海森矩阵A是实对称矩阵，因此可以将其分解成一组实特征值和特征向量的正交。（∧ = X'AX,X是A的特征向量矩阵，

∧是特征值对角阵).见P87

凸优化算法只对凸函数适用——海森矩阵处处半正定的函数（f''(x)>=0）

无监督学习 ：p(x) 有监督学习:p(y|x) P100

wx+b,与 wx；不通过添加偏置参数b，仍然可以使用仅含权重w的模型，但是x要添加一项永远为1的元素。对应于额外1

的权重起到了偏置参数的作用。

决定机器学习算法效果是否好的因素：

1.降低训练误差（欠拟合） 2.缩小训练误差和测试误差的差距（过拟合） 。 P106

模型的容量是指其拟合各种函数的能力。容量不足的模型不能解决复杂任务。容量过高的模型，有可能会过拟合

假设空间：即被选为解决方案的学习算法函数集。

奥卡姆剃刀：同样能够解释已知观测现象的假设中，应该挑选“最简单”的那一个。

统计学习理论中最重要的结论阐述了训练误差和泛化误差之间差异的上界随着模型容量增长而增长，但随着训练样本增多而下降。

VC维越大，容量越大

Hoeffding 不等式： P(|X-EX|>=&)<=exp(-2&^2\*n^2)

假设预先知道，从真实分布p(x,y)预测而出现的误差被称为贝叶斯误差。

模型固定后，训练集越大，训练错误越大（数据量越大，越难拟合）,同时，测试错误会减小。P118 原版

表示对函数的偏好是比增减假设空间的成员函数更一般的去控制模型容量的方法。（正则化与多种模型选择比较）P112

正则化是指我们对学习算法所做的趋向于降低泛化误差而非训练误差的修改。

将机器学习问题转化成优化问题的最简单方法是最小化训练集上的期望损失。这意味着用训练集上的经验分布代替真实分布

观察model complexity penalty项，可以看到，h越大，model complexity penalty就会越大。N越大，model complexity penalty则会越小。

大致上讲，越复杂的模型有着越大的h（VC dimension），所以为了使得模型有着好的generalization，需要有较大的N来压低

model complexity penalty。 这就是为什么深度学习的模型需要大量的数据来训练，否则模型的generalization会比较差，也就是过拟合

加入权重衰减前 w<- w − ϵ∇wJ(w; X, y). 加入权重衰减后 w ← (1 − ϵα)w − ϵ∇wJ(w; X, y).

在加入权重衰减后会修改学习规则——在每一步执行通常的梯度更新之前对权重向量乘以一个常数因子以收缩权重向量。

线性回归的代价函数是平方误差之和：(Xw − y)⊤(Xw − y).解为w = (X⊤X)−1X⊤y

添加L2正则项后，目标函数变为 (Xw − y)⊤(Xw − y) + αw⊤w. 解为w = (X⊤X + αI)−1X⊤y.

不同点在于在对角线上加了a。这个矩阵的对角线对应于每个输入特征的方差（海塞矩阵的对角线元素是不同特征的方差）。

L2正则化能让学习算法“感知”到具有较高方差的输入x，因此与输出目标的协方差较小的特征的权重将会被收缩。

如果对数据进行PCA预处理，那么海塞矩阵就是一对角阵P199

伪逆的一个定义： +X= limα↘0(X⊤X + αI)−1X⊤.

对输入数据、权重、输出目标添加噪声

Bagging是通过结合几个模型降低泛化误差的技术。主要想法是分别训练几个不同的模型，然后让所有模型表决测试样例的输出。

Bagging涉及构造k个不同的数据集。每个数据集与原始数据集具有相同数量的样例，但从原始数据集中有替换采样构成。

机器学习算法区别于一般优化算法的一方面是其目标函数通常可以分解为关于训练样本的总和。

机器学习优化算法通常使用整个损失函数中的一部分项去更新其参数。（随机梯度）

使用整个训练集的优化算法被称为批量（batch）或确定下（deterministic）梯度算法，术语"批量梯度下降"指使用全部训练集

每次只使用一个单一样本的优化算法有时被称为随机或在线算法。术语“在线”通常是指从连续产生样本的数据流中抽取样本的

情况，而不是从一个固定大小的训练集中遍历多次采样的情况。

如果一个足够大的训练集可以唯一确定一组模型参数，那么该模型被称为可辨认的

这些模型可辨认性问题意味着神经网络损失函数具有非常多，甚至无穷的局部极小解。然而，所有这些由于不可辨识性问题（如权重对称性）

产生的局部极小解都有相同的损失函数值。因此，这些局部极小解并非是非凸所带来的问题。P235

一种能够排除局部极小值是主要问题的检测方法是画出梯度范数随时间的变化。如果梯度范数没有缩小到一个微小的值，那么该问题既不是

局部极小值，也不是其他形式的临界点。

对于很多高维非凸函数而言，局部极小值（或极大值）事实上都远少于另一类梯度为零的点（偏导的乘积导致消失的梯度）：鞍点。鞍点附近的某些点比鞍点有更大的损失，

而其他点则有更小的损失。在鞍点处，海森矩阵同时具有正负特征值。

多类随机函数表现出以下性质：低维空间中，局部极小值很普遍。在更高维空间中，局部极小值很少，而鞍点则很常见

对于牛顿法而言，显然鞍点会是一个问题。梯度下降旨在朝“下坡路”移动，而非明确寻求临界点。

而牛顿法的目标是寻求梯度为零的点。没有适当的修改，牛顿法会跳进一个鞍点。高维空间中鞍点的扩散或许解释了为什么

二阶方法不能成功替换梯度下降用于神经网络训练中。

P248我们几乎总是初始化模型的权重为高斯或均匀分布中随机抽取的值。

关于如何初始化网络，正则化和优化有非常不同的视角。优化视角建议权重应该足够大以成功传播信息，但正则化希望其小一点。

标准初始化，Wi,j~U(-sqrt(6/(m+n),sqrt(6/(m+n)))),m个输入，n个输出

P254 牛顿方法（求梯度为零的点）

牛顿参数更新规则：

@=@-H-1∇J(@)

牛顿放只适用于海塞矩阵H是正定的情况，在深度学习中，目标函数的表明通常有非凸的很多特点，如鞍点。

通常可通过正则化海塞来避免。常用的正则化技巧包含在海森矩阵对角线上增加常数a，正则化更新为

@=@-(H+aI)-1∇J(@)

在直接训练目标模型求解目标问题之前，训练简单模型求解简化问题的方法统称为预训练。

设计有助于优化的模型P263

在实践中， 选择一族容易优化的模型比使用一个强大的优化算法更重要。神经网络学习在过去30年的大多数进步主要来自于改变

模型族，而非改变优化过程。1980年代用于训练神经网络的带动量的随机梯度下降，仍然是现代神经网络应用中的前沿算法。

4.3基于梯度的优化方法：

假设f(x)=1/2\*x^2;so,f'(x) = x;

set x0= 2,a=0.5;

then f(x0)=2; f'(x0)=2; x=x0-a\*f'(x0)=2-0.5\*2=1;return last setp

鞍点（拐点）左右两端二阶导异号

一阶导（梯度）为0的点可能是极值点，也可能是鞍点()

最速下降建议新的点为 x′ = x - ϵ∇xf(x)

其中ϵ是学习速率，是一个确定步长大小的正标量。有种方法是根据几个ϵ计算f(x-ϵ∇xf(x)),并选择其中能产生最小目标函数值的ϵ.

