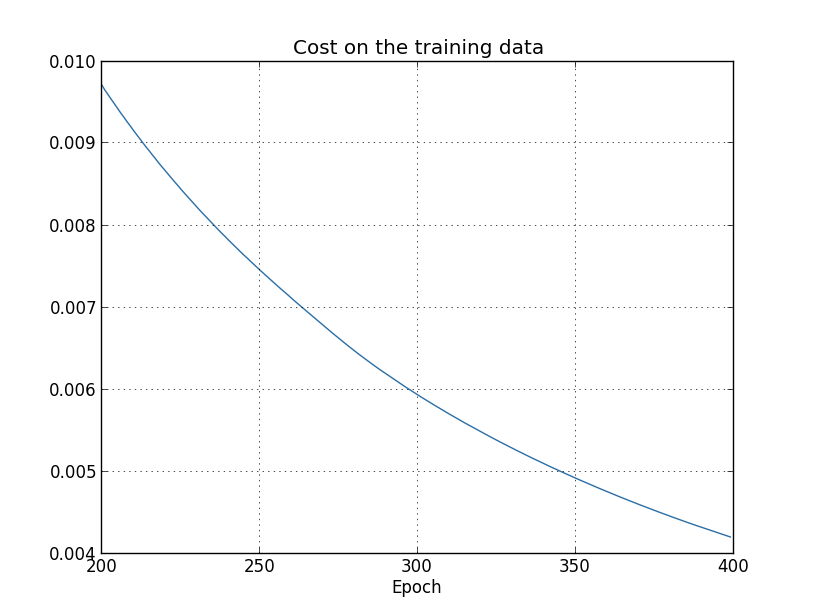
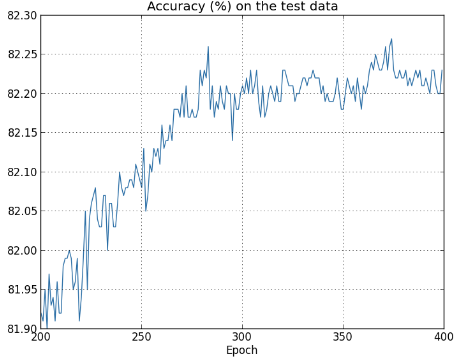
**机器学习/深度学习中**

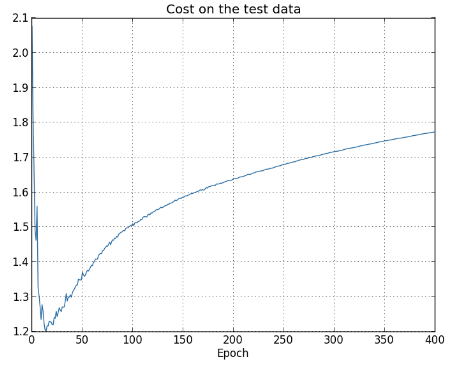
**的over-fitting**

**情形1**

这张图看起来很好，正如我们所期望的那样，训练集的cost函数展示出平滑下降的趋势。但是在看下面的测试集所表现出的精度图。

在前200步长时，精度就已经达到了82%。但是在280步长后，精度好像不在有所提高了，只是在随机的波动。相反的是，在训练集表现出的cost函数值还在下降，我们称其为超过280步长的over fitting。

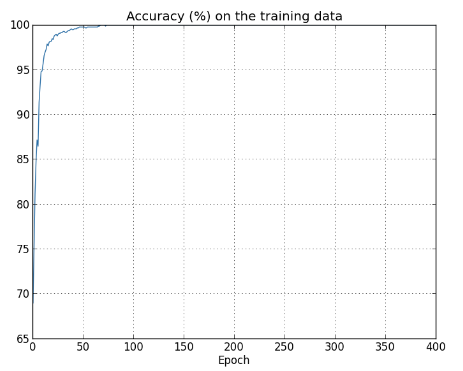
**情形二**

你可能好奇如果我们用测试集做一下cross-entropy cost检验会有什么结果相比于用训练集数据。同样的情形仍然会发生。

我们可以看到测试集的cost有所提高知道不成达到15左右，但在15之后，情况开始变得糟糕，尽管在训练集上cost表现的一直很好。此外，我们真正关心的测试集上的分类精度的提高，虽然测试集的cost不过是分类精度的一个表征。所以，将步长280作为在模型中over fitting将会控制learning的阈值是最具意义的。

**情形三**

训练集数据表现出的精度一直打到100%。意味着，模型能够正确地识别每一张图片。同时，测试集的精度最大才仅仅是82.27%。那几乎像是我们的模型在记忆训练集，并没有对数字有充分的理解将其推广到训练集。

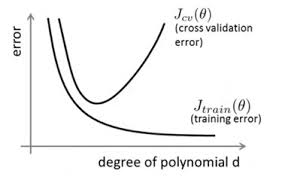


**机器学习/深度学习中**

**的正则化方法**

目的：防止过拟合，提高泛化能力

在训练数据不够多时，或者overtraining时，常常会导致overfitting（过拟合）。其直观的表现如下图所示，随着训练过程的进行，模型复杂度增加，在training data上的error渐渐减小，但是在验证集上的error却反而渐渐增大——因为训练出来的网络过拟合了训练集，对训练集外的数据却不work。

