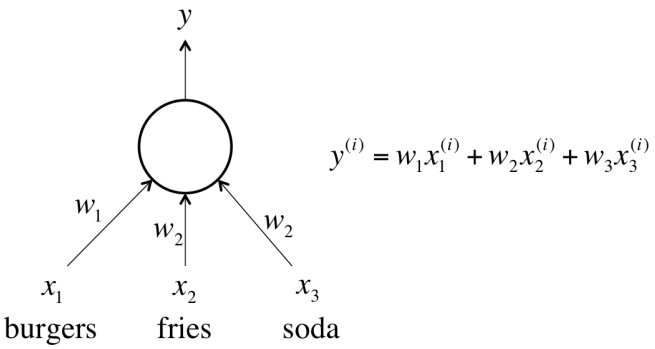
Chapter 2 训练前馈神经网络



采用线性神经元，即激活函数是线性函数，误差采用平方误差，

$E = \frac{1}{2}\Sigma_i \left(t^{(i)} - y^{(i)}\right)^2$ ，求令E最小的参数向量。（E越接近0 越好）

此时我们可能会想，当我们可以把这个问题当作一个方程组来解时，为什么要用error functions来烦扰自己呢?

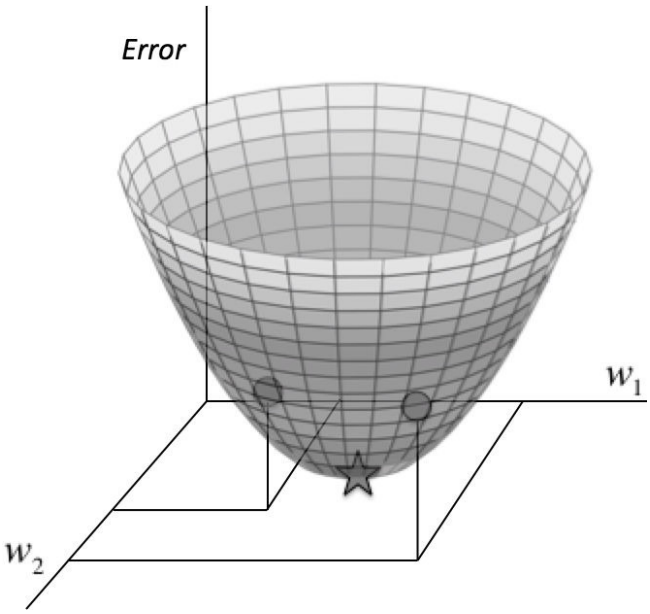
毕竟，我们有一堆未知数（权重），我们有一组方程式（每个训练例子中的一个）。如果我们有一组一致的训练示例，这将自动给我们带来0的error。

不幸的是，这样的结果泛化很差。尽管我们这里使用的是linear neuron，但linear neuron在实践中并没有被广泛使用，因为它们学到的东西很有限。

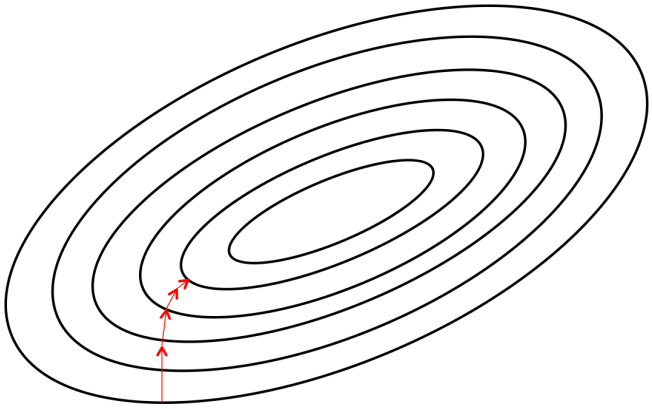
那时，我们开始使用 非线性神经元，如sigmoidal, tanh, ReLU neurons。我们不再能建立一个方程组（如使用linear neurons那样），显然我们需要一个更好的策略来处理训练过程。

Gradient Descent

假设有2个inputs, 想象一个三维空间：



俯视图：一组椭圆等高线，最小误差在椭圆的中心。



二维平面，维数对应2 weights。同一等高线上，每个位置的E都相等，

椭圆相互之间越靠近，坡度越陡峭。事实上，可证明，最陡（快）下降方向总是垂直于等高线。这个方向表示为一个向量，称为Gradient “梯度”

