

Deep Learning

一、Deep Neural Network

1. 对于神经网络的物理解释回顾：神经网络的每一层实际上做的事情就是一种 transformation，而 transformation 可以看作一种 encoding 也可以看作是一种特征的提取。
2. 对于浅层网络和深层网络的对比：

Shallow versus Deep Neural Networks

shallow: few (hidden) layers; deep: many layers

Shallow NNet

- more **efficient** to train (○)
- **simpler** structural decisions (○)
- theoretically **powerful enough** (○)

Deep NNet

- **challenging** to train (×)
- **sophisticated** structural decisions (×)
- **'arbitrarily' powerful** (○)
- more **'meaningful'?** (see next slide)

deep NNet (**deep learning**)
gaining attention in recent years

关于为什么 deep learning 更好用这个其实在吴恩达的 deeplearning.ai 中的解释大概如此：deep layers 中层与层之间相当于是一种排列组合的方式让我们更好的能得到结果，而只是让某一层比较胖类似于加法原则，这样不能够很好的枚举所有的情况——或者说相同情况下需要的变数是更多的。

3. 深度学习的意义在于，可以在每一层提取一些简单的特征，一层一层下去把简单的特征组合在一起从而得到了比较复杂的特征。（CS231N 的 visualization）
4. 深度学习中的挑战和关键技巧：
 - 困难的结构化设计：需要一些 domain knowledge 来设计 architecture
 - 模型复杂度高：通过大数据可以解决；正则化可以增强对于噪声的容忍程度：
 - dropout (优胜劣汰淘汰辣鸡神经元)
 - denoising (对于不太好的输入效果依然坚挺)
 - 困难的优化问题：需要小心的 initialization 反正不太好的 local minimum（一般到不了 global minimum），这一步叫做 pretraining
 - 算力需求大：提高硬件设施 GPU TPU 等

最显著的问题：

- regularization——结构化风险
- initialization——初始化

Challenges and Key Techniques for Deep Learning

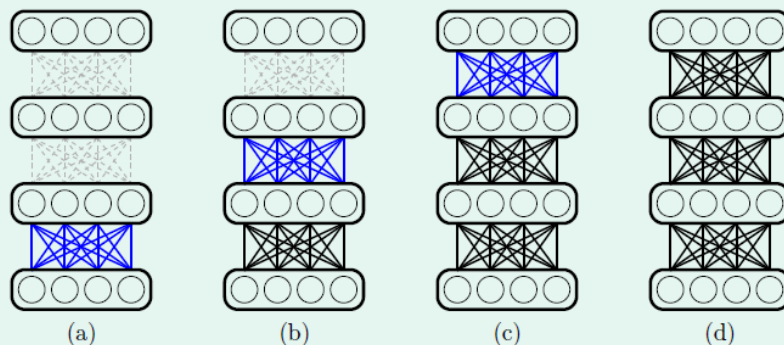
- difficult **structural decisions**:
 - subjective with **domain knowledge**: like **convolutional NNet** for images
- high **model complexity**:
 - no big worries if **big enough data**
 - **regularization** towards noise-tolerant: like
 - **dropout** (tolerant when network corrupted)
 - **denoising** (tolerant when input corrupted)
- hard **optimization problem**:
 - **careful initialization** to avoid bad local minimum: called **pre-training**
- huge **computational complexity** (worsen with **big data**):
 - novel hardware/architecture: like **mini-batch with GPU**

5. 一般DL:

A Two-Step Deep Learning Framework

Simple Deep Learning

- ① for $\ell = 1, \dots, L$, **pre-train** $\{w_{ij}^{(\ell)}\}$ assuming $w_*^{(1)}, \dots, w_*^{(\ell-1)}$ fixed



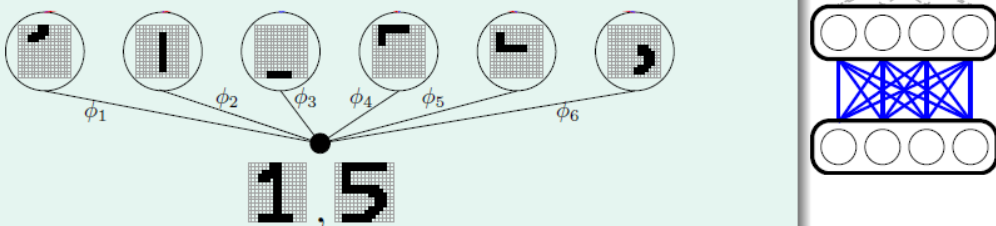
- ② **train with backprop** on **pre-trained** NNet to **fine-tune** all $\{w_{ij}^{(\ell)}\}$

