目录

1 课程概览

人工智能是目标，机器学习是手段（方法），深度学习是（实现）机器学习的一种技术。

1.1 机器学习（训练）三步骤

define a set of function
goodness of function
pick the best function

1.2 learning map

监督学习：数据都是正确标注的数据。包含**回归**与**分类**。

回归：输出为数值，是对连续变量的预测。

分类：输出为类别，是对离散变量的预测。

线性模型：

非线性模型：deep-learning，SVM，decision tree，K-NN...

结构学习: 分类之上，语音-》文字，机器翻译，人脸识别。（本质就是分类啦）

半监督学习：数据是同类别的有无标注的数据。

无监督学习：数据均无标注。关注“特征”而不是关注”数据”。机器阅读，创作。。。

迁移学习：数据是可不同类别的有无标注的数据。

强化学习：自行打分评价学习，常用于游戏领域。更灵活。毕竟监督学习中“老师”知识也有限。Alpha Go 是监督学习和强化学习共同作用下的成果。

2 第一章-回归

2.1 Step1:Model

模型分为线性模型和非线性模型。

线性模型：$y = b + \Sigma w_i x_i$, 权重，偏差，特征。

2.2 Step2:Goodness of Function

定义 Loss Function 损失函数 L：

$$L(f) = L(w, b) = \Sigma(\hat{y}^n - (b + \Sigma w_i x_i))^2$$

平方累加和通常用于回归学习。

最小化 Loss。通过梯度下降 Gradient Descent 来训练参数，最小化 loss。

训练过程：

（1）当前参数。

（2）计算偏导。

（3）更新参数。引入学习率。

$$w^{t+1} = w^t - \eta \frac{\theta L}{\theta w} \qquad w = w^t$$

注: 线性模型无需考虑局部最优全局最优的问题，没有局部最优。

注：loss 不会随着梯度下降一定越变越小，可能会跨越最优点。这取决与学习率 learning rate。

线性模型的偏导公式如下:

$$\frac{\theta L}{\theta w} = \sum 2(\hat{y}^n - (b + \Sigma w_i x_i^n))(-x_i^n)$$

$$\frac{\theta L}{\theta b} = \sum 2(\hat{y}^n - (b + \Sigma w_i x_i^n))(-1)$$

2.3 泛化能力 Generalization

我们真正关注的是在测试集上表现如何。

引入非线性模型，二次，三次，，，，。可以提高模型复杂度，更好地匹配训练集。但 test 不一定越来越好。-》**过拟合现象**。

欠拟合：模型没有训练好，没有很好的拟合数据，在训练和测试上都表现差。

方法：引入新特征，多项式特征等，提高模型复杂度；

减小正则化参数；

使用非线性模型，提高模型复杂度；

过拟合：在训练数据上表现好，但测试数据上表现差。

方法：获取和使用更多的数据（数据集增强）——解决过拟合的根本性方法；（非常有效，但大量数据不好获得）

特征降维: 人工选择保留特征的方法对特征进行降维；

加入正则化，控制模型复杂度；（正则化可能会增加 bias）

dropout 随机失活；（在于多个网络结构取平均）

early stop 提前结束；

BN(batch Normalization)（分批）数据归一化/标准化——统一量纲，更好训练；

正则化的概念和目的：引入权重到损失函数中，在最小化 loss 的同时尽量减小权重的值。更小的权重意味着曲线更平滑，抗噪声能力强（使得高次项对曲线形状的影响尽可能小）。可以有效解决过拟合问题。正则化项控制权重的更新，控制模型复杂度。不需要对 bias 进行正则化，因为 bias 只影响曲线上下平移，不影响曲折，不会过拟合。

过拟合本质原因：为了更好地拟合训练数据，引入了过多高次项。曲线过于曲折，往往导致过拟合，对 test 的误差更大。需要正则化控制曲线的蜿蜒程度，一致过拟合现象。（小的权重意味着更平滑）

总而言之，为了避免欠拟合和过拟合。综合的方式是引入适当高次项，不过同时在 loss 中引入正则化控制。

2.4 误差来源-variance 方差和 bias 偏差

variance：方差是指训练的结果与训练结果平均值之间的方差。方差度量了同样大小的训练集的变动所导致的学习性能的变化，即刻画了数据扰动所造成的影响。方差越大，说明数据分布越分散。

bias：偏差是指训练结果平均值与真实值之间的偏差。偏差度量了模型的期望预测与真实结果的偏离程度，即刻画了学习算法本身的拟合能力。偏差则表现为在特定分布上的适应能力，偏差越大越偏离真实值。