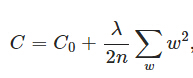


为了防止overfitting，可以用的方法有很多，下文就将以此展开。有一个概念需要先说明，在机器学习算法中，我们常常将原始数据集分为三部分：training data、validation data，testing data。这个validation data是什么？它其实就是用来避免过拟合的，在训练过程中，我们通常用它来确定一些超参数（比如根据validation data上的accuracy来确定early stopping的epoch大小、根据validation data确定learning rate等等）。那为啥不直接在testing data上做这些呢？因为如果在testing data做这些，那么随着训练的进行，我们的网络实际上就是在一点一点地overfitting我们的testing data，导致最后得到的testing accuracy没有任何参考意义。因此，training data的作用是计算梯度更新权重，validation data如上所述，testing data则给出一个accuracy以判断网络的好坏。

避免过拟合的方法有很多：early stopping、数据集扩增（Data augmentation）、正则化（Regularization）包括L1、L2（L2 regularization也叫weight decay），dropout。

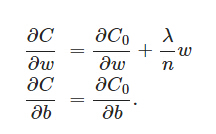
# ****L2 regularization（权重衰减）****

L2正则化就是在代价函数后面再加上一个正则化项：

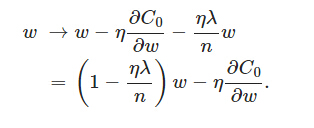


C0代表原始的代价函数，后面那一项就是L2正则化项，它是这样来的：所有参数w的平方的和，除以训练集的样本大小n。λ就是正则项系数，权衡正则项与C0项的比重。另外还有一个系数1/2，1/2经常会看到，主要是为了后面求导的结果方便，后面那一项求导会产生一个2，与1/2相乘刚好凑整。

L2正则化项是怎么避免over fitting的呢？我们推导一下看看，先求导：

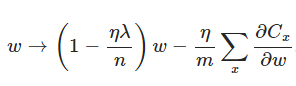


可以发现L2正则化项对b的更新没有影响，但是对于w的更新有影响:



在不使用L2正则化时，求导结果中w前系数为1，现在w前面系数为 1−ηλ/n ，因为η、λ、n都是正的，所以 1−ηλ/n小于1，它的效果是减小w，这也就是权重衰减（weight decay）的由来。当然考虑到后面的导数项，w最终的值可能增大也可能减小。

另外，需要提一下，对于基于mini-batch的随机梯度下降，w和b更新的公式跟上面给出的有点不同：

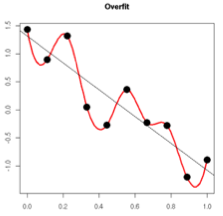


http://i.imgur.com/yDETU7x.jpg

对比上面w的更新公式，可以发现后面那一项变了，变成所有导数加和，乘以η再除以m，m是一个mini-batch中样本的个数。

到目前为止，我们只是解释了L2正则化项有让w“变小”的效果，但是还没解释为什么w“变小”可以防止over fitting？一个所谓“显而易见”的解释就是：更小的权值w，从某种意义上说，表示网络的复杂度更低，对数据的拟合刚刚好（这个法则也叫做奥卡姆剃刀），而在实际应用中，也验证了这一点，L2正则化的效果往往好于未经正则化的效果。当然，对于很多人（包括我）来说，这个解释似乎不那么显而易见，所以这里添加一个稍微数学一点的解释（引自知乎）：

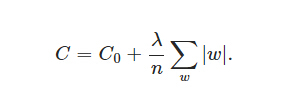
过拟合的时候，拟合函数的系数往往非常大，为什么？如下图所示，过拟合，就是拟合函数需要顾忌每一个点，最终形成的拟合函数波动很大。在某些很小的区间里，函数值的变化很剧烈。这就意味着函数在某些小区间里的导数值（绝对值）非常大，由于自变量值可大可小，所以只有系数足够大，才能保证导数值很大。



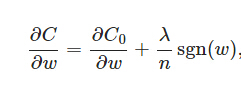
而正则化是通过约束参数的范数使其不要太大，所以可以在一定程度上减少过拟合情况。

# ****L1 regularization****

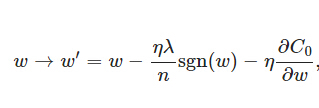
在原始的代价函数后面加上一个L1正则化项，即所有权重w的绝对值的和，乘以λ/n（这里不像L2正则化项那样，需要再乘以1/2，具体原因上面已经说过。）