二、Autoencoder

1. 权重可以看作编码：我们希望这种编码可以保留最多的信息。

Information-Preserving Encoding

- **weights**: feature transform, i.e. **encoding**
- **good weights**: information-preserving encoding
—next layer same info. with different representation
- **information-preserving**:
decode accurately after encoding



idea: **pre-train weights** towards
information-preserving encoding

2. 为了实现这个，我们可以先编码再解码：

- **autoencoder**: $d \rightarrow \tilde{d} \rightarrow d$ NNet with goal $g_i(\mathbf{x}) \approx x_i$
—learning to **approximate identity function**
- $w_{ij}^{(1)}$: encoding weights; $w_{ji}^{(2)}$: decoding weights

我们利用非线性的tanh来学习到一种identical的机制，也就是学习到相应的权重。identity function的机制作用如下

Usefulness of Approximating Identity Function

if $g(\mathbf{x}) \approx \mathbf{x}$ using some **hidden** structures on the **observed data** \mathbf{x}_n

- for supervised learning:
 - **hidden structure (essence)** of \mathbf{x} can be used as **reasonable transform** $\Phi(\mathbf{x})$
—learning '**informative**' **representation** of data
- for unsupervised learning:
 - density estimation: larger (**structure match**) when $g(\mathbf{x}) \approx \mathbf{x}$
 - outlier detection: those \mathbf{x} where $g(\mathbf{x}) \not\approx \mathbf{x}$
—learning '**typical**' **representation** of data

监督领域下我们试图发现对于输入数据的一个有效的特征转换，往往转换有着降维的效果，同时保留着代表性的数据

无监督领域下我们试图进行一些潜在结构的学习，比如密度估计或者排除一些无关的特征。

所谓自动编码器就是通过近似全等映射学习数据的表示

3. 基本的Autoencoder: $d \rightarrow \tilde{d} \rightarrow d$ NNET

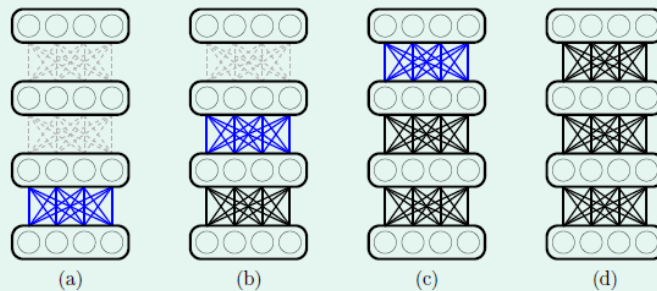
一般情况下hidden layer是降维处理的，正则化的时候我们需要让译码器和编码器的权重一样，一般这时候计算量会增加一些：

- backprop **easily** applies; **shallow** and **easy** to train
- usually $\tilde{d} < d$: **compressed** representation
- data: $\{(\mathbf{x}_1, \mathbf{y}_1 = \mathbf{x}_1), (\mathbf{x}_2, \mathbf{y}_2 = \mathbf{x}_2), \dots, (\mathbf{x}_N, \mathbf{y}_N = \mathbf{x}_N)\}$
—often categorized as **unsupervised learning technique**
- sometimes constrain $w_{ij}^{(1)} = w_{ji}^{(2)}$ as **regularization**
—more **sophisticated** in calculating gradient

总结一些用自动编码器预训练的过程：

Deep Learning with Autoencoders

- 1 for $\ell = 1, \dots, L$, **pre-train** $\{w_{ij}^{(\ell)}\}$ assuming $w_*^{(1)}, \dots, w_*^{(\ell-1)}$ fixed



by **training basic autoencoder on** $\{\mathbf{x}_n^{(\ell-1)}\}$ **with** $\tilde{d} = d^{(\ell)}$

- 2 **train with backprop** on **pre-trained** NNet to **fine-tune** all $\{w_{ij}^{(\ell)}\}$

三、Denoising Autoencoder

1. 总结一下DL中 regularization的操作：

high **model complexity**: **regularization** needed

- structural decisions/**constraints**
- weight decay or weight elimination **regularizers**
- **early stopping**

- 一些限制条件（比如加入拉格朗日因子）
- weight decay 或者 weight elimination regularizers
- early stopping

2. 介绍一种新的方法：人为加入噪声使得模型鲁棒性更加强，通过去噪得到无噪的数据：

Dealing with Noise

- direct possibility: **data cleaning/pruning, remember? :-)**
- a **wild** possibility: **adding noise** to data?