模型越复杂，偏差越小，方差越大；模型越简单，方差越小，偏差越大。

方差大模型复杂易导致过拟合；偏差大模型简单易导致欠拟合。两种误差之间需要 trade-off。

2.5 交叉验证（也可以减轻过拟合）

（1）train set 分为 train 和 validation。训练 train。

（2）找在 validation 中表现最好的模型（loss 最小）。

（3）不变参数，在整个 train set 上重新训练，即为最终模型。

3 第二章-梯度下降

梯度下降方向为 loss 图像等高线法线方向。

3.1 动态设置学习率

学习率过大会振荡/起飞，过小太慢。——》引出每次迭代都调整学习率。一般而言开始离终点远学习率较大，后面离终点近学习率应该调小。

甚至不同参数设置不同学习率。

3.1.1 1/t decay

$$\eta^t = \frac{\eta}{\sqrt{t+1}}$$

3.1.2 Adagrad

使用过往梯度的均方根。

$$\frac{\eta^t}{\sigma^t} = \frac{\eta}{\sqrt{\Sigma(g^t)^2}} \qquad t \in [0,t]$$

$$w^{t+1} = w^t - \frac{\eta^t}{\sigma^t} g^t$$

问题来了，按理说 grad 大 move 大，这里引入均方根显然不能保证 grad 最大 move 大。事实上，确实不是当前 grad 大移动就一定大的。

Adagrad 表示 How Surperise it is(体现反差)，示例如下，Adagrad 会强调当前的反差，如果梯度突然变大或变小，通过 Adagrad 会表现出它的反差（比如，对于突然变大的梯度，我们给予更多的关注，步长也就越大；对于突然变小的梯度，步长越小)。

另一点，大 grad 不一定大 move。因为考虑两个参数时就不满足了。eg：两个开口不同程度的二次函数，一个导数大但里中点近，移动小。因此 adagrad 中 g^t 是一阶导，$\sqrt{\Sigma(g^t)^2}$ 是二阶导估计值。因此 adagrad 提供了跨参数有效的梯度下降学习率更新。

3.2 随机梯度下降 Stochastic

普通梯度下降每次迭代都要累计所有的梯度/loss, 训练较慢。随机梯度下降为了更快，不需要计算所有训练样本，随机挑一个训练样本的 loss 值作为整个迭代的 loss。虽不精确但快，而且结果其实不差多少。

3.3 特征归一化/标准化 Feature Scaling

统一量纲，将不同特征分布缩放到同一范围内。这样 loss 更接近与一个正圆，好训练。

操作方法：对每一维特征计算均值和标准差。这样每一维特征的均值为 0，标准差为 1。

$$x_i^r = \frac{x_i^r - \mu_i}{\sigma_i}$$

3.4 梯度下降的数学解释

发现附近以 θ 半径圈的最小值，移动到那里。理论是泰勒级数的近似（忽略高次项)。移动方向与梯度方向（导数）相反时内积最小。

使用前提：

（1）loss 无限可微。

（2）附近圈小，可以忽略高次项。意味着学习率不宜过大。

4 第三章-分类

可以用回归解决分类吗？比如有四类，0-0.25 为第一类，0.25-0.5 为第二类，下面依次。

不可以，这样暗含着第一类与第二类相比于第一类和第三类更近。problematic。**不可根据回归计算的数值直接分类，分类间隐含关联性**。

分类问题的 loss 可以就是统计模型预测和真实不一致的个数吗?

不可以，这样的 loss 不可微分，不能使用梯度下降方法求解模型。

4.1 生成模型法 Generative model

根据贝叶斯公式可根据先验概率求后验概率

$$P(C1|x) = \frac{P(C1)P(x|C1)}{P(C1)P(x|C1) + P(C2)P(x|C2)}$$

根据假设分布（一般为高斯分布）求 $P(x|C1)$。要求出两类别的均值 μ 和协方差矩阵 Σ。

均值计算公式 $\mu = \frac{1}{N}\sum x^n$

协方差矩阵计算公式 $Cov(x) = \frac{1}{N}\sum(x^n - \overline{x})(x^n - \overline{x})^T$

然而当两个协方差矩阵都考虑时，模型太复杂，参数太多，往往效果不好。-》减小参数提高准确率-》两个协方差矩阵综合为一个。-》**简化参数，两类共享参数训练**。-》**共享协方差后出来的分布结果分布线是线性的 linear**。

$$\Sigma = \frac{N1}{N}\Sigma^1 + \frac{N2}{N}\Sigma^2$$

生成模型是通过假设分布的均值标准差最大化生成模型可能性的。最大化的公式如上，因此，生成模型倒是没什么训练过程，根据公式算。=》**关注假设分布的** N, μ, Σ。

4.2 逻辑回归法 Logistic regression-判别模型

当共享协方差均值时，并使用另一个假设分布（朴素贝叶斯）。可以得到如下公式：(Sigmoid)

$$P(C1|x) = \sigma(wx + b)$$

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

我们可以尝试直接训练参数（w,b）提取特征计算**误差最小化**的参数，而不是假设分布获得**分布可能性最大化**的参数。

a set of model

$$a = \sigma(z)$$

$$z = \sum w_i x_i + b$$

假设 $\hat{y}^n$:1 for class 1,0 for class 2。

goodness of a function[logistic regression+Cross entropy]

$$L(w, b) = f_{w,b}(x^1) f_{w,b}(x^2)(1 - f_{w,b}(x^3))...$$

这里 x3 为第二类，loss 表示为第一类所有可能概率乘积，如果分类都准确百分百的话，loss 应该为 1。也就是说这里求最大化 loss 的参数取值。因为概率 $f_{w,b} \in [0, 1]$，所以 ln 它为负值。所以可以通过最小化 $-lnL(w, b)$ 的方式-》转到梯度下降方法。