这种策略被称为线搜索。

如果我们有一个函数f:Rm->Rn,f的雅可比矩阵J∈Rn x m,定义为J(i,j) = ∂f(x)i/∂ xj

二阶导数告诉我们一阶导数将如何随着输入的变化而改变。二阶导数是对曲率的衡量。

海森矩阵H等价于梯度的雅可比矩阵。

海森矩阵H(f)(x ,通常海森矩阵是对称的。因为海森矩阵是实对称的，我们可以将其分解成一组实特征值和特征向量的正交。在特定方向d上的二阶导数可以写成.当d是H的一个特征向量时，这个方向的二阶导数就是对应的特征值。对于其他的方向d,方向二阶导数是所有特征值的加权平均，权重在0和1 之间，且与d夹角越小的特征向量有更大的权重。

我们可以通过(方向)二阶导数预期一个梯度下降步骤能表现得多好。我们在当前点处作函数f(x)的近似二阶泰勒级数：

f(x)+(x-+1/2(x-(4.8)

如果我们使用学习速率,那么新的点x将会是。代入上述的近似，可得

f()-+1/2 (4.9)

其中有3项： 函数的原始值，函数斜率导致的预期改善，函数曲率导致的校正。当最后一项太大时，梯度下降实际上是可能向上移动的。当为正时，此时通过计算可得，使近似泰勒级数下降最多的最优步长为

=(4.10)

在临界点处(∇xf(x)=0)，通过检验海森矩阵的特征值来判断该临界点是一个局部极大点，局部极小点还是鞍点。当海森矩阵是正定的(所有特征值都是正的)，则该临界点是局部极小点。因为方向二阶导数在任意方向上都是正的。 如果海森矩阵的特征值中至少一个是正的且至少一个是负的，那么x是f某个横截面的局部极大点，却是另一个横截面的局部极小点。

式（4.8）的临界点为： -H(f g(4.12)

当f是一个正定二次函数时，牛顿法只要应用一次4.12就能直接跳到函数的最小点。

曲率越大，表示弯曲的程度越厉害

多维情况下，单个点处每个方向上的二阶导数是不同的。海森的条件数衡量这些二阶导数的变换范围。当海森的条件数很差时，梯度下降法也会表现得很差。这是因为一个方向上的导数增加的很快，而在另一个方向上增加的很慢。因此步长必须足够小，以免冲过最小而向具有较强的正曲率方向上升，这通常意味着步长太小，以至于在其他较小曲率的方向上进展不明显。

5.9随机梯度下降：

梯度下降需要计算

∇ΘJ(Θ)= 1/m\*∇ΘΣmL(x(i); y(i); Θ)

i=1

随机梯度下降的核心是，梯度是期望。期望可用小规模的样本近似估计。具体而言，在算法的每一步，我们从训练集中均匀抽出一

minibatch样本B = {x(1)..,x(m')}.minibatch的数目m'通常是一个相对较小的数，从一到几百

梯度的估计可表示为g = 1/m'\*∇ΘΣm'L(x(i); y(i); Θ) 使用来自minibatch的样本。然后，随机梯度下降算法使用如下的

i=1

梯度下降估计 Θ<- Θ-ϵg,其中，ϵ是学习速率

6.4.1 通用近似性质和深度

普遍近似定理表明。一个前馈神经网络如果具有线性输出层和至少一层具有一种“挤压”性质的激活函数的隐藏层，只要给予网络足够数量的隐藏单元，它可以以任意的精度来近似任何从一个有限维空间到另一个有限维空间的Borel可测函数。

八.深度模型中的优化

8.1 学习和优化有什么不同

通常，损失函数可写为训练集上的平均，如

J(Θ) = (8.1)

其中是经验分布。式8.1定义了训练集上的目标函数。通常，我们更希望最小化期望取自数据生成分布Pdata,而不仅是有限个训练集上的对应目标函数：

= (8.2)

8.1.1 经验风险最小化

机器学习算法的目标是降低式8.2所示的期望泛化误差。这个数据量被称为风险。

将机器学习问题转换成优化问题的最简单方法是最小化训练集上的期望损失。这意味着用训练集上的经验分步(x,y)替代真实分布p(x,y).如此，我们将最小化经验风险.

=. 其中m表示训练样本的数目

基于最小化如上平均训练误差的训练过程被称为经验风险最小化。

经验风险最小化容易过拟合。最有效的现代优化算法是基于梯度下降的。

8.1.2 替代损失函数和提前终止

8.1.3 （小）批算法

机器学习优化算法通常使用整个损失函数中的一部分项去更新其参数

优点：牺牲少部分精度，大大提高计算速度

批量梯度下降指使用全部训练集，而术语“批量”单独出现时指一组样本。

每次只使用一个单一样本的优化算法有时被称为随机或在线算法。

用一批数据的方法被称为minibatch随机方法。

minibatch大小通常由以下原因驱动：

1.更大的批量会计算更准确的梯度估计，但是回报却是小于线性的。

2.多核架构通常未充分利用极小批量。这促使我们使用一些绝对最小批量，低于这个值的小批量处理不会减少处理时间。

小批量是随机抽取的这点也很重要(通常会将数据随机打乱，然后每次迭代都会再随机取.因为多次迭代相同顺序会造成有偏)

很多机器学习上的优化问题都可以分解成并行地计算不同样本上单独的更新。、

8.2 神经网络的优化挑战

通常，机器学习会小心设计目标函数和约束，以确保优化问题是凸的，从而避免一般优化问题的复杂度。

8.2.1 病态

在优化凸函数时，会遇到一些挑战，其中最突出的是海森矩阵H的病态。 病态问题一般被认为存在于神经网络训练过程中，病态体现在随机梯度下降会“卡”在某些情况，此时即使很小的更新步长也会增加损失函数。

8.2.2 局部极小值

几乎所有的深度模型基本上都会有非常多的局部极小值，然而，我们会发现这并不是一个严重的问题。在局部极小值点处只有正特征值。此时嗨森矩阵是正定矩阵，而梯度为0，此时损失值必然增加。

如果一个足够大的训练集可以唯一确定一组模型参数，那么该模型被称为可辨认的。

一种能够排除局部极小值是主要问题的检测方法是画出梯度范数随时间的变化。当然许多并非局部极小值的结构也具有很小的梯度。

8.2.3高原,鞍点和其他平坦区域

局部极小值(或极大值)事实上都远小于另一类梯度为零的店:鞍点。在鞍点处，海森矩阵同时具有正负特征值。我们可以将鞍点视为损失函数某个横截面上的局部极小值，同时也可以视为损失函数某个横截面上的局部极大值。

多类随机函数表现出以下性质：低维空间中，局部极小值很普遍。在更高维空间中，局部极小值很少，而鞍点则很常见。

当我们达到损失较低的区间时，海森矩阵的特征值维正的可能性更大。和抛硬币类比，这意味着如果我们处于低损失的临界点时，抛掷硬币正面朝上n次的概率更大。

牛顿法的目标是寻求梯度为零的点。高维空间中鞍点的扩散或许解释了为什么二阶方法不能成功替换梯度下降用于神经网络中。Dauphin介绍了二阶优化的无鞍牛顿法

8.2.4 悬崖和梯度爆炸

多层神经网络通常有像悬崖一样的斜率较大区域，这是由于几个较大的权重相乘导致的。幸运的是，可以使用第10.11.1节介绍的启发式梯度截断来避免其主要缺点。

8.2.5 长期相关性

当计算图变得非常之深时，反复使用相同的参数产生了尤为突出的困难。

例如，假设某个计算图中包含一条重复与矩阵W相乘的路径。那么t步后，相当于和相乘。假设W有特征值分解W = (Vdiag() = Vdiag( （8.11）

当特征值不在1附近时，若在量级上大于1则会膨胀到很大；若小于1则会收缩到很小。

8.2.6 非精确梯度

8.2.7 局部和全局结构间的弱对应

许多现有研究方法在求解局部结构复杂的问题时，旨在寻求良好的初始点，而不是开发非局部范围更新的算法。

8.2.8 优化的理论限制

8.3 基本算法

8.3.1 随机梯度下降

随机梯度下降(SGD)及其变种很可能是一般机器学习中用得最多的优化算法，特别是深度学习中。

SGD算法中一个关键参数是学习速率。在实践中，有必要随着时间的推移逐渐降低学习速率

保证SGD收敛的一个充分条件是

= (8.12) ,且 < (8.13)