- idea: **robust** autoencoder should not only let $g(\mathbf{x}) \approx \mathbf{x}$ but also allow $g(\tilde{\mathbf{x}}) \approx \mathbf{x}$ even when $\tilde{\mathbf{x}}$ slightly different from \mathbf{x}

- **denoising** autoencoder:

run basic autoencoder with data
 $\{(\tilde{\mathbf{x}}_1, \mathbf{y}_1 = \mathbf{x}_1), (\tilde{\mathbf{x}}_2, \mathbf{y}_2 = \mathbf{x}_2), \dots, (\tilde{\mathbf{x}}_N, \mathbf{y}_N = \mathbf{x}_N)\}$,
where $\tilde{\mathbf{x}}_n = \mathbf{x}_n + \text{artificial noise}$

—often used **instead of basic autoencoder** in deep learning

- useful for data/image processing: $g(\tilde{\mathbf{x}})$ a **denoised** version of $\tilde{\mathbf{x}}$
- effect: ‘constrain/regularize’ g towards **noise-tolerant** denoising

artificial noise/hint as **regularization**!

—practically also useful for other NNet/models

我们增加噪声，把输出作为无噪的数据从而学习到一些去噪的自动编码器，实际上这个效果往往更好。噪声可以看作是一种人为的正则化操作了。

四、Principal Component Analysis

1. 考虑了非线性的去噪方法（例如tanh）我们回过头来看看线性的模型的效果：

linear hypothesis for k -th component $h_k(\mathbf{x}) = \sum_{j=0}^{\tilde{d}} w_{kj} \left(\sum_{i=1}^d w_{ij} x_i \right)$

consider three special conditions:

- **exclude** x_0 : range of i **same** as range of k
- constrain $w_{ij}^{(1)} = w_{ji}^{(2)} = w_{ij}$: **regularization**
—denote $\mathbf{W} = [w_{ij}]$ of size $d \times \tilde{d}$
- assume $\tilde{d} < d$: ensure **non-trivial** solution

linear autoencoder hypothesis:

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

向量化之后的表示十分简单（考虑到正则化的时候让译码解码的权重相等，我们就有）

$$h(x) = \mathbf{W}\mathbf{W}^T x \quad (1)$$

对应的error function我们有：

$$E_{in}(h) = E_{in}(\mathbf{W}) = \frac{1}{N} \sum_{n=1}^N \|x_n - \mathbf{W}\mathbf{W}^T x_n\|^2 \text{ with } d \times \tilde{d} \text{ matrix } \mathbf{W} \quad (2)$$

由于eigen-decompose的定理我们直到, WW^T 可以分解成 $V\Gamma V^T$, Γ 是一个对角阵, V 是一个正交阵

let's familiarize the problem with linear algebra (**be brave! :-)**)

- eigen-decompose $WW^T = V\Gamma V^T$
 - $d \times d$ matrix V **orthogonal**: $VV^T = V^TV = I_d$
 - $d \times d$ matrix Γ **diagonal** with $\leq \tilde{d}$ non-zero
- $WW^T \mathbf{x}_n = V\Gamma V^T \mathbf{x}_n$
 - $V^T(\mathbf{x}_n)$: change of **orthonormal basis** (**rotate** or reflect)
 - $\Gamma(\dots)$: set $\geq d - \tilde{d}$ components to 0, and **scale** others
 - $V(\dots)$: reconstruct by coefficients and **basis** (**back-rotate**)
- $\mathbf{x}_n = VIV^T \mathbf{x}_n$: **rotate** and **back-rotate** cancel out

我们对于 x_n 也作类似的分解。我们试图找到最优化的 Γ 和 V :

首先是 Γ : 因为试图最小化, 所以我们去掉左边的 V 最小化得到的参数是一样的, 因为 V 的效果在几何上就是一个旋转的操作

The Optimal Γ

$$\min_V \min_{\Gamma} \frac{1}{N} \sum_{n=1}^N \left\| \underbrace{VIV^T \mathbf{x}_n}_{\mathbf{x}_n} - \underbrace{V\Gamma V^T \mathbf{x}_n}_{WW^T \mathbf{x}_n} \right\|^2$$