引入**交叉熵**计算 loss。有

$$H(p, q) = -\sum p(x) ln(q(x))$$

假设 $\hat{y}^n$:1 for class 1,0 for class 2。(伯努利交叉熵)

$$-lnL(w, b) = -\sum(\hat{y}^n ln f_{w,b}(x^n) + (1 - \hat{y}^n) ln(1 - f_{w,b}(x^n)))$$

更新参数： Sigmoid 的导数为 $y(1 - y)$, 这里 y 是 $f_{w,b}(x^n)$

整个导的过程为 $(lny)' --> \frac{1}{y}(y)' --> (1 - y)(z)' --> (1 - y)(wx + b)'$

$$w^{t+1} = w^t - \eta \sum(-(\hat{y}^n - f_{w,b}(x^n))x_i^n)$$
$$b^{t+1} = b^t - \eta \sum(-(\hat{y}^n - f_{w,b}(x^n)))$$

这么看来和回归时的梯度下降平方误差 loss 的更新差个"2" 而已。但是若这里用上平方误差 loss 以及参数更新则不然。

为什么回归时的平方误差（square error）不能用？如果有 [**logistic regression+square error**]

$$L(w,b) = \frac{1}{2}\sum(f_{w,b}(x^n) - \hat{y}^n)^2$$

有

$$\frac{\vartheta L}{\vartheta w_i} = (f_{w,b}(x) - \hat{y})f_{w,b}(x)(1 - f_{w,b}(x))x_i$$

那么当 $f_{w,b}(x)$ 为 1 还是为 0 时都是梯度为 0，显然不能够训练。（无法区分两个类）

从 loss 图像上面也可以看出，交叉熵更容易下降，而平方误差技术是个平面。

4.2.1 逻辑回归和线性回归对比

（1）模型引入一个 Sigmoid。

（2）使用交叉熵。找 $-lnL(f)$ 最小化时的模型参数。

（3）更新参数的式子一样。（就是差个常数系数，这个无所谓）。

4.3 Discriminative 判别模型 v.s.Genetative 生成模型

前者直接训练使误差最小化参数提取特征，使用梯度下降训练参数。后者则是假设分布并计算分布可能性最大时的分布参数，再根据朴素贝叶斯计算分类概率。

生成模型有一个分布假设，判别模型无假设。所以没有判别模型表现好一般。

但是生成模型假设分布能使用更少的数据训练并抗噪音更好一点。（因为已经假设了分布）

4.4 多分类

多分类一样利使用逻辑回归计算每种分类的可能性概率，并使得所有概率之和为 1（缩放）。用一个 softmax 选择最大的作为结果。

hardmax 最大的特点就是只选出其中一个最大的值，即非黑即白。Softmax 的含义就在于不再唯一的确定某一个最大值，而是为每个输出分类的结果都赋予一个概率值，表示属于每个类别的可能性。（这一点和半监督学习中 self-learning 中的 soft label 和 hard label 相似。）

这里交叉熵直接利用原始公式 $H(p,q) = -\sum p(x)ln(q(x))$ 计算来评估 loss。交叉熵越小越好。

4.5 逻辑回归的局限性

（我们是从共享协方差矩阵引出的逻辑回归）逻辑回归的分类线只能是个线性的线条。也就是-》**线性不可分**：图像表示所有样本中如果找不到一条线分类，那么不能使用逻辑回归。

不过这一点可以通过转化样本参数使它们有一条线性分界线来解决。

即**特征转化 Feature Transformation+ 分类 Classfication**!

由此我们就引出了深度学习啦！隐藏层就是层层的特征转换，最后输出层就是分类！因此深度学习可以处理复杂的非线性模型。

5 第四章-深度学习

Neural Network->goodness of function->pick the best function

一个网络架构就对应之前步骤中的一个 function。

神经元之间的不同连接构成了不同的网络结构。

网络参数：所有神经元的所有的权重和偏差。(层数，神经元数属于训练前指定的结构参数（超参数)，而不是训练出来的参数，不过，实践中也有人训练最佳层数。)

超参数：人为设定的调优参数，比如层数、神经元数，不是训练得到的。

结构：**input layer+hidden layers+output layer**

只有大于等于 1 个隐藏层的模型才叫深度学习。隐藏层用于特征转换提取（本质特征工程，特征提取器)，输出层则最后作为一个多分类器。深度意思就是很多隐藏层，一般随深度增加，特征提取更“明显清晰”，loss 不断变小。