实践中，一般会线性衰减学习速率到第次迭代：

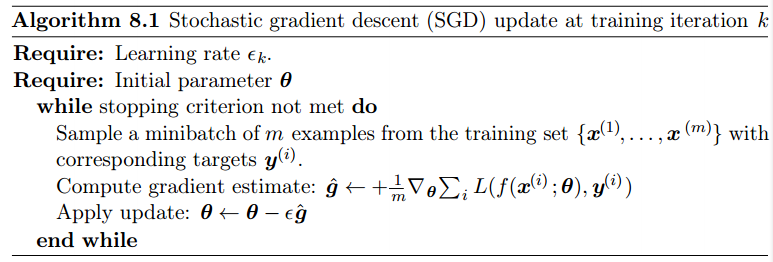
=（1-）

其中= k/。在使保持常数。 通常应设为大约1%的，

SGD和相关小批量或在线基于梯度的优化的最重要的性质是每一步更新的计算实践不会随着训练样本数目而增加。即使训练样本数目非常大，这也能收敛。对于足够大的数据集，SGD可能会在处理整个训练集之前就收敛到最终测试集误差的某个固定容差范围内(训练集之间存在相似性)

研究优化算法的收敛率，一般会衡量额外误差：J()-min J(),即当前损失函数超出最低可能损失的量。Cramer-Rao界限指出，k步迭代后的泛化误差的下降速度不会快于O(1/k)

了解SGD更多的信息，请参看Bottou（1998）



8.3.2 动量

动量方法(Polyak,1964)旨在加速学习，特别是处理高曲率，小但一致的梯度，或是带噪音的梯度.动量算法积累了之前梯度指数级衰减的移动平均,并且继续沿该方向移动.

动量方法旨在解决两个主要问题:1.poor conditioning of the Hessian matrix and variance in the stochastic gradient.

从形式上看，动量算法引入了变了v充当速度的角色——它代表参数在参数空间移动的方向和速度。速度被设为负梯度的指数衰减平均。更新规则如下：

, (8.15)

(8.16)

一般取值为0.5,0.9和0.99，一般初始值是一个较小的值，随后会慢慢变大。

8.3.3 Nesterove加速梯度算法

8.4 参数初始化化策略

目前对于初始点如何影响泛化的理解是相当原始的，几乎没有提供如何选择初始点的任何指导。也许完全确知的唯一特性是初始参数需要在不同单元间"破坏对称性"。

更大的初始权重有更强的破坏对称性的作用，有助于避免冗余的单元。但如果初始权重太大，那么会在前向传播或反向传播中产生迅速膨胀的值。特别是在很多层的神经网络和循环网络中，很大的权重也可能导致混论。较大的权重也会产生使得激励函数饱和的值，导致饱和单元的梯度完全丢失。

有些启发式方法可用于选择权重的初始大小。一种初始化m个输入和n输出的全连接层的权重的启发式方法是(Glorot 2011a)

(8.23)

该式由不含非线性的链式矩阵乘法网络的假设推导得出。现实的神经网络显然会违反这个假设，但很多设计于线性模型的策略在其非线性对应中效果也不错

数值范围准则的一个缺点是，设置所有的初始权重具有相同的标准差，会使得层很大时每个单一权重会变得极其小。

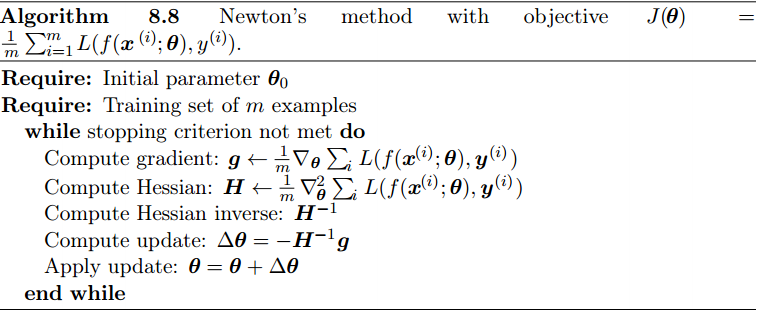
其他参数相对于权重的初始化，通常更容易

8.5 自适应学习率的算法

AdaGrad RMSProp Adam

8.6 二阶近似方法

8.6.1 牛顿法



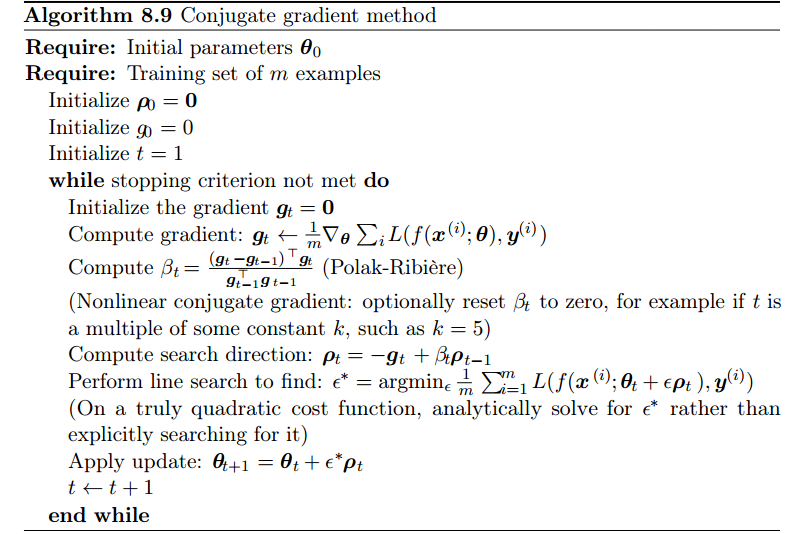
8.6.2 共轭梯度

共轭梯度是一种通过迭代下降的共轭方向以有效避免海森矩阵求逆计算的方法。

两个方向 ***d****t* 和 ***d****t−*1 被称为共轭的，如果 ***d****⊤ t* ***Hd****t−*1 = 0，

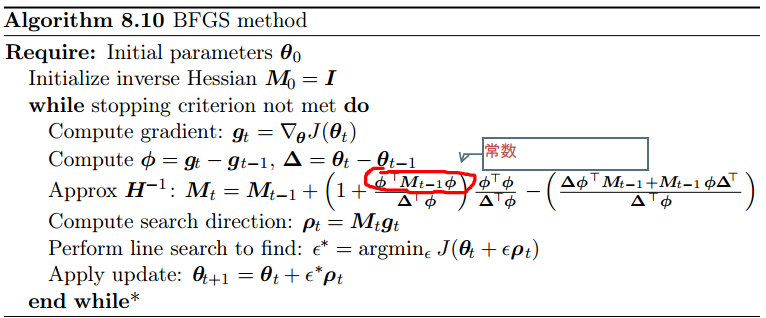
对于二次曲面而言，共轭方向确保梯度沿着前一方向大小不变。因此，我们在前一方向上仍然是极小值。其结果是，在k-维参数空间中，共轭梯度只需至多k次线性搜索就能达到极小值。

共轭梯度法无需计算二阶导数，且在实践中使用非线性共轭梯度算法训练神经网络是合理的



8.6.3 BFGS

BFGS虽然不需要计算海森计算的逆，但必须存储海森逆矩阵，需要O()的存储空间，使得BFGS不适用于大多数具有百万级参数的现代深度学习模型。



8.7 优化技巧和元算法

8.7.1 batch normalization

标准化H(H是需要标准化的某层的minibatch激励函数)

