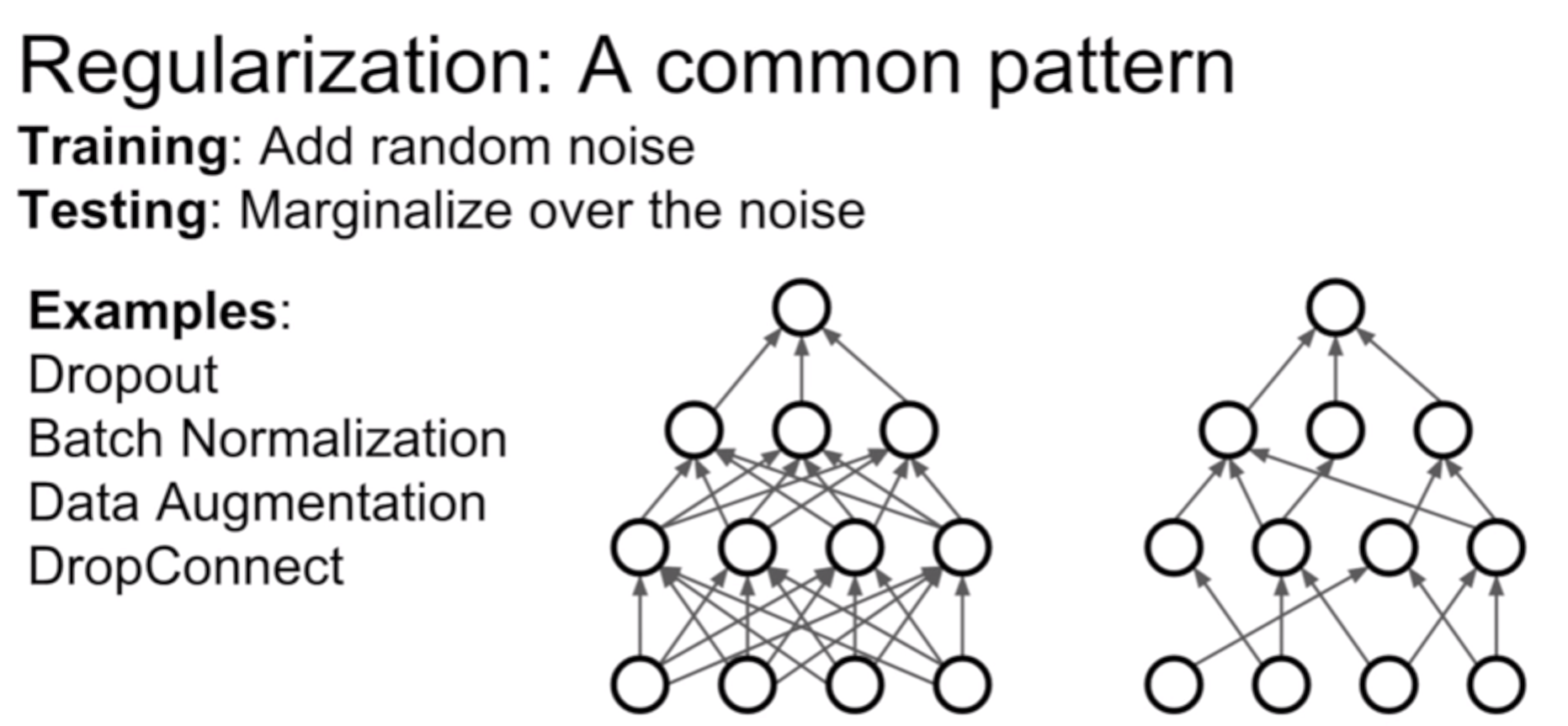
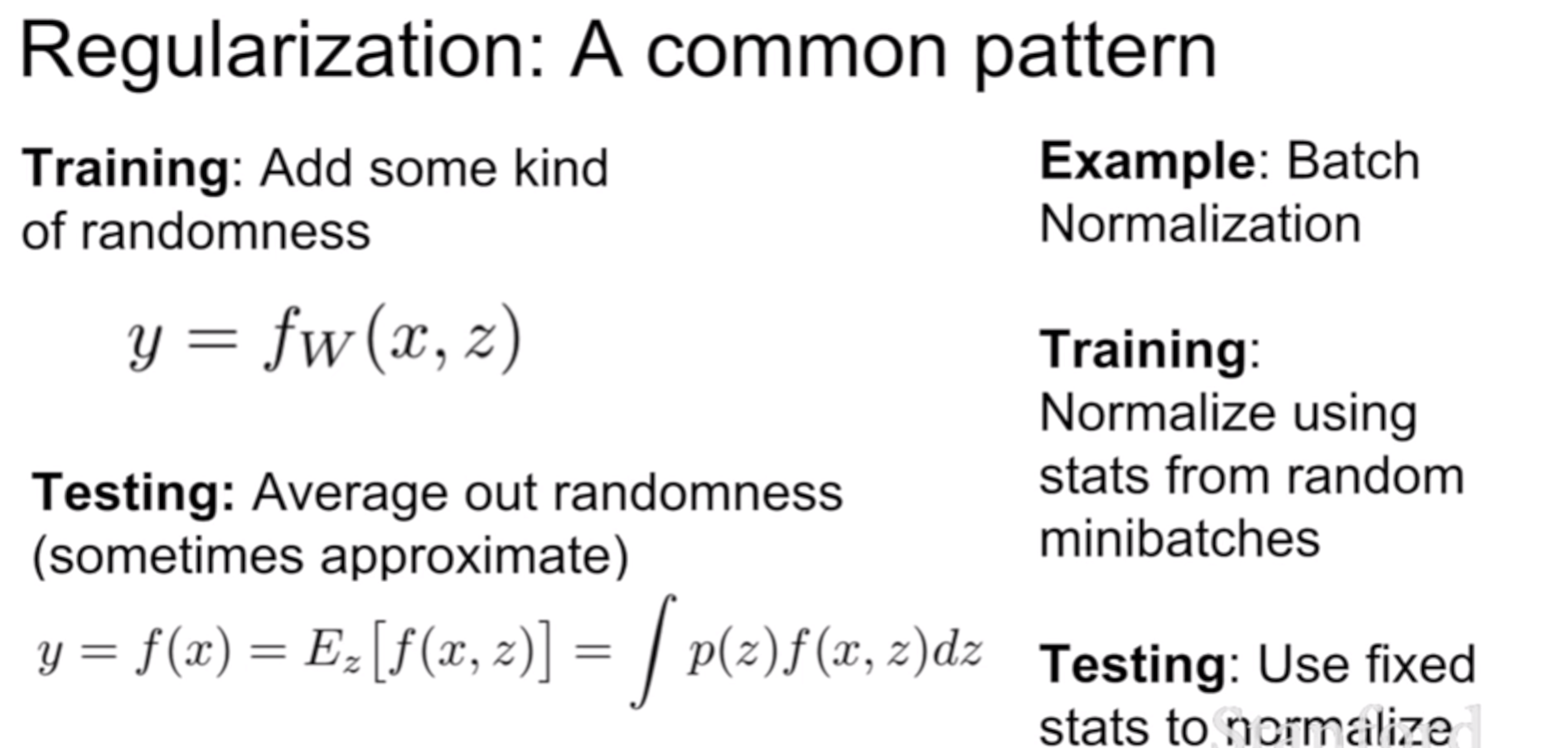