深度学习神经网络可以并行矩阵计算，适用于 GPU。

神经网络通过神经元的设计，可以降维（分类)。

5.1 Fully Connect Feedforward Network 全连接反馈网络

每一个上层神经元输出进入下层所有神经元，每一个下层神经元接收所有上层神经元。

5.2 如何计算 loss

交叉熵

$$l(y, \hat{y}) = -\sum \hat{y}_i ln y_i$$

$$Totalloss = \sum l^n$$

找到使 loss 最小的网络参数。如何找：梯度下降-》那么多层，梯度怎么下降：反向传播。

将其视为树结构，对于计算复杂度来说，正向传播和反向传播都是 O(n)，n 是节点总个数。

但是内存复杂度来说，反向传播因为需要保留正向传播时所有的中间结果，所以需要 O(n)，这也是神经网络特别耗 GPU 资源（爆显存）的祸源。

正向传播内存复杂度为 O(1), 但每计算一个变量的梯度都要扫一遍。

反向传播从根节点向下扫，可以保证每个节点只扫一次（在计算一个变量梯度时不用管同层的其他变量)；正向传播从叶子节点向上扫，会导致上层节点可能会计算多次。

5.3 为什么要 deep，而不是一层平铺神经元

普遍定理：**浅层网络可以完成任何功能，但是层多的结果更有效**。注意：要比较浅层和深层网络的性能，需要保持参数数目一致，但注意，参数一致可不等于神经元数目一致。

同时，deep 可以模块化。

举个例子，直接分类长发男女为四类（长发男，长发女，短发男，短发女）不 deep 直接分类的话，效果不好，因为显然长发男训练样本较少，这个分类器是 weak 的，结果不会很好。然而如果使用 deep 的话，可以通过隐藏层进行特征提取，比如先提取男女，长短发，再最后分类。这样每个分类器都有足够的训练样本。

更引申地说，deep 神经网络由于本身的复杂性，可以相较于其他的使用较少的数据训练出较好的结果。这是由于**隐藏层的提取特征**的本质。-》**can be trained by little data**。对着 deep 加深，特征提取地越来越详细。

5.4 Backpropagation 反向传播 BP

相较于正向传播计算更有效率。

5.4.1 链式法则

求导的链式法则。

$$x = g(s) \qquad y = h(s) \qquad z = k(x, y)$$

$$\frac{dz}{ds} = \frac{\vartheta z}{\vartheta x}\frac{dx}{ds} + \frac{\vartheta z}{\vartheta y}\frac{dy}{ds}$$

5.5 最后一层输出层的梯度传播

$$L(\theta) = \sum l^n(\theta) - - - - > \frac{\vartheta L(\theta)}{\vartheta w} = \sum \frac{\vartheta l^n(\theta)}{\vartheta w}$$

$$z = \sum w_i x_i + b$$

$$\frac{\vartheta l}{\vartheta w} = \frac{\vartheta l}{\vartheta z}\frac{\vartheta z}{\vartheta w}$$

正向传播就是对每个参数计算 $\frac{\vartheta z}{\vartheta w}$; 显然就是输入的 x_i

反向传播就是对每个激活函数计算 $\frac{\vartheta l}{\vartheta z}$;

$$a = \sigma(z)$$

$$\frac{\vartheta l}{\vartheta z} = \frac{\vartheta l}{\vartheta a}\frac{\vartheta a}{\vartheta z}$$

其中 $\frac{\vartheta a}{\vartheta z}$ 就是激活函数的导数 $\sigma'(z)$。

根据链式法则，注意 l 始终是输出层最后的交叉熵。

$$\frac{\vartheta l}{\vartheta a} = \frac{\vartheta z'}{\vartheta a}\frac{\vartheta l}{\vartheta z'} + \frac{\vartheta z''}{\vartheta a}\frac{\vartheta l}{\vartheta z''}$$

其中 z', z'' 是后面层激活函数前的值。

总结，**BP 对每个激活函数计算的**，中间的激活函数

$$\frac{\vartheta l}{\vartheta z} = \sigma'(z)(w_3\frac{\vartheta l}{\vartheta z'} + w_4\frac{\vartheta l}{\vartheta z''})$$

最先的输出层计算，前面为交叉熵导，后面为激活函数导。

$$\frac{\vartheta l}{\vartheta z'} = \frac{\vartheta l}{\vartheta y_1}\frac{\vartheta y_1}{\vartheta z'}$$

$$\frac{\vartheta l}{\vartheta z''} = \frac{\vartheta l}{\vartheta y_1}\frac{\vartheta y_1}{\vartheta z''}$$

最后，回到初始公式，**计算梯度为**：前面一项是反向传播计算保存的，后面一项是正向得到的。

$$\frac{\vartheta l}{\vartheta w} = \frac{\vartheta l}{\vartheta z}\frac{\vartheta z}{\vartheta w}$$

6 第五章-卷积神经网络 CNN

CNN 主要是用来特征提取，不止可以处理图像，语音，文本都可以特征提取。可以用来减少参数量，使用小参数与小特征相关联。

6.1 图像特性

（1）有些特征图案比整个图案小得多。因此可以使用小参数与小特征相关联。-》xx 特征探测器

（2）相同特征图案可能出现在不同位置。-》同个探测器（参数相同）探测整个图像。

（3）对图像降低像素不会改变物体本身是个什么东西。-》子采样，减少参数量

在全连接神经网络前通过重复的卷积层（（1）（2））和池化层（（3））来进行特征提取，减少参数量。最后平铺参数，全连接神经网络训练。另外：卷积和池化之间可以添加激活函数，通常为 ReLU。