现在我们可以开发一个高层次的策略来找到 如何使错误函数最小化的权重值：

1、假设我们随机初始化weights，我们在水平面上的某个位置；

2、评估所在位置的梯度（方向向量），我们可以找到the direction of steepest descent 最快下降的方向，并朝这个方向走一步；

3、我们会处在一个新位置，这个位置比之前的跟接近中心（ 最小误差）；

4、重复步骤2

以上算法（策略）即是Gradient Descent

Test Sets, Validation Sets, and Overfitting

人工神经网络的主要问题之一是，模型太复杂了。这导致的问题是，太容易发生overfitting。

模型很好地fit 训练集，但是对新进来的数据，表现很差。也就是 泛化generalize 能力很差，不能举一反三，这种phenomenon 称作overfitting。

深度学习中，这个问题尤其突出。因为（深度）神经网络中有大量layers，layers里又含有很多neurons。

也就是说，一个模型中的 the number of connections 极其庞大，可达百万级。连接错综复杂，overfitting 实在是家常便饭。

如图是神经网络的上述表现：

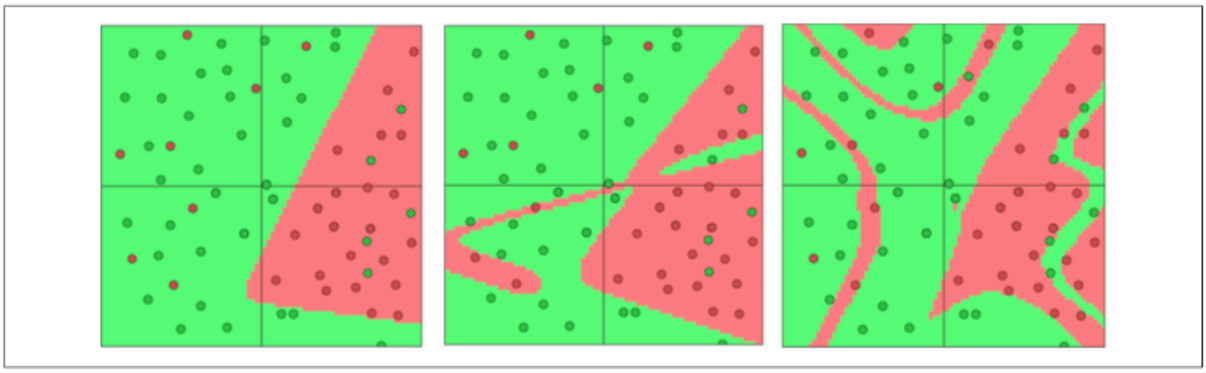


Figure 2-11. A visualization of neural networks with one, two, and four hidden layers (in that order) of three neurons each

三个观察启示：

1、需要在模型复杂度和overfitting 之间取得平衡，之后介绍神经网络里几种应对overfitting的对策：

2、需要拆分出Training data 、Testing data，以evaluate 模型对new data 的泛化能力

3、在training set 上训练时，也可能overfitting，自某一刻开始不再学习有用的features。

为了避免这种情况，训练过程中一有过拟合的苗头出现，我们希望能够及时停止训练。为此，

我们将整个训练过程分成几个epoch——一个epoch 是整个数据集上的一次迭代。

换句话说，假设 training set of size——d，采用小批量梯度下降，

batch size ——b ， 那么，

An epoch = d/b model updates.