H'= （8.35）

8.72 坐标下降

在某些情况下，将一个优化问题分解成几个部分，可以更快地解决原问题。如果我们最小化f(x)相对于某个单一变量相对于另一个变量,等等，反复循环所有的变量，我们会保证达到(局部)极小值、这种做法被称为坐标下降。

当优化一组变量明显比优化所有变量效率更高时，坐标下降最有意义。

当一个变量的值很大程度地影响另一个变量的最优值时，坐标下降不是一个很好的方法。

8.7.3 Polyak 平均

+(1-) (8.39)

8.74 监督提前训练

在直接训练目标模型求解目标问题之前，训练简单模型求解简化问题的方法统称为预训练。

贪心算法将问题分解成许多部分，然后独立地在每个部分求解最优值。

提前训练算法，特别是贪心提前训练，在深度学习中是普遍存在的。

8.7.5 设计有助于优化的模型

改进优化的最好方法并不总是改进优化算法。相反，深度模型中优化的许多改进来自于设计易于优化的模型。

8.7.6 连续方法和课程学习

第九章 卷积神经网络

卷积网络，也叫作卷积神经网络(CNN),是一种专门用来处理具有类似网格结构的数据的神经网络。卷积是一种特殊的线性运算。卷积网络是指那些至少在网络的一层中使用卷积运算来替代一般的矩阵乘法运算的神经网络。

9.1卷积运算

卷积

s(t)= (9.1)

卷积运算通常用星号表示：

s(t)=(x\*w)(t) (9.2)

在卷积神经网络的术语中， 第一个参数（在上式中，函数x）叫做输入，第二个参数(函数w)叫做核，输出有时被称为特征映射。

多个维度进行卷积运算。例如，如果把二维的图像I作为输入，我们也相应的需要使用二维的核K：

S(i,j)=(I\*K)(i,j)= (9.4)

卷积是可以交换的，因此可以等价的写为:

S(i,j)=(K\*I)(i,j)= (9.5)

卷积运算可交换性的出现是因为我们相对输入翻转了核

互相关函数，和卷积运算几乎一样但是并不翻转核：

S(i,j)=(I\*K)(i,j)= (9.6)

许多机器学习的库使用互相关函数但是叫它卷积。在本书中我们遵循把两种运算都叫做卷积的这个传统

9.2动机

传统的神经网络使用矩阵乘法来建立输入与输出的连接关系。其中，参数矩阵的每一个独立的参数都描述了每个输入单元与每一个输出单元间的交互。然而，卷积神经网络具有稀疏交互（稀疏权重）的特征。这通过使得核的规模远小于输入的规模来实现。

参数共享是指在一个模型的多个函数中使用相同的参数；。

在传统的神经网络中， 当计算一层的输出时，权值矩阵的每一个元素只使用一次，当它乘以输入的一个元素后就再也不会用到了。在卷积神经网络中，核的每一个元素都作用在输入的每一位置上。卷积运算中的参数共享保证了我们只需要学习一个参数几何，而不是对于每个位置都要学习一个单独的参数集合。

如果函数f(x)和g(x)满足f(g(x))=g(f(x))，我们就说f(x)对于变换g具有等变性。如果令g是输入的任意平移函数，那么卷积函数对于g具有等变性。

9.3 池化

卷积神经网络的卷积层通常包含三级。在第一级中，卷积层并行地进行多个卷积运算来产生一组线性激活函数，在第二级中，非线性的激活函数如修正线性单元函数等作用在第一级中的每一个线性输出上，这一级有时也被称为探测级。在第三极中，我们使用池化函数函数来更进一步地调整卷积层的输出。

局部平移不变形是说当我们把输入平移一微小的量，大多数通过池化函数的输出值并不会发生改变。局部平移不变性是一个很重要的性质，尤其是当我们关心某个特征是否出现而不关心它出现的具体位置时。

9.4卷积与池化作为一种无限强的先验

我们可以把卷积的使用当做是对网络中的一层的参数引入了一个无限强的先验概率分布。这个先验是说该层需要学得的函数具有局部连接和平移等价性。类似的，使用池化也是一个无限强的先验：每一个单元都具有少量平移的不变性。

9.5 基本卷积函数的变体

带有单个核的卷积只能提取一种类型的特征。

在任何卷积神经网络的应用中都有一个重要性质，那就是能够隐含地对输入V用零进行填充使得它加宽。如果没有这个性质，表示的宽度在每一层就会缩减，缩减的幅度是比核少一个像素这么多。

有三种零填充设定的情况值得注意。第一种是无论怎样都不适用零填充的极端情况，并且卷积核只允许访问那些图像中能够完全包含整个核的位置。Matlab中，这称为有效卷积。输入的宽度为m，核的宽度是k，那么输出的宽度就会变成m-k+1

第二种特殊的情况是只进行足够的零填充来保持输出和输入具有相同的大小。在Matlab中，这称为相同卷积。

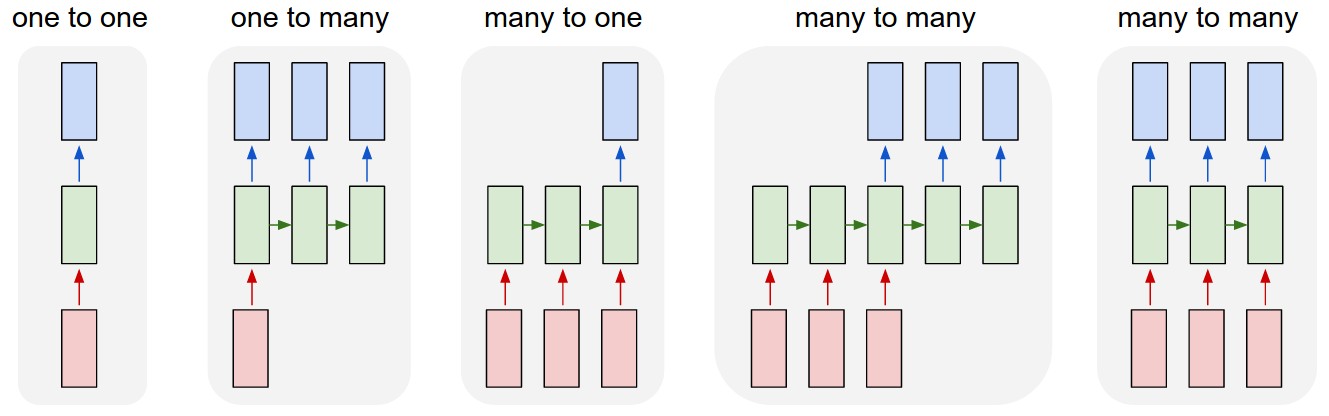
第三种情况是对每个方向上呗访问了k次的像素进行足够的零填充，最终输出的图像宽度为m+k-1.在Matlab中称为全卷积

