigma(\mathrm{u}_c^T \hat{\mathrm{v}})\big)\hat{\mathrm{v}}^T \tag{30}$$

$$\frac{\partial J}{\partial \bar{\mathrm{u}}_k} = \big(1-\sigma(-\bar{\mathrm{u}}_k^T \hat{\mathrm{v}})\big)\hat{\mathrm{v}}^T \tag{31}$$

2.3 GloVe

我们考虑结合 Statistical Based Method 和 Iteration Based Method 的优点——GloVe, 其 object function 为：

$$J = \frac{1}{2}\sum_{i,j=1}^{|V|} f(P_{i,j})\big(\mathrm{u}_i^T \mathrm{v}_j - \log P_{i,j}\big)^2 \tag{32}$$

其中 $f(x) = \max(1, 2x)$，$\mathrm{u}_i, \mathrm{v}_j$ 为输出输入向量，$P_{i,j}$ 为单词 i, j 联合在窗口中出现的频率。

2.4 Tips

我们考虑一些实现的技巧，具体如下：

- 最后的词向量是什么？可以简单地让 word vector = input vector + output vector。
- 超参数的选择：$n \sim [25, 1000]$, $m \sim$?。
- 参数的初始化：用小的随机数初始化。

2.5 Evaluation: Intrinsic and Extrinsic

我们考虑对 word vectord 的评估，在实际的应用中，word vector 的模型是整个机器学习系统中的某一个子系统，并且这样的一个模型受到其超参数 (hyperparameter) 的影响，我们需要调整 hyperparameter 参数使得整个系统的表现尽量得好。

首先考虑 Intrinsic Evaluation，因为调整超参数后运行整个机器学习的系统会非常耗时，所以我们仅仅考虑 word vector 这样一个子系统，我们需要人为定义一个指标 P 来衡量 word vector 这个子系统的表现的好坏，同时，我们希望这样一个人为的指标 P 与整个机器系统的表现水平 P_f 正相关。

然后我们考虑 Extrinsic Evaluation，Extrinsic Evaluation 就是考虑整个系统表现水平。

Intrinsic Evaluation: Word Vector Analogies

我们考虑 word vector 在 word vector analogies 这项指标上的好坏，考虑这样一组不完整的 word analogy:

$$a : b :: c : ?$$

我们通过训练好的 word vector 求出对应 ? 的 d[33]，然后再看是否和实际情况匹配。

$$d = \arg\max_i \frac{(\mathrm{x}_b - \mathrm{x}_a + \mathrm{x}_c)^T \mathrm{x}_i}{\|\mathrm{x}_b - \mathrm{x}_a + \mathrm{x}_c\|} \tag{33}$$

这样的 word vector analogies 主要有两种，syntax 的和 semantic 的，其要求形式上就是 $\mathrm{x}_b - \mathrm{x}_a \approx \mathrm{x}_d - \mathrm{x}_c$，下面是一些例子：

- semantic: queen:king::actress:actor, Chicago:Illinois::Austin:Texas
- syntax: bad:worst::big:biggest, dancing:danced::flying:fly

在具体评测中我们考虑以下几个超参数：

- 词向量的维数 n。
- corpus 的来源和数量
- context window size m
- context 的对称性

我们评估 word vector 在 word vector analogies 上的表现 (准确率)，我们可以得出以下的一些结论：

1. 目前最好的是 GloVe，准确率最高在 75.0%(整体), 81.9%(semantics), 69.3%(syntax)。
2. 一般而言，数据集越大，效果越好；wiki 的数据相比较于报刊的数据效果更好。
3. n 的取值比较适中的时候效果最好。
4. m 越大，semantic 的效果越好，syntax 的效果先升后降。

Intrinsic Evaluation: Correlation Evaluation

另外一种评价方法是给定一对单词，让计算机和人独立评估这两个单词的相关性 $[0, 10]$，比较两者的结果。

Extension: Ambiguity*

Improving word representations via global context and multiple word prototypes (Huang et al, 2012)