6.2 卷积层

多种卷积核（可以不同大小）作为探测器进行特征提取。规定卷积核移动步长。一次结果为卷积核矩阵内对应位置的乘积之和。多个卷积核生成特征图 Feature Map。

每个卷积核都是一个 channel，通道。**卷积 v.s. 全连接**

本质上卷积也是一个神经元连接方式。不过卷积**连接的参数更少**。eg：卷积 6*6 每 3*3 得到 4*4。就是 16 个 neuron，每个 neuron 有 9 个参数。而全连接的神经元为 36（可能小于这个，自己规定的），每个 neuron 参数为 36。即卷积进行了神经网络的**稀疏操作**。同时卷积中每个神经元共享这 9 个参数。即**权值共享**。这两种特性都**显著地降低了参数量**。

6.3 池化层

规定池化池大小，在每个池化池中选择最大地跳出来，其他的直接扔。进行**子采样**。(类似降低像素) 不影响它是什么。

卷积和池化最终生成了一个新的小的图片。通道数等于累计卷积核数。

6.4 Flatten-全连接

平铺 Feature Map 地每一个，进入全连接神经网络训练。

6.5 CNN 学了什么

卷积和池化提取特征构成特征图。平铺每个特征进入全连接训练。神经网络中的每个神经元都对应着一个特征图像。然而维度过高，人类无法直接通过神经元特征图像理解机器学习地学到地东西。另一方面来说，这一特点也意味着深度神经网络很容易收到噪音愚弄。

另外 CNN 也可以进行 xx 合成。因为 CNN 的本质是提取特征。可以从一个里面提取 content，另一个提取 style。这一个功能和 AE 的特征接纠缠应用类似。

playing Go 也可以用 CNN，围棋游戏，提取特征（由于图像特性 (1)，(2))，比如马上要围了。但显然，池化不能用。

7 第六章-神经网络训练技巧

整个训练流程为：define a set of function-》goodness of function -》pick the best function

注意有时候 test 不好，不一定就是过拟合。要首先考虑 train set 是否训练充分了。

7.1 训练不好

7.1.1 调整激活函数-Relu，Maxout

梯度消失和梯度爆炸：这是针对前面几层的梯度。随着链式法则反向传播梯度。往往由于激活函数导致梯度指数减小/增大的现象。Sigmoid 导数最大为 0.25，通常导致梯度消失。

前面的梯度消失了，会导致学习的很慢，参数很难引起波动，甚至不学习。因此后面层参数学的都收敛了，前面还几乎是随机的。

激活函数的作用为了拟合非线性函数，加入非线性激活函数，能训练更复杂的模型。

线性修正单元 ReLU

一个分段函数，小于 0 取 0，大于 0 取 x (正比例)。优点：计算快；符合真实生物学；可解决梯度消失。

取 0 时相当于这个神经元没了。使神经网络动态地变瘦（每一个不同 x 不一样)。剩下的都是相当于线性激活，不会造成前面层小梯度。

ReLU 的变体

Leaky ReLU：小于 0 的部分改为 a=0.01z。允许小于 0 学，尽管学得慢。

Parametric ReLU：小于 0 的部分改为 a=pz。其中 p 也可以通过梯度下降学出来合适的。

Maxout

ReLU 是 Maxout 的特殊情况。Maxout 是一个可学习的激活函数。Maxout 的原理是同层神经元几个一组，取其中最大的 z 作为 a。也是分段函数，非线性。其中分段函数的形式，完全由同组内的 w，b 决定。($z = \Sigma wx + b$)。因此是可学习的激活函数。

也是动态的神经网络，提高复杂性和更 thinner。不同的实例对应不同的 thin 的分段线性网络。

7.1.2 调整学习率-adagrad，RMSProp，Momentum，Adam

adagrad

$$w^{t+1} = w^t - \frac{\eta}{\sqrt{\Sigma(g^i)^2}} g^t \qquad i \in [1, t]$$

adagrad 本质是使用一阶导估计二阶导。

RMSProp 自适应学习率方法

loss 图可能会很复杂。

$$W^{t+1} = w^t - \frac{\eta}{\sigma^t} g^t$$

$$\sigma^t = \sqrt{\alpha(\sigma^{t-1})^2 + (1-\alpha)(g^t)^2}$$

其实相较于 adagrad 的本质就是调节当前梯度的比重，这个值可以自适应学出来。

Momentum 动量

存在一些梯度为 0 或约为 0 的临界点。几乎平的位置；鞍点 (=0)；局部最优点 (=0)。怎么脱离出去?