第十章 序列建模：循环和递归网络

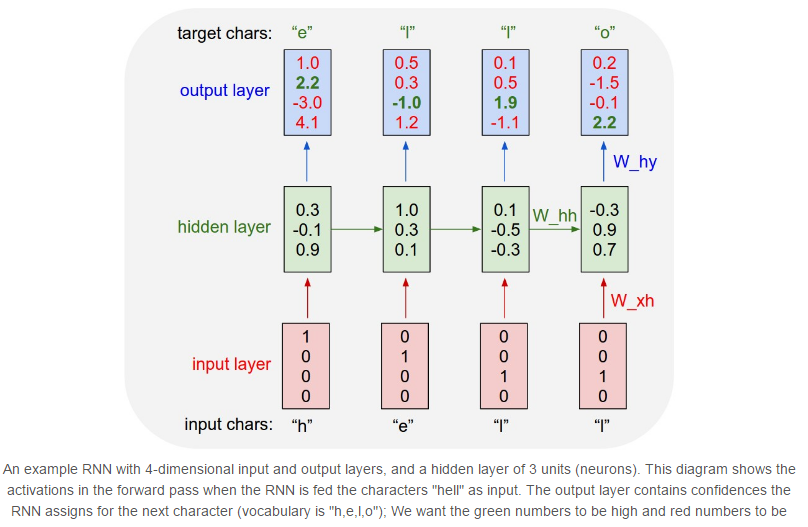
The Unreasonable Effectiveness of Recurrent Neural Networks( Andrej Karpathy blog)

{

CNN,DNN限制大，输入必须是固定长度的向量，输出也是一固定长度的向量，并且这些模型的层数等是固定的。



Each rectangle is a vector and arrows represent functions (e.g. matrix multiply). Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state (more on this soon). From left to right: **(1)** Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification). **(2)** Sequence output (e.g. image captioning takes an image and outputs a sentence of words). **(3)** Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment). **(4)** Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French). **(5)** Synced sequence input and output (e.g. video classification where we wish to label each frame of the video). Notice that in every case are no pre-specified constraints on the lengths sequences because the recurrent transformation (green) is fixed and can be applied as many times as we like.



standard Softmax classifier (also commonly referred to as the cross-entropy loss)

}

循环神经网络(RNN)是一类用于处理序列数据的神经网络。就像是卷积网络是专门用于处理网格化数据X(如一个图形)的神经网络，循环神经网络是专门用于处理序列,...的神经网络。大多循环神经网络也能处理可变长度的序列。

一个相关的想法是跨越1维时间序列的卷积。这种卷积方法是时延神经网络的基础。

10.1 展开计算图

计算图是一种形式化一组计算结构的方式。

考虑动态系统的经典形式：

f() (10.1)

其中称为系统的状态。s在时刻t的定义需要参考时刻t-1时同样的定义，因此(10.1)是循环的。

现在，考虑由外部信号驱动的动态系统

f() (10.4)

为了表明状态是网络的隐藏单元，我们使用变量h代表状态来重写式(10.4):

f() (10.5)包含(,....)

当循环网络被训练位根据过去预测未来，网络通常要学会使用作为过去序列(直到t)与任务相关方面的有损摘要。

我们可以用一个函数f()=() (10.6)

函数将全部的过去序列()作为输入来生成当前状态。但是展开的循环架构允许我们将分解为函数f重复的应用，因此，展开过程引入两个主要优点：

1. 无论序列的长度，学习好的模型始终具有相同的输入大小，因为它指定的是从一种状态到另一种状态的转移，而不是在可变长度的历史状态上操作。

2、我们可以在每个时间步使用具有相同参数的相同转移函数f。

这两个因素使得学习在所有时间步和所有序列长度上操作的单一模型f是可能的，而不需要在所有可能时间步学习独立的模型.学习单一的共享模型允许泛化到没有见过的序列长度。

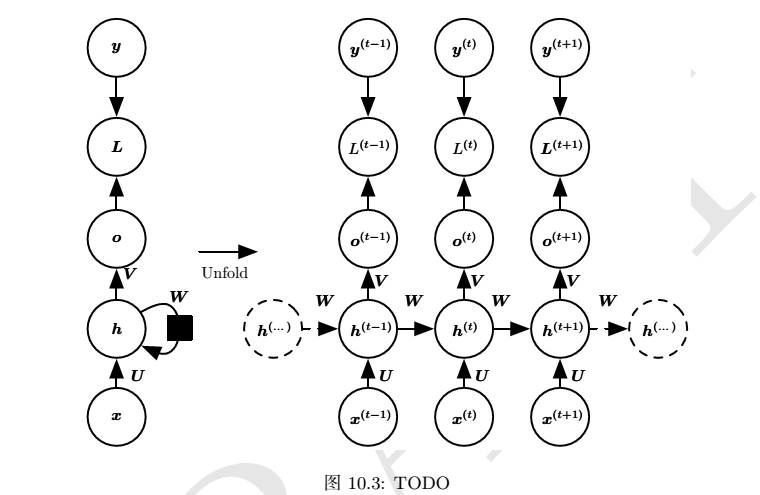
10.2 循环神经网络

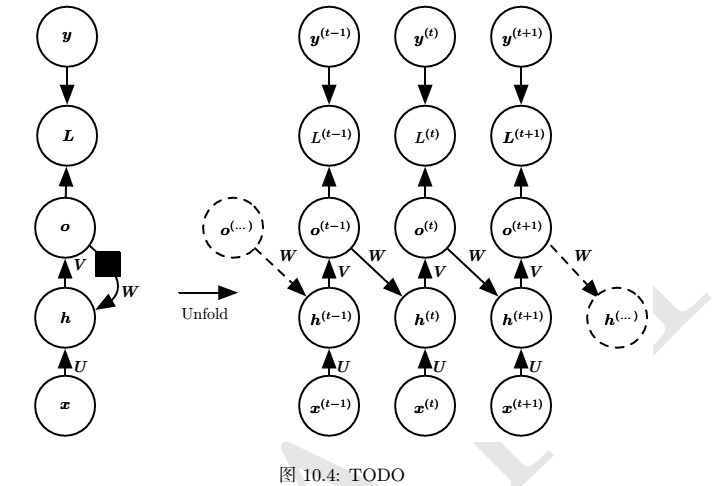
循环神经网络中一些重要的设计模式包括以下几种：

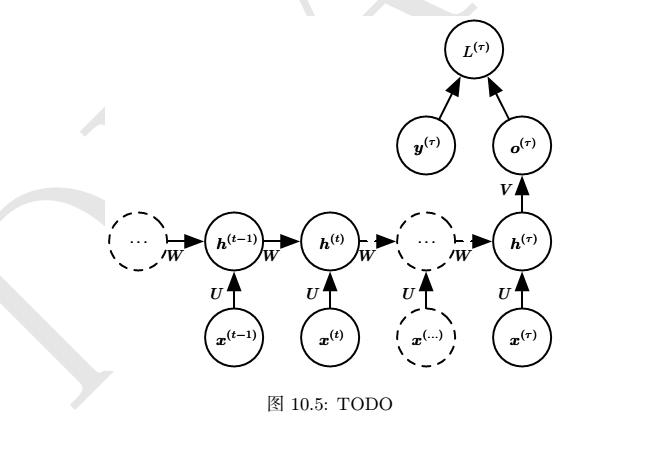
1.每个时间步都有输出，并且隐藏单元之间有循环连接的循环网络，如图10.3

2.每个时间步都产生一个输出，只有当前时刻的输出到下个时刻的隐藏单元之间有循环连接的循环网络，如图10.4

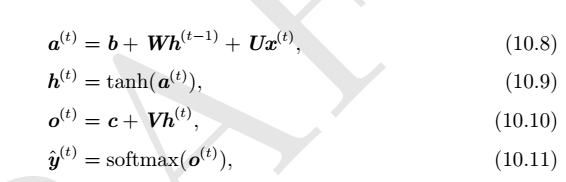
3.隐藏单元之间存在循环连接，但读取整个序列后产生单个输出的循环网络，如图10.5