主要思想为将 k-means 和 iteration methods 相结合。

Extrinsic Tasks

我们考虑一些简单的 NLP 的任务：

- named-entity recognition: 给一些专有名词分种类 (比如 John 是人，2006 是时间)。
- sentiment analysis: 判断一个单词 / 短语 / 句子的情感是积极的还是消极的。

实际上，将这些问题形式化了之后都是解决这样的分类问题：
给定数据集

$$\{x^{(i)}, y^{(i)}\}_1^N$$

其中 $x^{(i)}$ 为原始文本，$y^{(i)}$ 为给此文本做的分类标注 (假设是 one-hot vector，并且 $\in \mathbb{R}^C$)，那么我们就可以用最基本的 softmax 线性分类器来进行训练。

在已经获得了 word vector 的前提下，我们就可以用 word vector 代替原文本作为分类器的输入，在具体实现上可以把一个单词作为输入，也可以把几个单词 (一个 window) 作为输入 (window classification)。同时，我们不光可以训练分类器的参数 W，也可以再训练 word vector x，如果要进行对 word vector 的再训练的话，我们需要保证训练的数据集足够大使得其几乎覆盖 word vector 中的全部的单词。

3 Neural Networks

3.1 Max-Margin Object Function

我们考虑一个二元分类问题中一个与传统的交叉熵形式不同的 object function，max-margin object function，我们会最小化

$$J = \max(\Delta + s_2 - s_1, 0) \tag{34}$$

在这里面，我们假设对于特征空间中的每个点，我们通过神经网络 (计算图) 都能求出一个 score s，我们不妨假设我们可以做到使得第一类中 score 的最大值 s_1 小于第二类中 score 的最小值 s_2，我们希望两个类在 (通过神经网络) 投影到 $\mathbb{R}$ 上时存在一个长度为 Δ 的边界，即 $s_1 + \Delta \leq s_2$，而作为 object function 的式 [34] 就能够实现这一点。

3.2 Neuron, Terminology, Foward Propagation

我们考虑一个神经元的组成，一个神经元上 (结点标号为 i) 的 Foward Propagation 通常由以下两步组成：

$$z_i = b_i + \sum_j a_j w_{ij} \tag{35}$$

$$a_i = f(z_i) \tag{36}$$

其中 w_{ij} 为 weights，b_i 为 bias，都是需要学习的参数；$f(\cdot)$ 为 activation function。对于整个网络的 Foward Propagation，我们先对网络进行拓扑排序，然后按拓扑序对每个节点分别做 Propagation。

我们一般会使用 fully-connected 的神经网络，通常这种网络会有 layer 的概念，layer 分为 input layer x，hidden layer $\mathrm{h}^{(i)}$，output layer y，一般而言最后的输出 $y = g(\mathrm{s})$，这里的 s 称为 score，在这样分层数的神经网络中的 Forward Propagation 过程如下：

$$\mathrm{h}^{(1)} = f(\mathrm{z}^{(1)}) = f(W^{(1)}\mathrm{x} + \mathrm{b}^{(1)}) \tag{37}$$

$$\mathrm{h}^{(i)} = f(\mathrm{z}^{(i)}) = f(W^{(i)}\mathrm{h}^{(i-1)} + \mathrm{b}^{(i)}) \tag{38}$$

$$\mathrm{y} = g(\mathrm{s}) = f(W^{(n+1)}\mathrm{h}^{(n)} + \mathrm{b}^{(n+1)}) \tag{39}$$

这样的网络通常被称为 n 层的神经网络，一般而言这里的 n 层指的是 hidden layer 有 n 层，也有的地方说 $n+2$ 层 (连带输入输出层)。

上面的三个式子是向量化的表述，在程序实现的时候我们倾向于用向量式的描述实现，其中 $\mathrm{z}^{(i)}, \mathrm{h}^{(i)}, \mathrm{b}^{(i)} \in \mathbb{R}^{m_i}$，$W^{(i)} \in \mathbb{M}_{m_i \times m_{i-1}}$，其中 m_i 为第 i 层的神经元数。