移动方向为目前最后的运动方向减去梯度。因此，即使梯度为 0，也能照着之前的方向继续走。为什么是减，因为之前梯度方向是运动的反方向。类似物理的惯性。移动不知考虑梯度，还考虑上一步的运动。

$$v^1 = \lambda v^0 - \eta grad$$

$$\theta^1 = \theta^0 + v^1$$

因此 momentum 方法可跨越临界点（不保证一定）和加速收敛。

Adam

综合使用了 RMSProp 自适应学习率和 Momentum 动量方法。

7.2 训练好但 test 不好

7.2.1 提前结束

（训练模型时）不要迭代太多次，提前结束。迭代太多次模型太贴合 train，太陡峭，过拟合。

7.2.2 正则化-L1,L2

在 loss 中引入权重，在最小化 loss 的同时，使一组权重还接近于 0。默认权重小的函数较平缓。

L2

$$L'(\theta) = L(\theta) + \lambda\frac{1}{2}|\theta|_2$$
$$|\theta|_2 = (w^1)^2 + ...$$

求导

$$\frac{\vartheta L'}{\vartheta w} = \frac{\vartheta L}{\vartheta w} + \lambda w$$

更新的函数化简一下可以看到 L2 正则化使用乘法进行**权重衰减**。

$$w^{t+1} = (1 - \eta\lambda)w^t - \eta\frac{\vartheta L}{\vartheta w}$$

L1

$$L'(\theta) = L(\theta) + \lambda\frac{1}{2}|\theta|_1$$
$$|\theta|_1 = |w^1| + ...$$

求导 (上面是绝对值,sgn(x) 表示返回 x 的符号)

$$\frac{\vartheta L'}{\vartheta w} = \frac{\vartheta L}{\vartheta w} + \lambda sgn(w)$$

更新的函数化简一下可以看到 L2 正则化使用加法进行**权重衰减**。(绝对值整体是让 w 往 0 靠，可小于大于 0)

$$w^{t+1} = w^t - \eta\frac{\vartheta L}{\vartheta w} - \eta\lambda sgn(w^t)$$

7.2.3 Dropout 随机失活

训练时随机使%p 的概率使神经元失活。(即不训练) 每次每个 example 失活的都不同，最后是整体都训练了。网络不断再变同时更瘦。(网络瘦的时候起码计算量少一点。。。。) For each mini-batch,we resample the dropout neurons.

测试时没有 dropout。同时为了保持结果一致。将每个神经元的参数乘以%p。

这种方法的好处是，训练时参与的神经元少，每个神经元干的任务更重，训练更好更彻底?

同时这种方式相当于**在训练时训练多种不同结构的网络, 取平均**。

7.3 临界点梯度问题

鞍点：特征值有大有小。$v^THv > 0$ < 0 都有。

局部最大: 所有特征值都是负的 $L(\theta) < L(\theta')$H 是负定矩阵。$v^THv < 0$。

局部最小：所有特征值都是正的 $L(\theta) > L(\theta')$ H 是正定矩阵，$v^THv > 0$。

计算方式：选定点 θ' 表示 loss。计算一阶和二阶导。得出关于 w 的 “协方差矩阵” (?) H。计算 v^THv。v 是距离选定点的距离向量。

7.4 分批处理 batch

一轮迭代可将样本分为多个 batch，每轮迭代后对 batch 进行重新洗牌。

有无 batch 都需要过完整个迭代计算 loss。只是有 batch 时在一次迭代中每次 batch 完毕后都更新一下参数。

batch 多更新频率快，快速收敛，还能洗牌。每次更新冷却时间短（间隔时间），但易受噪声影响。显然更多更小的 batch 需要花费同等数据量更多时间计算。更新快但一轮迭代满。

一般 batch 多点好。测试表现好。大 batch size 可能会导致优化失败。

batch 少时可能会卡住（就一条线）。

batch 少一次迭代 epoch 快；batch 多的优化能力和泛化能力（test 上）更强。自己权衡。

8 第七章-循环神经网络 RNN

之前学到的神经网络 DNN,CNN 都是独立看待每一个 x。一个图像什么什么的都是单独分类。二对于语言文本处理的任务，若是一个字一个字/一个词一个词处理，都需要上下文联系，比如“去北京”和“离开北京”。因此我们的神经网络需要让上一个 x 对下一个进行影响。即引入**记忆**。

8.1 word 怎么变成输入的向量格式

（1）词典。apple 对应 [1,0,,,0]。其他依次类推。缺点是向量长度 n 太大，为一个词典大小。

（2）引入 other，限制向量长度。

（3）word hashing。比如 26*26*26 的字母表。aaa，aab，，，，zzz。apple 可表示为“app”，“ppl”，“ple”。转为数字编码。