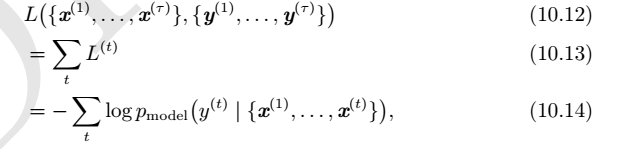




现在来研究图10.3 RNN的前向传播公式。更新方程为：



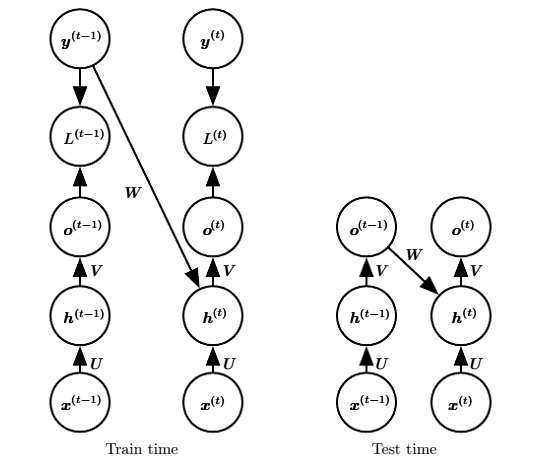
与x序列配对的y的总损失就是所有时间步的损失之和。例如



通过时间反向传播（BPTT）（10.3）

10.2.1 Teacher Forcing和输出循环网络

（图10.4）仅在一个时间步的输出和下一个时间步的隐藏单元间存在循环连接的网络确实没有那么强大，但消除隐藏到隐藏循环的优点在于，任何基于比较时刻t的预测和时刻t的训练目标的损失函数，所有时间步都解放了。因此训练可以并行化，在各时刻t分别计算梯度。因为训练集提供输出的理想值，所以没有必要计算前一时刻的输出



10.2.2 计算循环神经网络的梯度

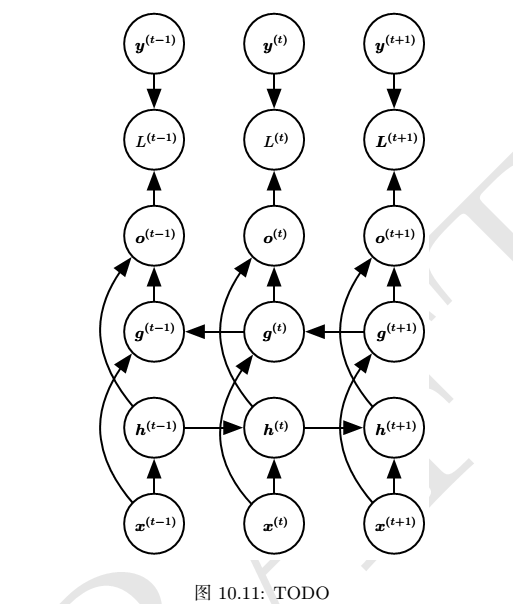
计算循环神经网络的梯度是容易的，可以简单的将反向传播算法应用于展开的计算图

，不需要特殊化的算法。

10.3 双向RNN

在许多应用中，我们要输出的预测可能依赖于整个输入序列。（未来影响现在）。例如，在语音识别中，由于协同发音，当前声音作为音素的正确解释可能取决于未来几个音素，甚至潜在的可能取决于未来的几个词。

双向RNN结合时间上从序列起点开始移动的RNN和另一个时间上从序列末尾开始移动的RNN。图10.11展示了典型的双向RNN，其中代表通过时间向前移动的子RNN的状态，代表通过时间向后移动的子RNN的状态。

****

10.4基于编码-解码的序列到序列架构

本节讨论如何训练RNN,使其将输入序列映射到不一定等长的输出序列。前一系统是对另一个机器翻译系统产生的建议进行评分，而后者使用独立的循环网络来生成翻译。这些作者分别将该架构称为编码-解码或序列到序列架构，如图10.12所示。这个想法非常简单：

(1)编码器RNN处理输入序列。编码器输出上下文C（通常是最终隐藏状态的简单函数）。

(2)解码器RNN则以固定长度的向量为条件产生输出序列Y=(),这种架构对比本章前几节提出的架构的创新之处在于长度和可以彼此不同，在一个序列到序列的架构中，两个RNN共同训练以最大化logP()

10.5 深度循环网络

大多数RNN中的计算可以分解成三块参数及其相关的变换：

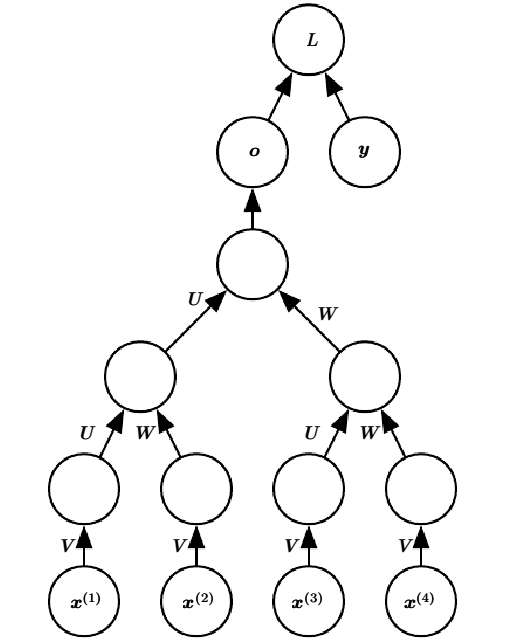
1.从输出到隐藏状态。

2.从前一隐藏状态到下一隐藏状态，以及

3.从隐藏状态到输出

10.6 递归神经网络

递归神经网络代表循环网络的另一个扩展，它被构造为深的树状结构而不是RNN的链状结构，因此是不同类型的计算图。对于递归网络的典型的计算图如图10.14所示。



Bottou描述了这类网络的潜在用途——学习推论。递归网络已成功地用于输入是数据结构的神经网络，如自然语言处理和计算机视觉。 递归网络的一个明显优势是，对于相同的长度为的序列，深度可以急剧地从减小为O(log)，这可能有助于解决长期依赖。

10.7 长期依赖的挑战

循环神经网络所使用的函数组合有点像矩阵乘法。我们可以认为，循环联系

(10.36)

是一个非常简单的，缺少非线性激活函数和输入x的循环神经网络。

(10.36)

而当W符合下列形式的特征分解

W=Q (10.38)

其中Q蒸饺，循环性可进一步简化为：

Q (10.39)

即特征值提升到t次后，导致幅值不到一的特征值衰减到零，而幅值大于一的就会激增。任何不予最大特征向量对齐的的部分将最终被丢弃。学习长期依赖的问题仍是深度学习中的一个主要挑战。

10.8 回声状态网络(ESN)

从到的循环权重映射以及从到的输入权重映射是一个循环网络中最难学习的参数。研究者提出避免这种困难的方法是设定循环隐藏单元，使其能很好地捕捉过去输出历史，并且只学习输出权重。回声状态网络以及流体状态机分别突出了这种想法，后者是类似的。ESN和流体状态机都被称为储层计算，因为隐藏单元形成了可能捕获输入历史不同方面的临时特征池。

10.9 渗漏单元和其他多时间尺度的策略

处理长期依赖的一种方法是设计工作在多个时间尺度上的模型，使其某些部分在细粒度时间尺度上操作并能处理小细节，而其他部分在粗时间尺度上操作并能把遥远过去的信息更有效地传递过来。

10.9.1 时间维度的跳跃连接

增加从遥远过去的变量到目前变量的直接连接是得到粗时间尺度的一种方法。

10.9.2 渗漏单元和一系列不同时间尺度

获得导数乘积接近1的另一方式是设置线性自连接单元，并且这些连接的权重接近1.