同样先计算导数：



上式中sgn(w)表示w的符号。那么权重w的更新规则为：

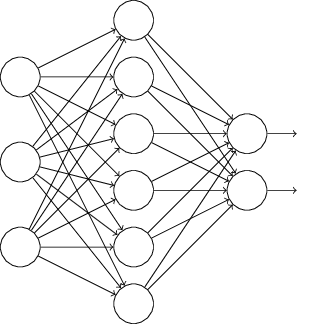


比原始的更新规则多出了η \* λ \* sgn(w)/n这一项。当w为正时，更新后的w变小。当w为负时，更新后的w变大——因此它的效果就是让w往0靠，使网络中的权重尽可能为0，也就相当于减小了网络复杂度，防止过拟合。

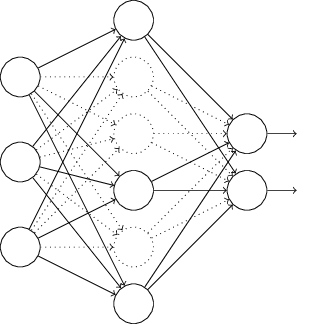
另外，上面没有提到一个问题，当w为0时怎么办？当w等于0时，|W|是不可导的，所以我们只能按照原始的未经正则化的方法去更新w，这就相当于去掉η\*λ\*sgn(w)/n这一项，所以我们可以规定sgn(0)=0，这样就把w=0的情况也统一进来了。（在编程的时候，令sgn(0)=0,sgn(w>0)=1,sgn(w<0)=-1）

# ****Dropout****

L1、L2正则化是通过修改代价函数来实现的，而Dropout则是通过修改神经网络本身来实现的，它是在训练网络时用的一种技巧（trike）。它的流程如下：



假设我们要训练上图这个网络，在训练开始时，我们随机地“删除”一半的隐层单元，视它们为不存在，得到如下的网络：



保持输入输出层不变，按照BP算法更新上图神经网络中的权值（虚线连接的单元不更新，因为它们被“临时删除”了）。

以上就是一次迭代的过程，在第二次迭代中，也用同样的方法，只不过这次删除的那一半隐层单元，跟上一次删除掉的肯定是不一样的，因为我们每一次迭代都是“随机”地去删掉一半。第三次、第四次……都是这样，直至训练结束。

以上就是Dropout，它为什么有助于防止过拟合呢？可以简单地这样解释，运用了dropout的训练过程，相当于训练了很多个只有半数隐层单元的神经网络（后面简称为“半数网络”），每一个这样的半数网络，都可以给出一个分类结果，这些结果有的是正确的，有的是错误的。随着训练的进行，大部分半数网络都可以给出正确的分类结果，那么少数的错误分类结果就不会对最终结果造成大的影响。

# ****数据集扩增（data augmentation）****

“有时候不是因为算法好赢了，而是因为拥有更多的数据才赢了。”

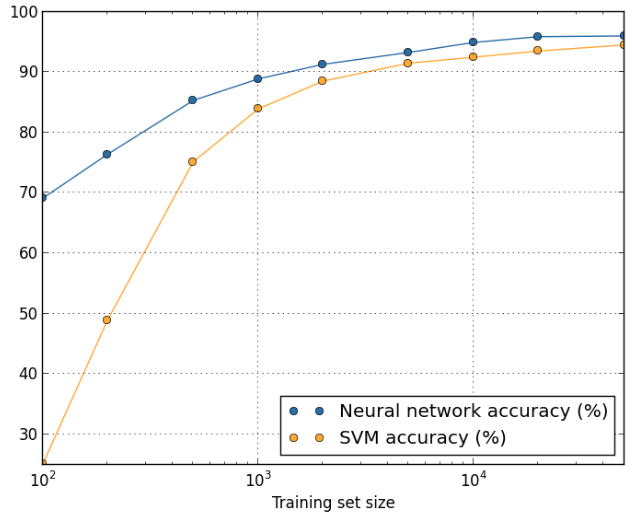
不记得原话是哪位大牛说的了，hinton？从中可见训练数据有多么重要，特别是在深度学习方法中，更多的训练数据，意味着可以用更深的网络，训练出更好的模型。

既然这样，收集更多的数据不就行啦？如果能够收集更多可以用的数据，当然好。但是很多时候，收集更多的数据意味着需要耗费更多的人力物力，有弄过人工标注的同学就知道，效率特别低，简直是粗活。

所以，可以在原始数据上做些改动，得到更多的数据，以图片数据集举例，可以做各种变换，如：

* 将原始图片旋转一个小角度
* 添加随机噪声
* 一些有弹性的畸变（elastic distortions），论文《Best practices for convolutional neural networks applied to visual document analysis》对MNIST做了各种变种扩增。
* 截取（crop）原始图片的一部分。比如DeepID中，从一副人脸图中，截取出了100个小patch作为训练数据，极大地增加了数据集。感兴趣的可以看《Deep learning face representation from predicting 10,000 classes》.

更多数据意味着什么？

用50000个MNIST的样本训练SVM得出的accuracy94.48%，用5000个MNIST的样本训练NN得出accuracy为93.24%，所以更多的数据可以使算法表现得更好。在机器学习中，算法本身并不能决出胜负，不能武断地说这些算法谁优谁劣，因为数据对算法性能的影响很大。