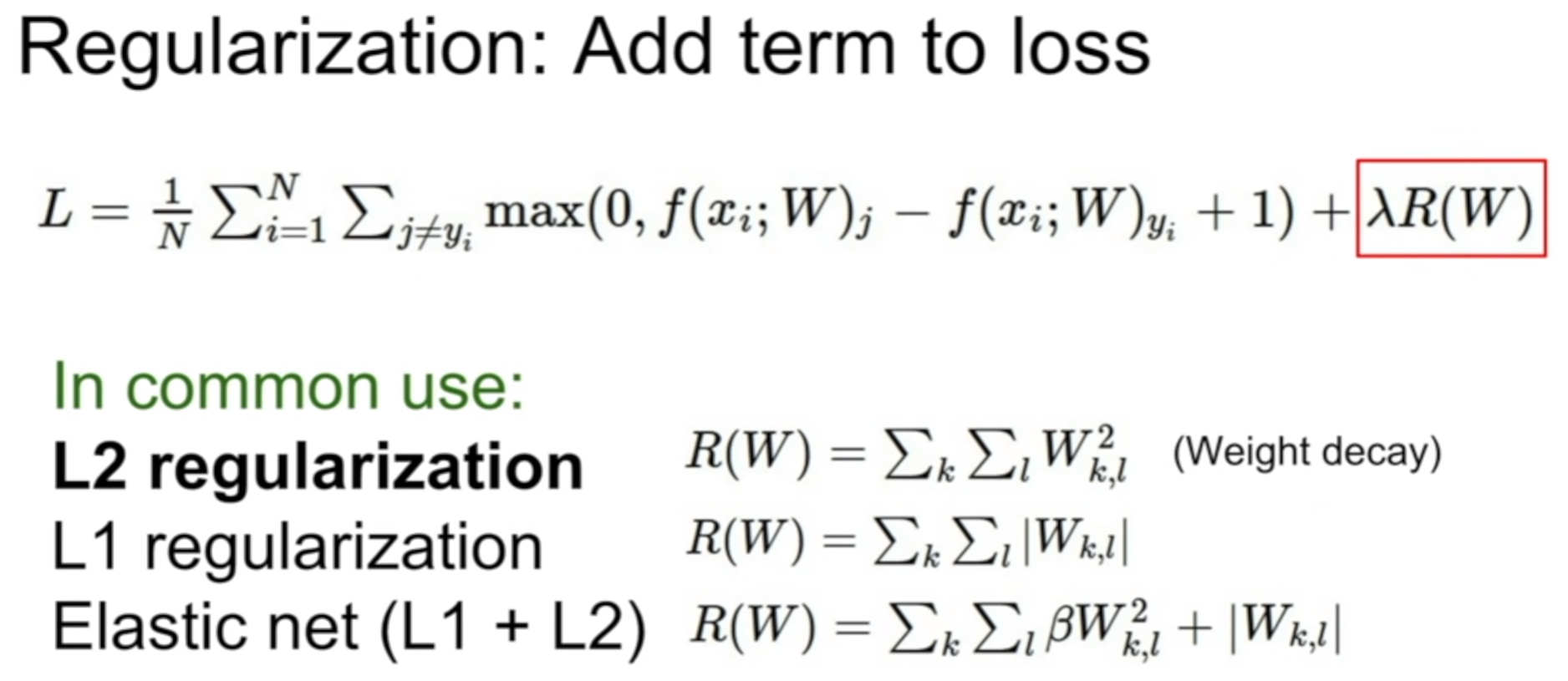
**正则化的通用模式：**