线性自连接的隐藏单元可以模拟滑动平均的行为。这种隐藏单元称为渗漏单元。、

10.9.3 删除连接

10.10. 长短期记忆和其他门限RNN

实际应用中最有效的序列模型称为门限RNN，包括基于长短期记忆和基于门限循环单元的网络。

像渗漏单元一样，门限RNN想法也是基于生成通过时间的路径，其中导数既不消失也不爆炸。我们希望神经网络学会决定何时清楚状态，而不是手动决定。这就是门限RNN要做的事

10.10.1 LSTM(长短期记忆)

引入自循环的巧妙构思，以产生梯度长时间持续流动的路径是初始长短期记忆的核心贡献。其中一个关键拓展是使自循环的权重视上下文而定，而不是固定的。LSTM已经在许多应用中期的重大成功，如无约束手写识别，语音识别，手写识别，机器翻译，为图像生成标题和解析

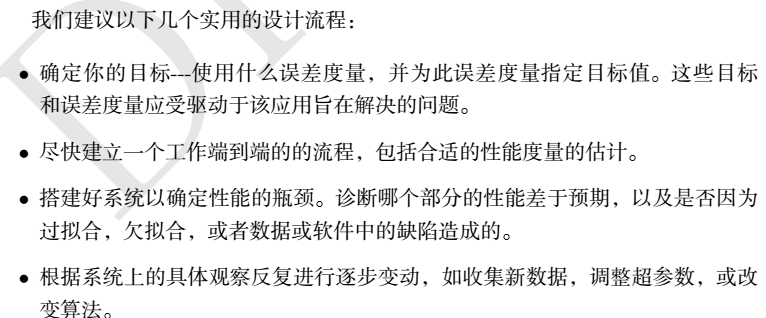
10.11 优化长期依赖

10.11.1 截断梯度

10.11.2 引导信息流的正则化

10.12 外显记忆

第十一章 实用方法



11.1 性能度量

确定你的目标，即用什么误差度量，是必要的第一步。

对于大多数应用而言，不可能实现绝对的零误差。即使有无限的训练数据，并且恢复了真正的概率分布，贝叶斯误差仍然定义了你能达到的最小错误率。这是因为你的输入特征可能无法包含关于输出变量的完整信息，或是因为系统可能本质上是随机的。

除了性能度量这个目标值之外，另一个重要的考虑是度量的选择。

PR曲线，y表示精度，x表示召回率。精度是模型报告的检测是正确的比例，而召回率则是真实事件被检测到的比例。

11.2 默认的基准模型

确认性能度量和目标后，任何实际应用的下一步是尽快建立一个合理的端到端的系统

11.3 决定是否收集更多数据

在第一个端到端的系统建立后，就可以去测量算法性能，并确定如何改进它。

怎样决定是否要收集更多的数据？

首先，确定训练集上的性能是否可接受。如果大的模型和仔细调试的优化算法没有效果，那么问题可能是训练数据的质量。该数据可能含太多噪音，或是可能不包含需要预测期望输出的正确输入。

如果训练集上的性能是可接受的，然后度量测试组上的性能。如果测试集上的表现也是可以接受的，那么就顺利完成了。如果测试集上的性能比训练集要差得多，那么收集更多的数据是最有效的解决方案之一。关键因素是收集更多数据和其他方法降低测试误差的成本和可行性，并且数据的数量预计能够显著提高测试集性能。

如果收集更多的数据是不可行的，那么改进泛化误差的唯一方法是改进学习算法本身，这将归为研究领域，而不是给应用实践者的建议。

11.4 选择超参数

有两种选择超参数的基本方法:手动选择和自动选择。手动选择超参数需要了解超参数做了些什么，以及机器学习模型如何取得良好的泛化。自动选择超参数算法大大减少了了解这些想法的需要，但它们往往需要更高的计算成本。

11.4.1 手动调整超参数

手动搜索超参数的主要目的是调整模型的有效容量以匹配任务的复杂性。有效容量受限于三个因素：模型的表示容量（具有更多层，每层有更多隐单元的模型具有较高的表达能力），学习算法成功最小化用于训练模型的成本函数的能力，以及代价函数和训练过程正规化模型的程度。

对于某些超参数而言，当泛化误差作为某个超参数的函数绘制出来时，通常会表现为U型曲线。超参数的值太大时会发生过拟合(如隐藏层)，太小也会发生过你和(最小的可允许权重衰减系数零对应着学习算法最大的有效容量。)

很多超参数是离散的，它们并不能够探索整个U形曲线。

学习率可能是最重要的超参数。它以一种更复杂的方式控制模型的有效容量---当学习率适合优化问题时，模型的有效容量最高，此时学习率既不是特别大也不是特别小。学习率关于训练误差具有U形曲线。

手动调整超参数时，不要忘记最终目标：测试集上的优异性能。

11.4.2 自动超参数优化算法

11.4.3 格点搜索

当有三个或更少的超参数时，常见的超参数搜索方法是格点搜索。通常，格点搜索大约会在对数刻度下挑选合适的值。通常重复进行格点搜索时，效果会最好。令人遗憾的是，由于格点搜索指数级增长计算代价，即使并行也无法提供令人满意的计算能力。

11.4.4 随机搜索

11.4.5 基于模型的超参数优化

11.5 调试技巧

一些重要的调试检测包括以下这些。

可视化模型的行为

可视化最严重的错误：大多数模型能够输出运行任务时的某种置信度量

使用训练和测试误差检测软件

比较反向传播导数和数值导数

监控激励函数值和梯度的直方图

第十二章 应用

12.1 大规模深度学习

深度学习的基本思想是建立在连接机制上的：尽管机器学习模型中的单个生物性的神经元或者说是单个特征不是智能的，但是大量的神经元或者特征作用在一起往往能够表现出智能。

12.1.1 快速的CPU实现。

在今天，神经网络用单台机器的CPU来训练被视为是不可取的。

12.1.2 GPU实现

许多现代的神经网络的实现是基于图形处理器。相比于CPU,GPU的极高的内存带宽称为了一个显著的优势。神经网络的训练算法通常并不包括分支运算和复杂的控制指令，所有更适合在GPU上训练。GPU的并行特性也是其一大优势。

由于相对简便的编程语言，强大的并行能力以及巨大的内存带宽，通用GPU是我们神经网络训练的理想平台。

12.1.3 大规模的分布式实现

分布式的推断是容易实现的，因为每一个输入的样本都可以在单个机器上运行。这也被称为是数据并行。

模型并行也是可行的，其中多个机器共同运行一个数据点，每个机器负责模型的一个部分。

训练中，数据并行在某种程度上说更难。梯度下降的标准定义完全是一个串行的过程。

在第t步的梯度是一个第t-1步所获得的参数的函数。 这个问题可以通过异步随机梯度下降来解决。在这个方法中，几个处理器的核共用了存有参数的内存。每一个核在无锁的情况下读取了这个参数，然后计算其对应的梯度，然后在无锁的状态下更新了这个参数。

12.1.4 模型压缩

减少推断所需要的开销中的一个关键的策略是模型压缩。模型压缩的基本思想是用一个更小的需要更少的内存和时间的模型取代原始的耗时的模型。

12.1.5 动态结构

12.1.6 深度网络的专用硬件实现

12.2 计算机视觉

12.2.1 预处理

将图像格式化为具有相同的比例严格上说是唯一一种必要的预处理。

12.2.1.1 对比度归一化

12.2.2 数据集增强

通过增加训练集的额外副本来增加训练集的大小，从而改进分类器的泛化能力。

12.3 语言识别

语音识别任务是将一段包含了自然语言发音的声音信号投影到对应的说话人的词序列上。自动语言识别(ASR)任务指的是构造一个函数(X) = （12.4）

使得它能够在给定语言序列X的情况下计算最有可能的y序列。其中是给定输入值X时对应目标y的条件分布。

从20世纪80年代到2009-2012年，最先进的语音识别系统是隐马尔可夫模型（HMMs）和高斯混合模型(GMMs)的结合.GMMs对语言特征和音位之间的关系建模，HMMs对音位序列建模。