8.2 提取文本关键词归于哪个槽（slot）

本质也是分类问题。目的地和始发地的区分则是我们所说的记忆。**神经网络需要记忆**。记忆可视为另一种输入。

普通线性激活函数 DNN 网络，可以每层引入内存单元。存储上一层的中间结果（记忆）。

因此对于这种网络而言，改变文本单词顺序，会改变输出结果。

注意不同的 x 依次训练使用的是同一套网络。

通过再每一层神经网络上加上记忆单元即可实现深度神经网络。

（1）Elman Network：将平行层的中间结果作为同层记忆给下一个。记忆中间结果。

（2）Jordan Network：将上一个的输出结果作为记忆给下一个。记忆最后结果。

双向 RNN，x_2 的训练 needx_1 和 x_3 的记忆。“上下文”。

8.3 LSTM 长短时记忆网络

有四个输入，一个输出。四个输入种有一个为 data，另外三个为控制程度信号。类似闸门，通过开放程度控制。

四个模块：输入门（data,control1), 记忆单元，遗忘门（control2），输出门（control3）。

控制信号的激活函数通常为 Sigmoid，可压缩到 01 之间。类似闸门。

经过记忆单元后 $c' = f(z_1)g(data) + f(z_2)c$，分别是输入门，遗忘门控制信号。输出为 $a = f(z_3)h(c)$，为输出门控制信号。

例子可有 [x1,x2,x3,1] 四个输入 x（参数），不同输入采取不同的权重。因此 LSTM 实现了**参数加倍**的功能。直接替换原始神经元为 LSTM。

CNN 的卷积即是稀疏操作，权值共享，以此减少了参数量，便于训练。但 RNN 不行，RNN 文字熵较大。因此成倍增加参数量，才能较好训练。

RNN 的基本思想是通过**时间的反向传播（BPTT)**。

8.4 梯度消失和梯度爆炸

梯度消失：loss 图像平稳项太多，前面的层几乎不学习。

梯度爆炸：loss 出现悬崖，来回震荡，学习不稳定。

本质是神经网络反向传播 (BP) 计算梯度时的激活函数的导数与 1 的关系。如果比 1 小，交叉熵（梯度）传播过程传到前面的层越来越小指数型梯度减小几乎没了。如果比 1 大，传到前面层的梯度越来越大指数型梯度增大。

具体到 RNN 为什么会出现梯度爆炸/消失。假设最普通的线性激活函数。一层一层传下去 w。if w=1.01，1000 次后就变成了 20000；if w=0.99，1000 次后就变成了 0。（连乘）

梯度爆炸可以用 clipping 解决，即限制梯度的最大下降限度。

LSTM 可解决梯度消失，因为其中复杂的门结构，更新时会记住前面几次的残留记忆。前面的影响不会消失，除非遗忘门关闭。因此，如果遗忘门打开，就无梯度消失问题。

8.5 RNN 的其他应用

可用于输入输出不等长的情况，以及输入不定长的序列。

“many to one” 提取文本意思对电影评分 “1，2，3....”。

“many to many” 输出比输入短，可用来语音转文字。语音可能有拉长音空音的现象，可提取文本。通过 CTC 解决叠词（拉长音）问题，在可能的空白处插入”null” 进行各种对齐匹配。

9 第八章-半监督学习

半监督学习：存在同类但无 label 的数据（注意，都是同类的）。

通常无标签数据比有标签多。因为收集的困难性，人工打标签贵。训练需要假设。

Transductive learning：无标记数据就是 test data。

Inductive learning：无标记数据不是 test data。

半监督学习目前做的所有假设/训练方法都是为了给未 label 数据分类。

9.1 半监督和监督下的生成模型

监督：求 $P(C_1), P(C_2), \mu_1, \mu_2, \Sigma for P(X|C_1), P(X|C_2)$。假定高斯分布，所有数据都能一下估计出。

半监督：首先使用 label 数据进行估计得出高斯分布假设下的相关参数。之后使用未 label 的数据对模型重新估计，更新参数模型。

（1）**初始化**参数 $P(C_1), P(C_2), \mu_1, \mu_2, \Sigma$。可以采用 label 数据，也可以随机。

（2）根据模型**计算**未 label 的 $P(C_1|X^u)$。

（3）根据以下公式**更新**参数。即 $\Sigma P(C_1|X^u)$ 约等于未 label 的 C_1 数目。

$$P(C_1) = \frac{N1 + \Sigma P(C_1|X^u)}{N}$$

$$\mu_1 = \frac{1}{N1}\Sigma x^r + \frac{1}{\Sigma P(C_1|X^u)}\Sigma P(C_1|X^u)x^u$$

（4）循环执行 2，3 步，直至参数收敛。

9.2 低密度分离假设

self-training：

（1）根据 label 数据训练模型，什么方法都可以。除了回归。

（2）根据模型对未 label 的数据计算伪标签（pseudo-lable）。

（3）选取一部分伪标签数据加入到标签数据。重复 1，2，3 直至模型收敛。这部分可以自己定怎么选，也可以自己定怎么加权重。

为什么回归没用，因为回归预测出来的数据重新用来做训练，不会影响模型的参数。（模型的参数本来就是这个）

self-training 其实是对未 label 数据指定一个分类，生成模型则是计算未 label 数据对于每个分类的概率。

hard label：计算伪标签时选取概率最大的记作“1”。

soft label：计算伪标签时直接带着概率。xx 为 0.7，xx 为 0.3。（类似生成模型）

9.3 基于熵的正则化

对未 label 数据进行估计后得到概率。根据交叉熵定义，若 x 属于某一类的概率越大，交叉熵越小。交叉熵（对于未 label 数据）：注意对于分类问题，计算出来的 y 为概率，0 1 之间，因此交叉熵为正值。