在训练时期，加入某些随机性到网络中来防止网络过拟合，相当于搞乱网络来防止网络对训练你和拟合过度。在测试时期，我们希望平均化这种随机性来改善泛化性能。所以为什么batch normalization也会有正则化效果，也是因为在训练时期归一化随机的minibatch，然后在测试时期采用固定的状态进行归一化，相当于平均化了结果。所以BN的效果跟dropout很相似，因为有些论文就说有了BN以后就不需要用dropout。虽然有了BN后可以不用dropout，但是dropout有个好处就是可以用概率p来控制正则化效果。

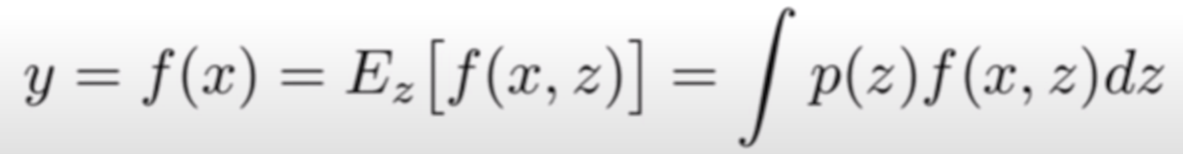


***L1/L2***

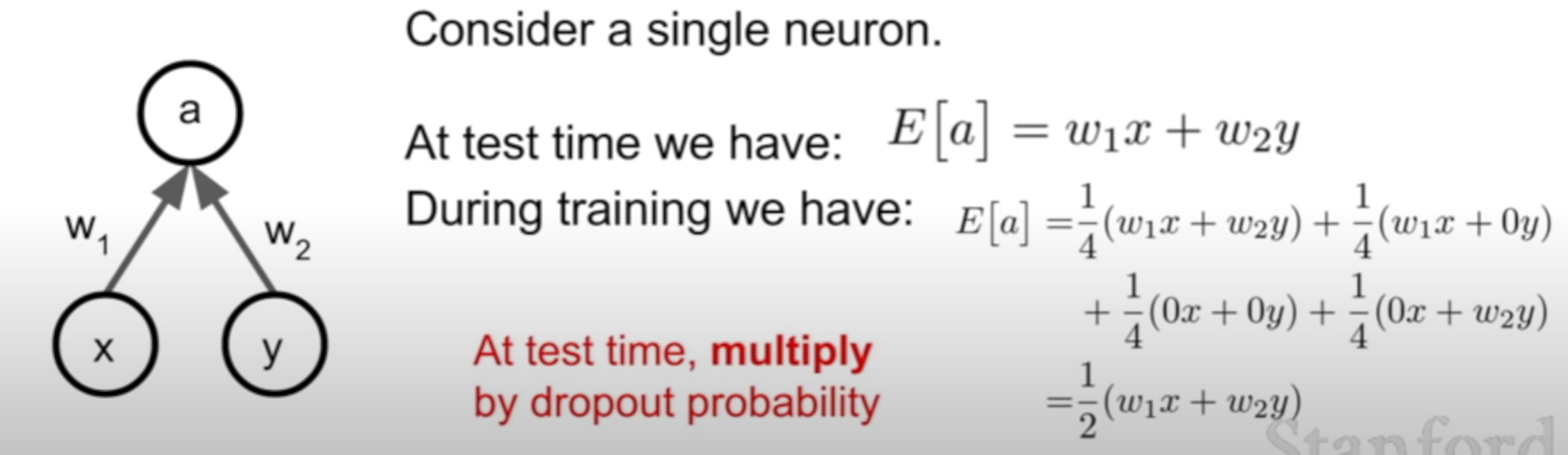
在损失函数加入正则项。但是L2在训练神经网络时候可能作用不大。



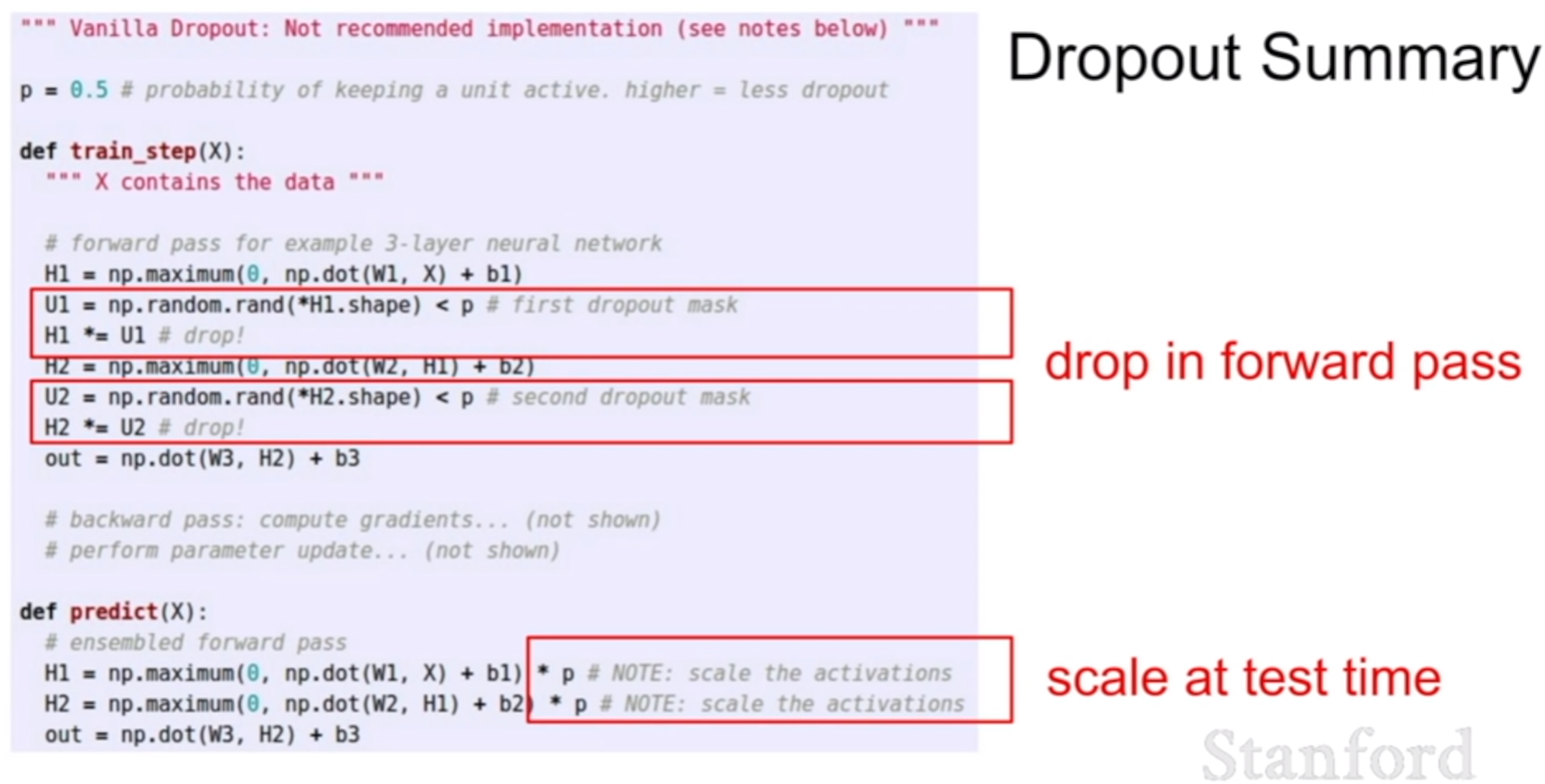