Each epoch，每次迭代临终时，我们都想度量模型的泛化能力如何。为此，我们使用另外的validation set。

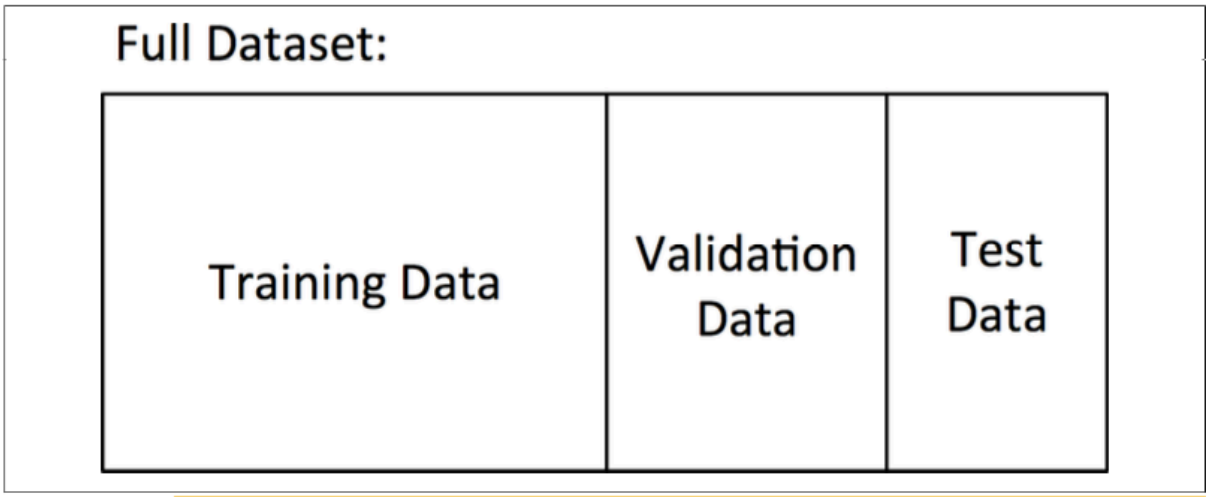


Figure 2-13. In deep learning we often include a validation set to prevent overfitting during the training process

一个epoch结尾时，validation set 将告诉我们模型在training set上的表现:

如果在training set 上的准确度继续增加，而validation set 上的准确的保持不变或降低，

那么，这很好地暗示“是时候stop training了”，开始overfitting了。

在超参数优化的过程中，基于validation set 进行accuracy 度量也很有用。如何寻找最优参数值？一个方法是用 grid search：

比如：

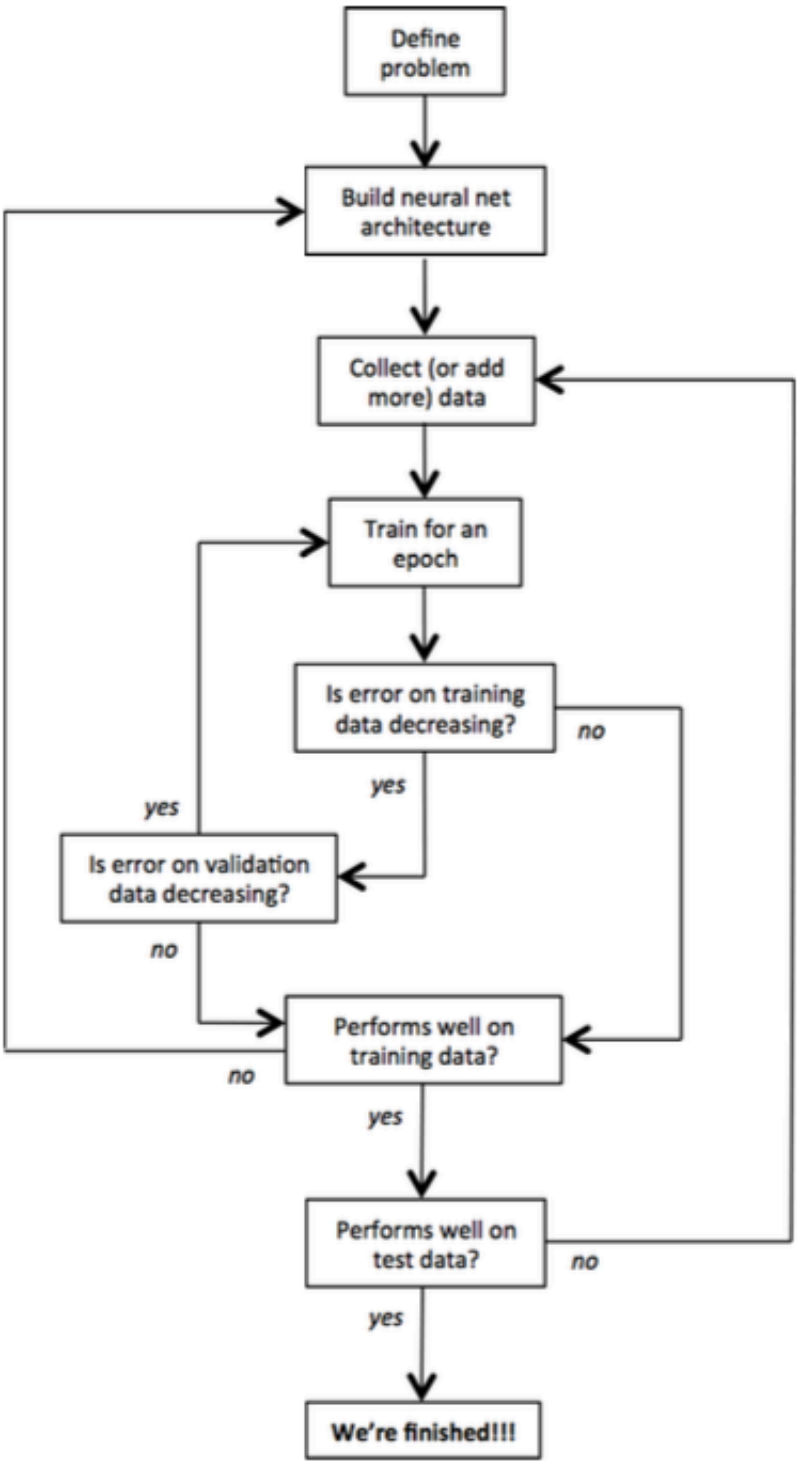
$\varepsilon \in \{ 0.001, 0.01, 0.1 \}$;