$$E(y^u) = -\Sigma y^u ln(y^u)$$

定义损失函数

$$L = \Sigma C(y^r, \hat{y}^r) + \lambda \Sigma E(y^u)$$

此函数可微分，通过梯度下降训练求解。

9.4 半监督 SVM（支持向量机）

枚举所有未 label 的可能分布。对每一种分法都进行 SVM。找到具有最大间隔和最小误差的那一种分法。

9.5 光滑性假设

假设相似的 x 具有相同的 y。那么如何定义这个相似性。

x_1 与 x_2 在一个高密度区接近（有一条高密度路径）就说明相似。

集群然后标记。

9.6 基于图的方法

如何知道两个 x 有一条高密度路径。构造图，评估。

定义相似度，把数据点之间相连的边加上去，加边方式一是最近的 k 个；二是半径为 e 的范围内。根据相似度成比例地设置边的权重。相似度通过图进行传播。

引入平滑因子 S

$$S = \frac{1}{2} W_{i,j} (y^i - y^j)^2 = y^T L y$$

定义损失函数

$$L = \Sigma C(y^r, \hat{y}^r) + \lambda S$$

10 第九章-无监督学习

无监督学习：自己学，无 label 数据。一般用于机器阅读/自己理解某个名词/创作。一种是只有函数输入：聚类、数据降维，化繁为简。一种是只有函数输出，无中生有。

无监督学习目前所做的操作就是为了降维提取（PCA，聚类）之后可再升维，可以用来评估相似性，没有 y 值的意义，既不是分类也不是回归，just 提取一个 x 或评估两个 x。

10.1 聚类 cluster

聚类：将数据集中样本划分为若干个通常不相交的子集。

k-均值算法（k-means）：

1. 将样本划分为 K 个类别。
2. 每个类别选定聚类中心点 c_i。
3. 将样本归到离得最近的中心点上（划分类）。
4. 计算每个类中样本的均值，根据均值移动调整中心点位置，重复 3，4 直至合适的聚类生成。

分层凝聚聚类 HAC：从上到下，从下到上依次划分/合并生成聚类，最终形成一个树结构。

分布式表示：聚类是只属于某个类，分布式表示则是属于每个类的概率。

10.2 PCA 降维：

如何降维：1. 特征选择（不好）2. 主成分分析 PCA（投影）$z = Wx$，一系列 w_i，一系列 z_i。投影矩阵 W 是正交矩阵，其中每一个向量为单位向量且是 x 协方差矩阵的特征向量，且互相正交。λ_i 是特征值。

PCA 达到的效果就是 decorrelation 去关联。

10.2.1 由最大方差原理出发

使投影后的 z_i 方差最大。其中 z_1 最大，依次类推。

$$Var(z_i) = \frac{1}{N}\sum(z_i - \overline{z_i})^2 \qquad |w_i|_2 = 1$$

推理得到 $Var(z_i) = (w_i)^T Cov(x) w_i$ 即找到 w_i 使得方差最大。

有 $Cov(z) = WCov(x)W^T$

协方差矩阵计算公式 $Cov(x) = \frac{1}{N}\sum(x - \overline{x})(x - \overline{x})^T$

10.2.2 由最小误差原理出发

分析基础成分 u_i,c_1 则是这些成分的数字图像。有 $x = \sum c_i u_i$。使得 $x - \overline{x}$ 最小。

SVD 奇异值分解来得到 c_i, u_i。

PCA 可以看作一个只有一层隐藏层且线性激活函数的神经网络。由此可引出自编码器 AE，ANN 人工神经网络。

10.3 Auto-encoder 自编码器

自编码器 AE：用于表征学习，有很多应用（可降维）。通过一个编码器 encoder 和一个解码器 decoder 进行 input 的降维在升维为 output。使 output 与 input 越相似越好。（感觉 encoder 降维是在化繁为简，decoder 升维是在无中生有，感觉）提取表征，特征。

应用一：去噪自编码器。可对 input 进行加噪，AE 后使得 output 与不加噪的越像越好，因此可以进行去噪。

应用二：特征解纠缠。encoder 后的 vector 可有很多方面的，包含不同方面的信息，将这些方面分开就得到了不同方面的表征信息。可用于语音转换技术，encoder 并进行特征解纠缠后 A 的内容与 B 的声音 decoder，得到 B 的声音说 A 的内容。

应用三：离散潜在表示。encoder 后有很多向量，制取其中最相似的 decoder。（VQVAE 矢量量化变分自编码器）。eg：对文档段落进行总结提炼。

应用四：decoder 部分可作为生成器，无中生有。

应用五：encoder 部分可作为压缩器，化繁为简，后面 decoder 再解压缩。

应用六：异常检测：对输入 input 进行异常检测，判断其相对于训练数据是否异常。方法：对正常的训练数据进行 AE，得到网络（model）。对 input 进入此 model 发现重建 loss 很大，说明不相似，input 异常。