**Dropout**

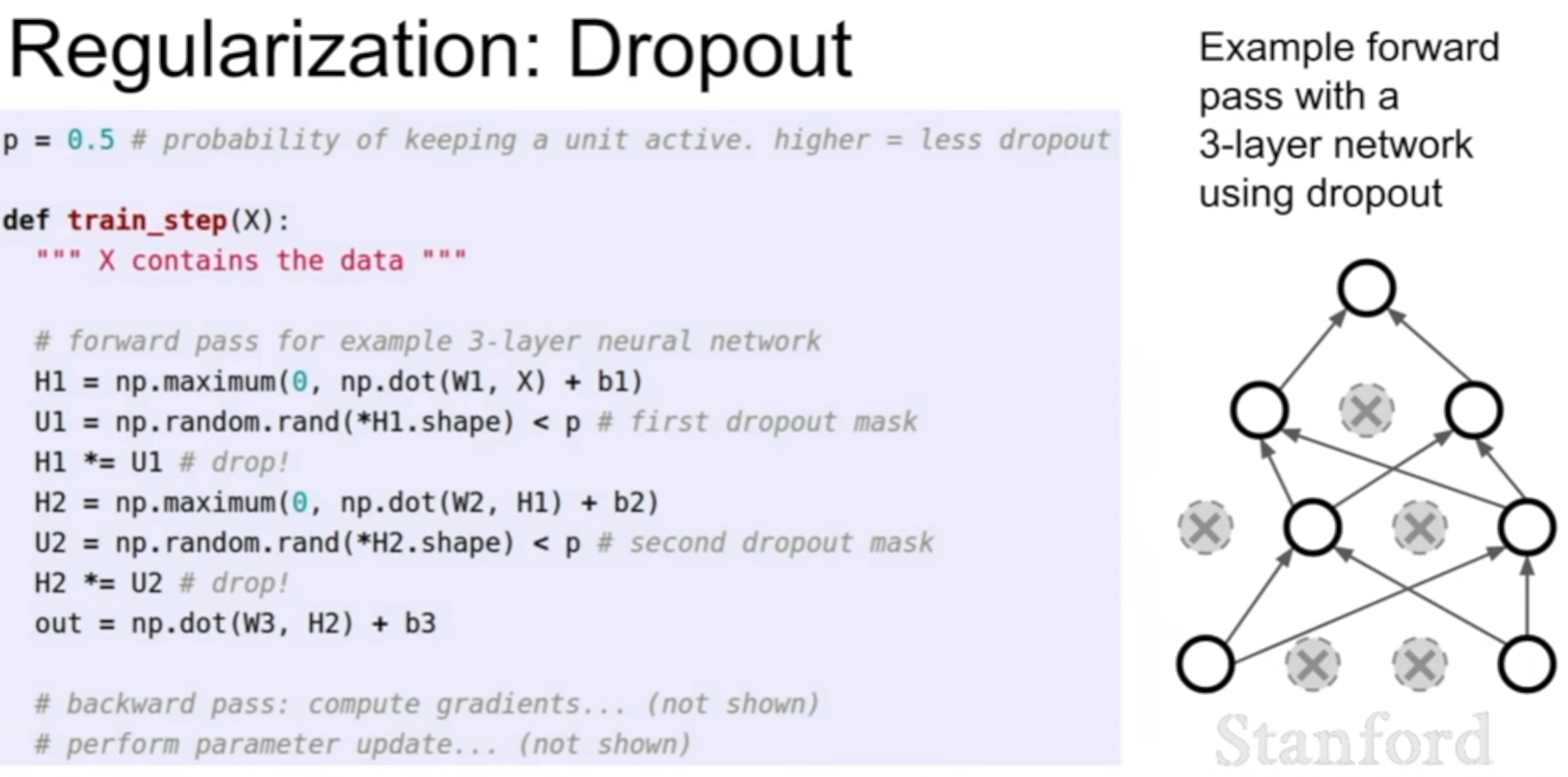
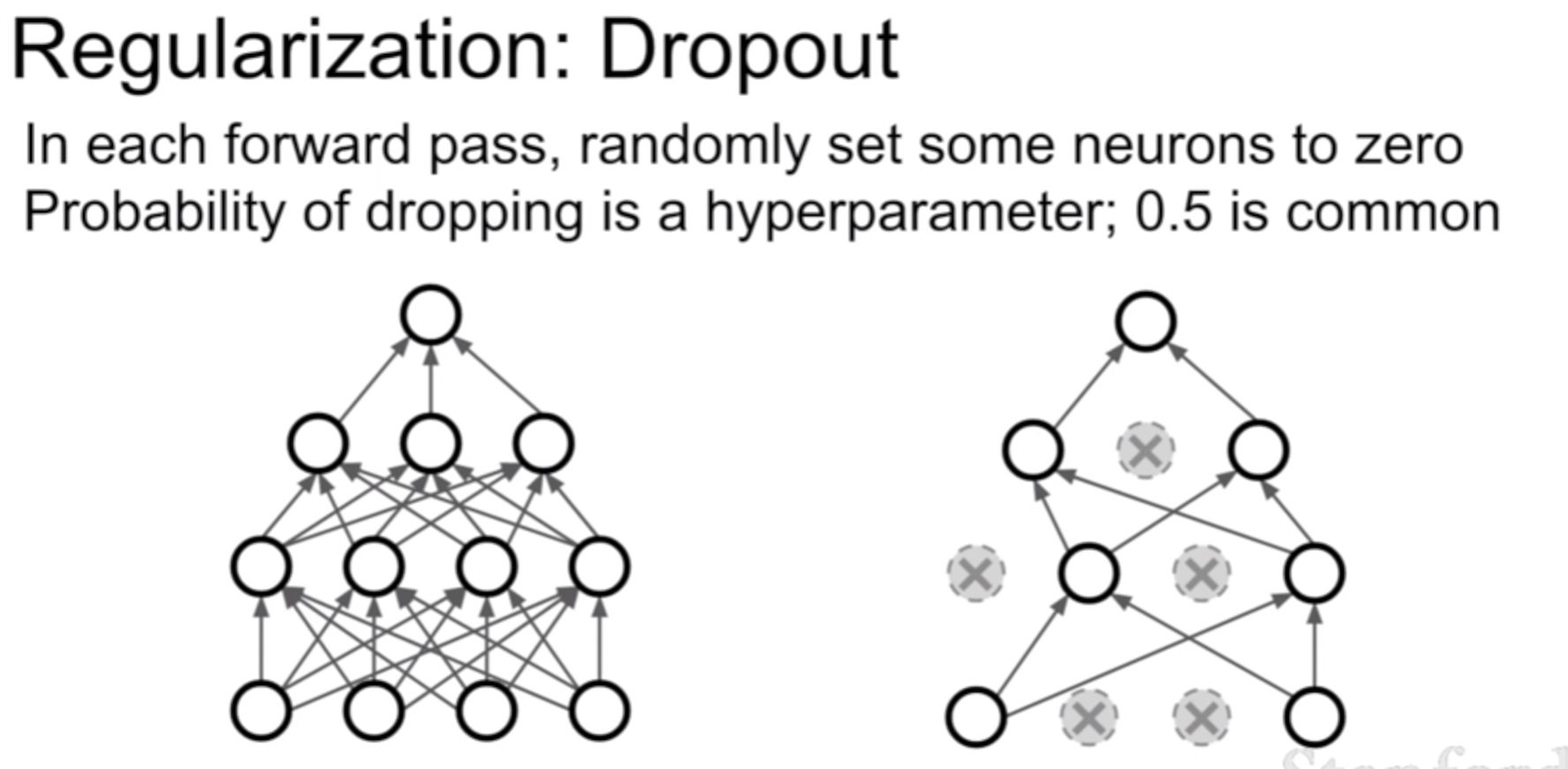
另一个正则化就是dropout，dropout能够work的一个原因是有时候输出并不需要所有的学习特征，只需要部分就能够确定下一轮的结果，所以能够更加robust。另一个原因是dropout的行为就像是聚类训练，得到多个不同的模型互相补充抵消，从而改善过拟合。但是dropout会给模型输出增加了随机性，这对测试时期是不好的，因为输出结果不稳定且随机。一个方法是在测试时期平均化这种随机性，如以下积分公式。但是这个公式无法计算，只能近似。实际上batch norm也是差不多这个意思。

所以在测试时期，可以用下面的方式：用近似来消除随机性，把dropout的概率乘起来。



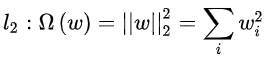
dropout是用来防止过拟合的过多参数才会容易过拟合，Ng课里说越多参数的全连接层要drop out的概率就大一点，卷积层用的少一点，但是也有论文用到的。最初的Dropout是用于输入层或者是全连接层，目的就是为了防止由于数据量或者模型过大导致的过拟合问题。标准的 Dropout一般是不能用于卷积层的，原因是因为在卷积层中图像中相邻的像素共享很多相同的信息，如果它们中的任何一个被删除，那么它们所包含的信息可能仍然会从仍然活动的相邻像素传递。所以卷积层中的dropout只是增加了对噪声输入的鲁棒性，而不是在全连接层中观察到的模型平均效果. 用dropout的话，训练时间更久，但是泛化性能更好



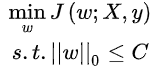
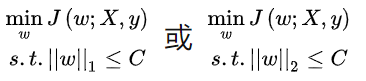


**L1/L2正则化：**

正则化的目的是控制模型复杂度，减小过拟合。最基本的正则化方法是在损失函数中添加惩罚项，对复杂度高的模型进行惩罚。，常用的Ω函数有两种，即L1范数和L2范数，相应称之为L1正则化和L2正则化。