3.3 Back Propagation, Computational Graph

我们首先考虑一般神经网络 (计算图) 的梯度的计算，其核心在于计算 [40]，通过链式法则我们可以求得其表达式 [41]

$$\delta_i \triangleq \frac{\partial J}{\partial z_i} \tag{40}$$

$$= f'(z_i) \sum_k \delta_k w_{ki} \tag{41}$$

那么，相关的参数的偏导也很好求了

$$\frac{\partial J}{\partial w_{ij}} = \delta_i a_j \tag{42}$$

$$\frac{\partial J}{\partial b_i} = \delta_i \tag{43}$$

继续考虑 n 层神经网络的梯度的计算：

$$\delta^{(k)} = f'(\mathrm{z}^{(k)}) \circ \left({W^{(k)}}^T \delta^{(k+1)}\right) \tag{44}$$

$$\nabla_{W^{(k)}} = \delta^{(k+1)} {\mathrm{a}^{(k)}}^T \tag{45}$$

其中 $\circ$ 为向量的 element-wise 的乘法运算。

3.4 Tips and Tricks

整个神经网络的设计调试包括以下几个方面：

1. 设计神经网络的结构，包括输入输出层，中间的连接层 (RNN, CNN, FULL)。
2. 对于全连接层，设计 activation function。
3. 调试，主要进行 gradient check。
4. 参数初始化。
5. 进行优化过程。
6. 判断模型是否足够强大会导致过拟合，如果会导致过拟合的话使用 regularization，否则重新设计模型。

我们下面依次考虑这些问题。

Non-linearities

主要有以下几种 activation function：

- $\sigma(z)$: 我们之间介绍的 sigmoid 函数。
- $\tanh(z)$: 一般而言，其效果比 sigmoid 函数好。
- ReLU: $f(z) = max(z, 0)$: 导数计算方便并且不会有 gradient vanishing 的问题。

还有一些函数比如 hard tanh, soft sign, Leaky ReLU 之类。

Gradient Check

通常 gradient check，我们使用以下式子

$$f_i'(\theta) \approx \frac{f(\theta^{(i+)}) - f(\theta^{(i-)})}{2\epsilon} \tag{46}$$

其中 $\theta^{(i+)}$ 为让 θ 的第 i 维加上 ϵ，其余维不变的向量，$\theta^{(i-)}$ 的含义类似，值得注意的是，在具体实现的时候，每次给第 i 维加上一个 ϵ 到迭代到第 $i+1$ 维的时候需要恢复第 i 维的结果。

Parameter Initialization

(Xavuer et al. 2010) 表示对于 $\tanh(z)$ 来说最好的 $W^{(i)}$ 的初始化的方法是：

$$W \sim \mathcal{U}\Big[-\sqrt{\frac{6}{m_i + m_{i-1}}}, \sqrt{\frac{6}{m_i + m_{i-1}}}\Big] \tag{47}$$

对于 sigmoid 函数来说，最好使用 $4W$。

Optimization Trick

我们一般使用 SGD 或者 Mini-batch SGD 来更新，关于 learning rate 与更新法则的相关优化有以下几种：

1. Momentum

改变更新法则变成：

$$v = \mu v - \alpha \nabla_\theta J(\theta) \tag{48}$$
$$\theta^{new} = \theta^{old} + v \tag{49}$$

一般取 $v = 0$，$\mu = 0.9$。实际上这种方法有一定的物理意义，在于让学习下降的时候不要冲得“太过”。

Adagrad

在这里面，我们对于每一个参数都设置不同的参数，定义 $g_{t,i} = \frac{\partial}{\partial \theta_i^t} J_t(\theta)$，那么，更新法则为：

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\alpha}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}}} g_{t,i}$$

Regularization

一般而言正则化的方法有以下几个：

- 在 object function 中加入正则项 (我们只惩罚 W 不惩罚 b)。
- Early-stopping
- Sparsity Contraints，让大多数情况下神经元不被激励。
- Dropout：在输入的时候让每一层随机选一半的神经元以 0 为输入，训练完成后把 weights 除以 2，这样能够有效阻止 feature co-adaptation，使得神经网络“记住“了某种模式从而丧失了推广性。

4 Recurrent Neural Networks

5 Recursive Neural Networks

6