batch size $\in \{16, 64, 128\}$

从所有取值组合中选择在validation set 上表现最好的参数组合，即为最优参数，然后report它们在 test set 上的accuracy。

用validation set 上的最优参数，在test set 上 report 模型性能的accuracy

Detailed work ow for training and evaluating a deep learning model



1、严格定义问题。This involves determining：inputs、outputs、the vectorized representations of both。

举例来说：假设目标是训练一个判别癌症的深度学习模型，

Input: RBG图像，由一组 像素值向量 表示；

Output: 3个互斥可能性的概率分布：1）正常，2）良性肿瘤，3）恶性肿瘤。

2、定义了问题之后，我们需要建立一个神经网络结构来解决问题：

Input layer 必须具有合适的size 才能接受来自图像的原始数据；

output layer： a softmax of size 3；

以及内部的隐层数、connectivities（比如CNN），等等。

3、然后，收集大量数据用于training or modeling.

这些数据可能是由医学专家标注的统一大小的病理图像。

我们将这些数据打乱，分成training、validation、testing set

4、准备开始梯度下降。在training set 上训练模型，一次一个epoch。

每个epoch结尾时（每轮迭代临终时），确保training 和validation set 上的误差都在降低。

当其中一个误差不再降低，

如果在 training set 上的误差不再降低，我们可能要改进特征工程，从数据中抓取重要features；

如果validation set 误差不再降低，我们可能要采取措施防止overfitting。

其中一个不再降低，我们就终止迭代，然后评估模型在train data 上的表现：

如果不尽人意，我们需要反思步骤2的模型结构；

如果满意，继续评估在test data 上的性能，

如果不好，需要重新考虑收集的数据， 是否足够多，是否能提供 预测所需的、代表性强的信息，

比如我们希望模型能够识别出“苹果”，那么收集投喂的数据不能是“榴莲”。

Otherwise，we are finished !

Preventing Over tting in Deep Neural Networks

Regularization：通过加入一个额外项来惩罚过大的权重，以修正目标函数。

目标函数可以理解为，误差（值或函数）+ 假设函数f：

objective function = target function = Error + λ f (θ)

==> 那么 Error = target f - λ f (θ)

f (θ) 随 θ 增大而增大；

λ 是正则化强度，是一个超参数，决定防止过拟合的防范程度。正则强度λ 越大，Error就越小从而达到防止过拟合（E_out 增大）的目的；

这里以 λ f (θ) 乘积表示，但不是真的乘入正则项，而是 【+ λ正则项】。当λ=0时，则无正

则项，不通过正则化措施来防止过拟合。
如果λ太大，那么在训练集上寻找最优参数时，模型优先使θ 保持尽可能小。
λ 的选择很重要，需要不断试错、调试。

常见的正则化类型

L2 regularization：
对于神经网络中的每个权值w，原误差函数变为：error function+1/2 λw^2 。
L2 正则化能直观诠释 对peaky weight vectors 的重惩罚 和 对 diffuse weight vectors 的 prefer。
注意：在梯度下降更新中，使用L2正则化最终意味着每一个权值都线性衰减到零。
因此，L2 regularization 也常被称为 权值衰减 weight decay。