**基于约束条件的最优化来理解正则化：**

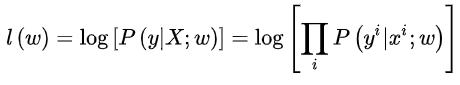
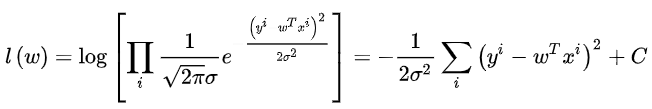
降低模型的复杂度从理论上来讲就是减少权重因子w的个数，因此在原优化问题中加入一个约束条件，即非零w的个数小于等于C。。但是这个问题是NP，不容易求解，所以对该约束条件稍微松弛了一下，即不严格要求某些w为0，而是要求w应接近于0，w尽量的小。因此才有使用L1和L2范数来接近L0范数。。利用拉格朗日算子法，可将这两个带约束条件的最优化问题转换为不带约束项的优化问题，构造拉格朗日函数：

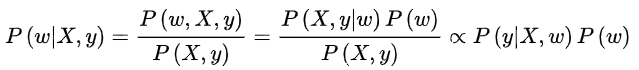
使用L2范数时，为方便后续处理，对||w||2进行平方，此时只需要调整C的取值即可。因此，L1/L2正则化等价于在原优化目标函数中增加约束条件。

**最大后验概率估计来理解正则化：**

假设标签服从高斯分布，相应的对数似然函数为：

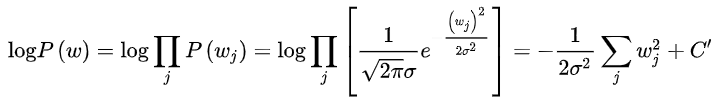
 

在最大后验概率估计中，将w看作随机变量，具有某种分布，因此根本naïve bayes可得：

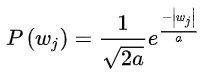
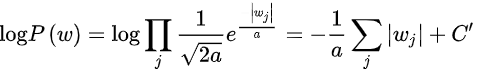
取对数：

可以看到在后面多了一项时logP(w)。

假设w的先验分布服从μ=0的高斯分布，则有

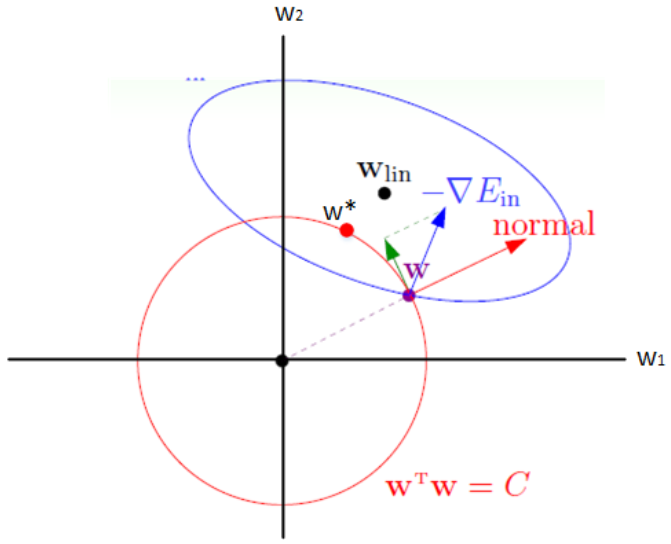
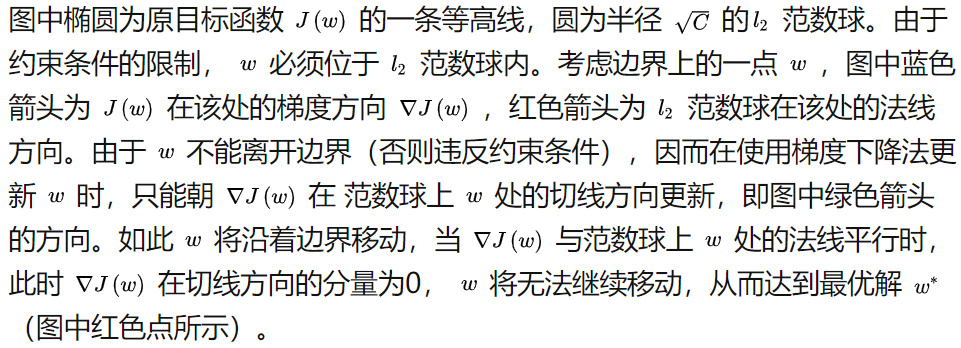
，此时相当于增加L2正则项。

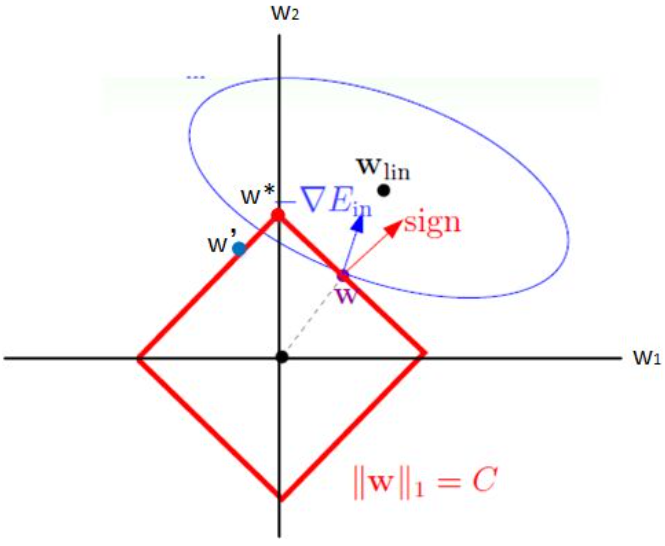
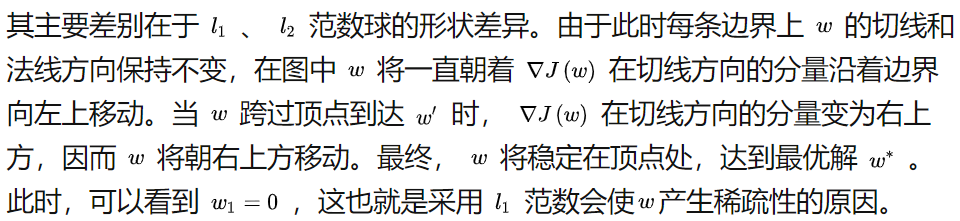
假设w服从μ=0，参数为a的拉普拉斯分布，则有