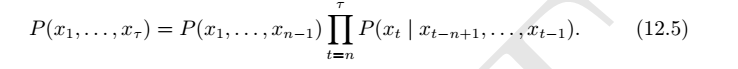
12.4 自然语言处理

自然语言处理让计算机使用人类语言，例如英语或法语。计算机程序通常读取和发出专门的语言，让简单的程序能够高效和明确的解析。（机器翻译）因为可能的词总数非常大，基于词的语言模型必须在极高维度和稀疏的离散空间上操作。为使这种空间上的模型在计算和统计意义上都高效，研究者已经开发了几种策略。

12.4.1 n-gram

语言模型定义了自然语言中的标记序列的概率分布。最早成功的语言模型基于固定长度序列的标记模型，称为n-gram.一个n-gram是一个包含n个标记的序列。

基于n-gram的模型定义给定前n-1个标记后的第n个标记的条件概率。该模型使用这些条件分布的乘积来定义较长序列的概率分布：



经典的n-gram模型特别容易引起维数灾难。

12.4.2 神经语言模型

神经语言模型（NLM)是一类设计用来克服维数灾难的语言模型，它使用词的分布式表示对自然语言序列建模。

12.5.1 推荐系统

第十三章： 线性因子模型

14 自动编码器

自编码器是一种无监督学习（Unsupervisered Learning）模型，可从数据中学习出更好的表达， 目前已经用于逐层贪婪的预训练（Greedy layer-wise pre-train）。 自动编码器是神经网络的一种，经过训练后能尝试将输入复制到输出。 自动编码器内部有一个隐藏层h，可以产生编码来表示输入。该网络可以看做两部分组成：一个编码器函数h=f(x)和一个生成重构的解码器r=g(h). 如果一个自动编码器学会简单地设置g(f(x))=x,那么这个自动编码器不会很有用。相反，自动编码器应该被设计成不能学会完美地复制。

14.1 欠完备自动编码器

从自动编码器获得有用特征的一种方法是限制h的维度比x小，这种编码维度小于输入维度的自动编码器称为欠完备自动编码器。学习欠完备的表示将自动编码器捕捉训练数据中最显著的特征，

学习过程可以简单地描述为最小化一个损失函数

L(x,g(f(x)) （14.1）

其中L是一个损失函数，衡量g(f(x))与x的不相似性，如均方误差

当解码器g(h)是线性的且L是均方误差，欠完备的自动编码器会学习出于PCA相同生成子空间。因此拥有非线性编码函数f和非线性解码函数g的自动编码器能够学习出更强大的PCA非线性推广。

14.2 正则自动编码器

编码维数小于输入维数的欠完备自动编码器可以学习数据分布最显著的特征。

包括稀疏表示，表示的小导数，以及对噪声或输入缺失的鲁棒性。

14.2.1 稀疏自动编码器

稀疏自动编码器简单地在训练时结合编码层的稀疏惩罚和重构误差：

L(x,g(f(x))+ (14.2)

稀疏自动编码器通常用于学习特征，以便用于其他任务如分类。 我们可以认为整个稀疏自动编码器框架是对带有隐变量的生成模型的近似最大似然训练，而不讲稀疏惩罚视为复制任务的正则化。

14.2.2 去噪自动编码器

去噪自动编码器最小化

L(x,g(f()) (14.9)

其中是被某种噪声损坏的x的副本。因此去噪自动编码器必须撤消这些损坏。

14.2.3 惩罚导数作为正则

另一正则化自动编码器的策略是使用一个类似稀疏自动编码器中的惩罚项

L(x,g(f(x))+ (14.10)

但的形式不同：

(14.11)

这迫使模型学习一个在x变化小时目标也没有太大变化的函数。这样的正则化的自动编码器被称为收缩自动编码器

14.3 表示能力、层的大小和深度

自动编码器也是前馈网络，因此加深网络也有很多优势。

训练深度自动编码器的普遍的策略是训练一堆浅层的自动编码器来贪心地预训练相应的深度架构。

14.4 随机编码器和解码器

14.5 去噪自动编码器

去噪自动编码器是一类接受损坏数据作为输入，并训练来预测原始未被损坏数据作为输出的自动编码器。；

14.5.1分数估计

分数匹配是最大似然的代替。

过程：

1.从训练数据中采一个训练样本x

2.从C()采一个损坏样本

3.将(x,)作为训练样本来估计自动编码器的重构分布.

其中h是编码器f()的输出，根据解码函数g(h)定义。

通常我们可以简单地对负对数释然-log.进行基于梯度法的近似最小化。

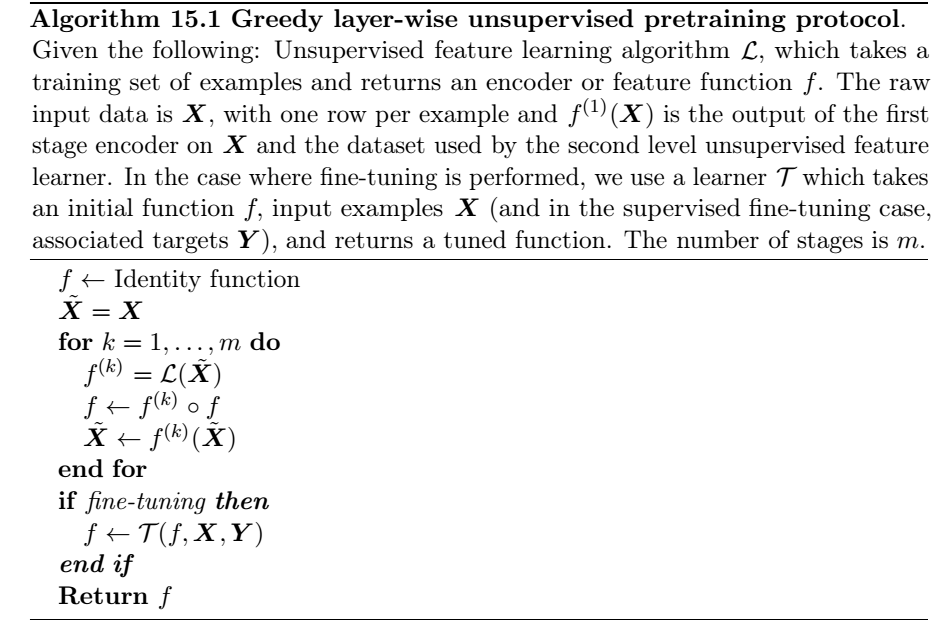
第十五章 表示学习

表示的选择通常取决于随后的学习任务。

表示学习特别有趣，因为它提供了一种方法来进行无监督学习和半监督学习。使用监督学习技巧于有标记的训练子集上通常会导致过拟合。半监督学习通过进一步学习未标记的数据，来解决过拟合的问题。具体地，我们可以从未标记的数据上学习出很好的表示，然后用这些表示来解决监督学习问题。

在本章中，我们专注于未标记的数据可以用来学习更好表示的假设。

15.1 贪心地逐层无监督预训练



贪心逐层无监督预训练被称为贪心的，是因为它是一个贪心算法那，这意味着它每次独立地优化一块解决方案，而不是联合优化所有的块

15.1.1 何时以及为何无监督训练有效？

无监督学习不同于权重衰减这样的正则化项，它不偏向于学习一个简单的函数，而是学习对无监督学习任务有用的特征函数。经过无监督预训练的神经网络会一致地停止在一片相同的区域，但未经过预训练的神经网络会一致地停在另一个区域。

15.2 迁移学习和领域自适应

迁移学习和领域自适应指的是利用一个设定(分布P1)中已经学到的内容去改善另一个设定(比如分布P2)中的泛化情况。