- back-rotate** not affecting length: ✗
- $\min_{\Gamma} \sum \|(I - \Gamma)(\text{some vector})\|^2$: **want many 0** within $(I - \Gamma)$
- optimal diagonal Γ with rank $\leq \tilde{d}$:

$$\left\{ \begin{array}{l} \tilde{d} \text{ diagonal components } 1 \\ \text{other components } 0 \end{array} \right\} \Rightarrow \text{without loss of gen. } \begin{bmatrix} I_{\tilde{d}} & 0 \\ 0 & 0 \end{bmatrix}$$

$$\text{next: } \min_V \sum_{n=1}^N \left\| \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & I_{d-\tilde{d}} \end{bmatrix}}_{I - \text{optimal } \Gamma} V^T \mathbf{x}_n \right\|^2$$

我们直到 $\text{rank}(\Gamma) \leq \tilde{d}$ 所以最好的情况下我们作差的结果是 $I_{\tilde{d}}$ 在左上的形式, 其余部分都是0。这种情况下我们的差的模长是最小的。因此最好情况下的 Γ 可以表示为:

$$\Gamma = \begin{bmatrix} I_{\tilde{d}} & 0 \\ 0 & 0 \end{bmatrix} \quad (3)$$

接下来我们考虑第二步优化：

The Optimal V

$$\min_{\mathbf{V}} \sum_{n=1}^N \left\| \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_{d-\tilde{d}} \end{bmatrix} \mathbf{V}^T \mathbf{x}_n \right\|^2 \equiv \max_{\mathbf{V}} \sum_{n=1}^N \left\| \begin{bmatrix} \mathbf{I}_{\tilde{d}} & 0 \\ 0 & 0 \end{bmatrix} \mathbf{V}^T \mathbf{x}_n \right\|^2$$

- $\tilde{d} = 1$: only first row \mathbf{v}^T of \mathbf{V}^T matters
 $\max_{\mathbf{v}} \sum_{n=1}^N \mathbf{v}^T \mathbf{x}_n \mathbf{x}_n^T \mathbf{v}$ subject to $\mathbf{v}^T \mathbf{v} = 1$
 - optimal \mathbf{v} satisfies $\sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^T \mathbf{v} = \lambda \mathbf{v}$
—using Lagrange multiplier λ , remember? :-)
 - optimal \mathbf{v} : ‘topmost’ eigenvector of $\mathbf{X}^T \mathbf{X}$
- general \tilde{d} : $\{\mathbf{v}_j\}_{j=1}^{\tilde{d}}$ ‘topmost’ eigenvectors of $\mathbf{X}^T \mathbf{X}$
—optimal $\{\mathbf{w}_j\} = \{\mathbf{v}_j \text{ with } [\gamma_j = 1]\} = \text{top eigenvectors}$

linear autoencoder: projecting to orthogonal patterns \mathbf{w}_j that ‘matches’ $\{\mathbf{x}_n\}$ most

我们转换一下视野，改成最大化。考虑 $\tilde{d} = 1$ 的简单情形：

$$\max_{\mathbf{v}} \sum_{n=1}^N \mathbf{v}^T \mathbf{x}_n \mathbf{x}_n^T \mathbf{v} \text{ subject to } \mathbf{v}^T \mathbf{v} = 1 \quad (4)$$

我们利用拉格朗日因子增加相应的项并且求导一下：

$$\sum_{n=1}^N (\mathbf{x}_n \mathbf{x}_n^T \mathbf{v} - \lambda \mathbf{v}) = 0 \quad (5)$$

我们转换角度看这个式子，其实就是说明 λ 是特征值！ \mathbf{v} 是特征向量

那么我们希望这个式子最大就是希望我们 \mathbf{v} 包含着矩阵 $\mathbf{X}^T \mathbf{X}$ 中所有最大的特征值。

因此我们推广到一般的 \tilde{d} 就有了结论。

线性自动编码器就是寻找到一个最匹配 \mathbf{x}_n 的正交模式 \mathbf{w}_j 来投影

2. 我们看一下更为常用的PCA方法：

多了第一步取平均的动作，最后回传的也是中心为0的特征转换

Principal Component Analysis

Linear Autoencoder or PCA

- 1 let $\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$, and let $\mathbf{x}_n \leftarrow \mathbf{x}_n - \bar{\mathbf{x}}$
- 2 calculate \tilde{d} top eigenvectors $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{\tilde{d}}$ of $\mathbf{X}^T \mathbf{X}$
- 3 return feature transform $\Phi(\mathbf{x}) = \mathbf{W}(\mathbf{x} - \bar{\mathbf{x}})$

- linear autoencoder:
maximize $\sum (\text{maginitude after projection})^2$
- principal component analysis (PCA) from statistics:
maximize $\sum (\text{variance after projection})$
- both useful for linear dimension reduction
though PCA more popular

linear dimension reduction:
useful for data processing

它在统计上有着很好的降维的作用，常用于数据的预处理。