 ，此时相当于增加L1正则项。

因此，L1正则化可假设权重w的先验分布为拉普拉斯分布，有最大后验概率估计导出。同样，L2正则化可假设权重w的先验分布为高斯分布，由最大后验概率估计导出。

**L1/L2正则化效果分析：**

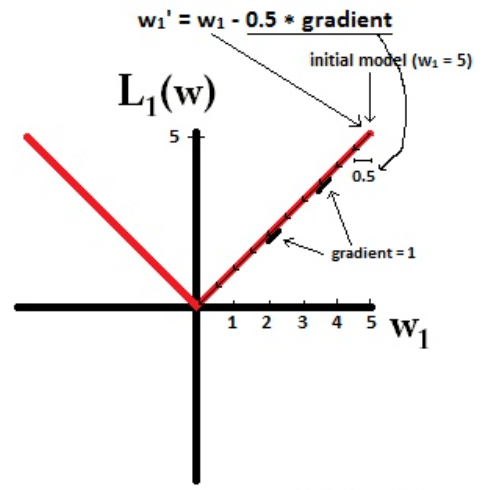
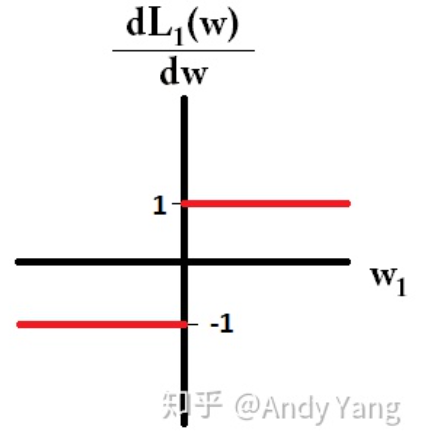
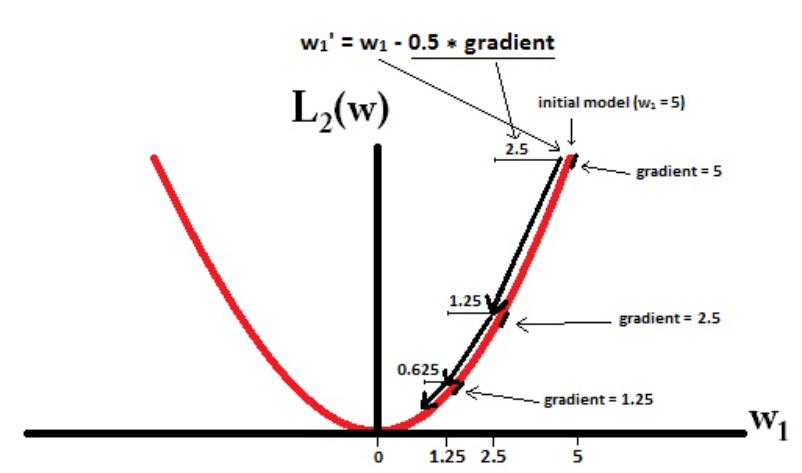
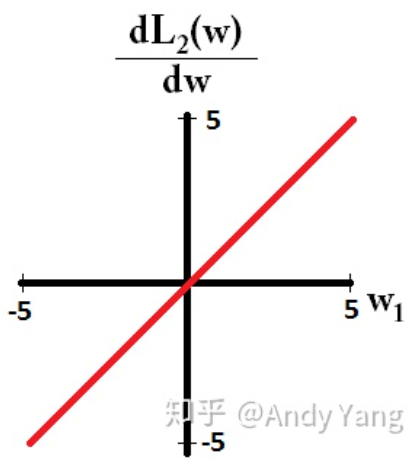
 