L1 regularization：add the term λ|w| for every weight w in the neural network.
L1 regularization的性质：优化过程中，使weight vectors 变得稀疏，即非常接近0。
换句话说，
L1正规化的神经元最终只用到其inputs的一小部分，并对inputs中的噪声具有很强的抵抗力。

相比而言，L2正则化的weight vectors通常是扩散的，小数量的。
当你想准确地理解哪些特征有助于决策时，L1正则化是非常有用的。
如果这一层次的特征分析是不必要的，我们更喜欢使用L2正则化，因为经验中它表现更好。

随机失活Dropout：

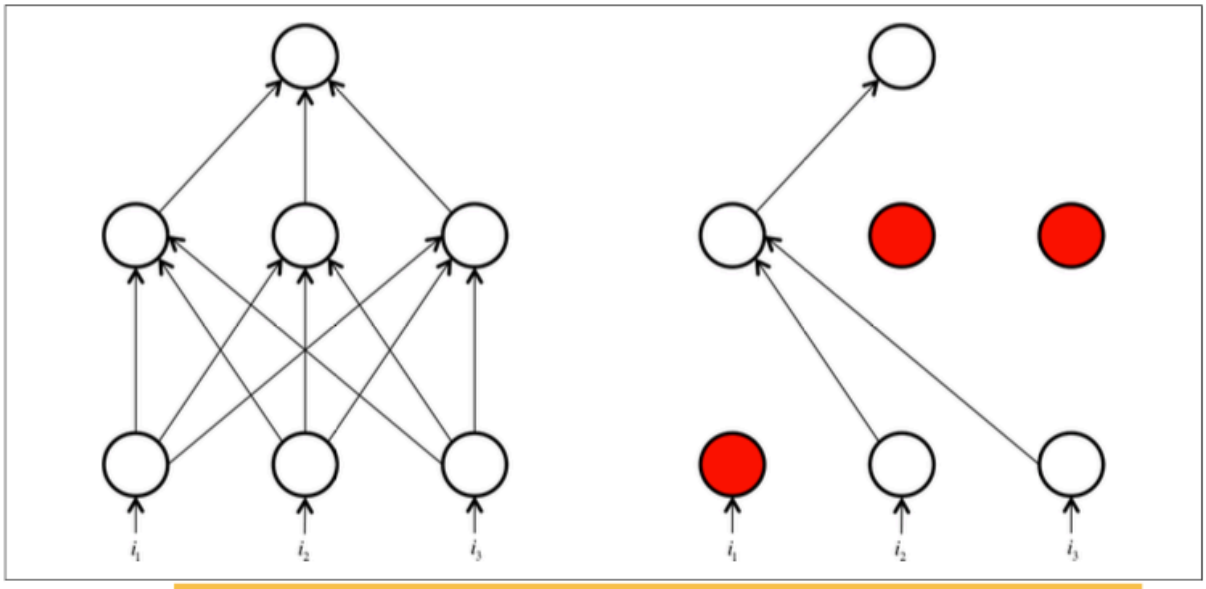


Figure 2-16. Dropout sets each neuron in the network as inactive with some random probability during each minibatch of training

在每轮训练的miniabtch 中，以随机概率将某一neuron 设置为inactive。

这是深度学习中最受青睐的防过拟合方法之一。
训练时，以p的概率让一个神经元保持活跃，或置其为0，p是超参数。
直观地说，即使在缺乏某些信息的情况下，这也会迫使网络模型保持准确。
它可以防止整个网络过于依赖 任意某个neuron或neurons组合。

Dropout的一些重要细节：
我们希望test time 时神经元的outputs = 其train time时的期望输出。
可以通过 缩放 test 时的outputs 达到。例如：若p=0.5，test 时，neurons 的outputs 必须缩减1/2。这是显而易见的，因为一个neuron 的output 有 1- p=0.5 的概率被设置为0，
也就是说，一个neuron 的output 在dropout之前 是x，那么dropout 之后，
期望输出为 E [output] = px + (1 – p) · 0 = px

这种简易的实现并不可取，因为它要求在test time 对neuron outputs进行缩放。模型在test time 的表现至关重要，因此使用 inverted dropout 更可取，即将缩放行为 放在train time 而不是 test time。

inverted dropout 中，任意有活性的neuron 的output 值传播到next layer 前，先除以 p：

$$E[\text{output}] = p \cdot \frac{x}{p} + (1 - p) \cdot 0 = x,$$

我们就能避免test time 时神经元输出的任意缩放。