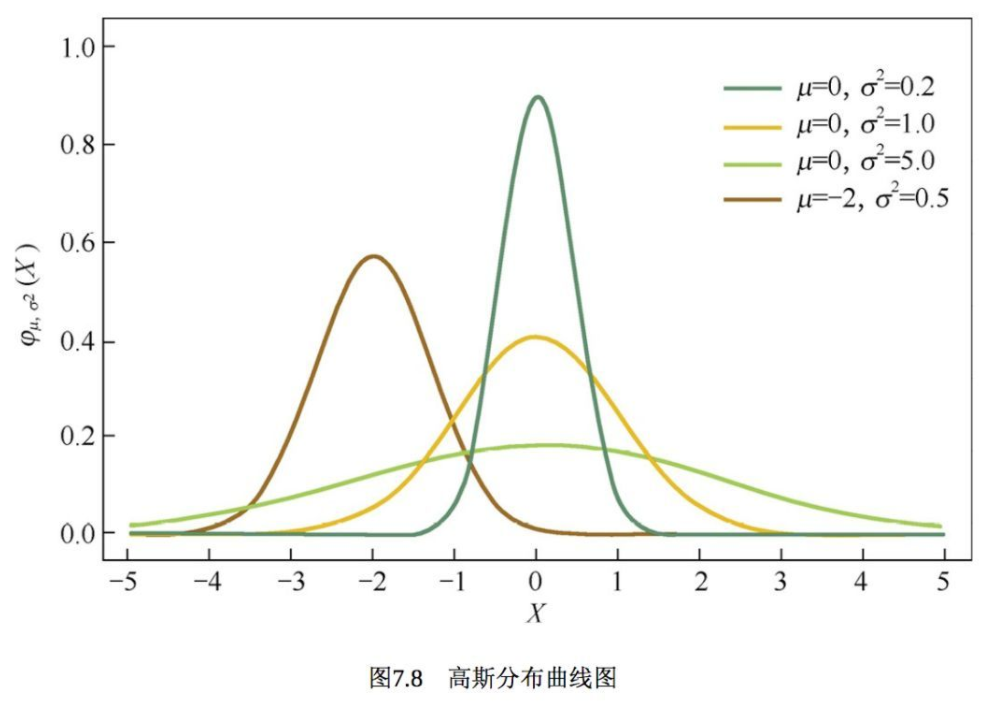
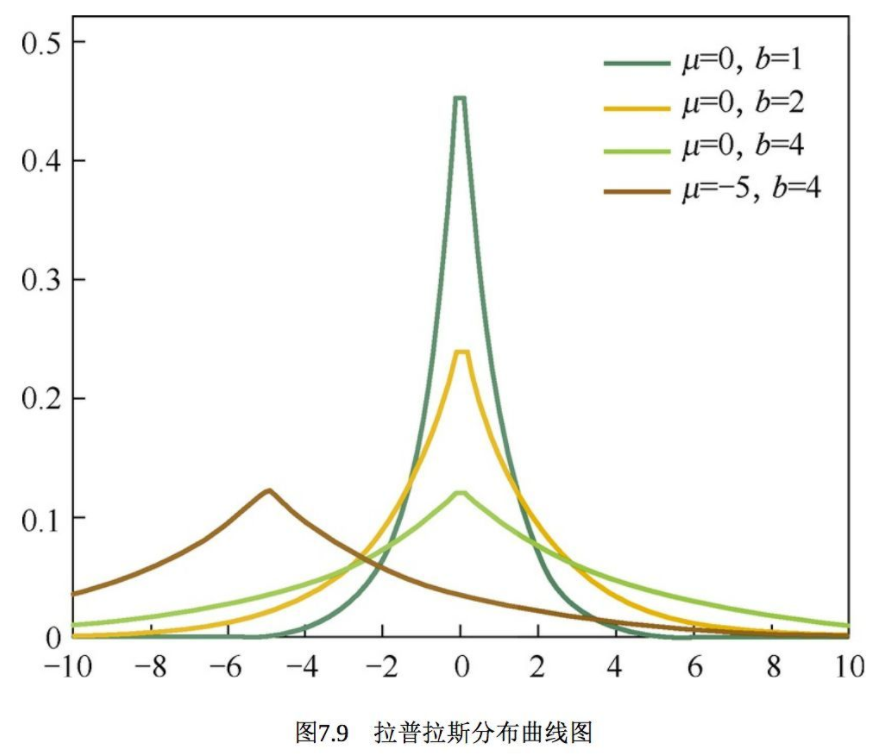
**L1与L1正则化的区别：**

（1）L1正则化是指在损失函数中加入权值向量w的绝对值之和，即各个元素的绝对值之和，L2正则化指在损失函数中加入权值向量w的平方和。（2）L1的功能是使权重稀疏，而L2的功能是使权重平滑。

**L1产生稀疏性的原因（L2不可以）：**

三种答案：（1）采用上述的画图方式；（2）采用函数解释：在损失函数中，对L1/L2求导可得导数分布情况，如下图。L1导数总是为1或者-1，在L1的体现就是w总是从两端 一直往0点稳定前进，不管过程如何，最终总能走到零点。而L2的导数则是越靠近0则越小，因此L2几乎不可能靠近零点，因为在极小值的时候，其导数/梯度也会变得非常小。（3）采用贝叶斯先验：L1相当于对权重w引入拉普拉斯先验，L2相当于引入高斯先验。从下面的分布图可以看出，当μ=0时，高斯分布在极值点处是平滑的，也就是高斯先验分布认为w在极值点附近取不同值的可能性是接近的。但对拉普拉斯分布来说，其极值点是一个尖峰，因为拉普拉斯先验分布中w取值为0的可能性要更高。

L1是假设参数服从拉普拉斯分布，利于保证权值向量的稀疏性，因此在特征选择中很有用，它会把不重要的特征权重置为0，而L2只能近似0；L2是假设参数服从高斯分布，利于防止过拟合。让L2范数的规则项||W||2最小，可以使得W的每个元素都很小，都接近于0，但与L1范数不同，它不会让它等于0，而是接近于0，这里是有很大的区别的。而越小的参数说明模型越简单，越简单的模型则越不容易产生过拟合现象。因此，通过L2范数，我们可以实现了对模型空间的限制，从而在一定程度上避免了过拟合。

一句话总结就是：L1会趋向于产生少量的特征，而其他的特征都是0，而L2会选择更多的特征，这些特征都会接近于0。Lasso (L1)在特征选择时候非常有用，而Ridge (L2) 就只是一种规则化而已。

**交叉验证**：先把训练数据库分成几份，然后取一部分做训练集，一部分做测试集，然后选择不同的λ用这个训练集来训练N个模型，然后用这个测试集来测试我们的模型，取N模型里面的测试误差最小对应的λ来作为我们最终的λ。λ为正则项前面的系数。