# 弱监督综述

[**https://blog.csdn.net/lyxleft/article/details/89742685**](https://blog.csdn.net/lyxleft/article/details/89742685)

1. Arachie 等人（https://arxiv.org/abs/1805.08877）在 AAAI 2019 上针对无标签分类器的训练任务提出了一种弱监督的方法——对抗性标签学习，该方法利用投影的「primal-dual」梯度下降法最小化分类器错误率的上限。最小化这个界限可以防止在弱监督方法中出现偏差和依赖。
2. 在 CVPR 2019 上，Zeng 等人（https://arxiv.org/pdf/1904.00566.pdf）针对像素级图像标注问题，提出了一种多源弱监督视觉显著性检测模型，他们设计了一个分类网络（CNet）和一个标题生成网络（PNet），分别学习预测对象以及生成图像标题。同时，他们还设计了一种能够在网络之间传播监督信号的注意力迁移损失，从而使通过一个监督源训练的网络能够帮助训练使用另一个监督源训练的网络。
3. Wu 等人（https://arxiv.org/abs/1805.02333）在 ACL 2018 上针对开放环境下的人机对话任务开发了一种利用未标注数据学习选择基于检索的聊天机器人的响应的匹配模型的方法。该方法采用序列-均衡结构(Seq2Seq)模型作为弱标注器来判断未标注对的匹配程度，然后对弱信号和未标注数据进行学习。

# 线性代数篇

**向量：一个向量是一列数。这些数是有序排列的。**

**X = [x1,**

**X2]**

**矩阵：一个二维数组。**

**张量：坐标超过二维的数组。**

**有时我们通过将向量元素作为行矩阵写在文本行里。**

**C = A + b**

**这种隐式地复制向量b到很多位置的方式叫广播。**

**矩阵和向量相乘：C=AB**

****

**对应元素的乘积：**

****

**矩阵乘积的转置有着简单的形式：**

**(AB)T =BT AT**

**单位矩阵和逆矩阵：**

**A-1** A = In

A **A-1** = In

**逆矩阵存在条件：**

**必须是方阵，所有的列向量必须是线性无关的，也就是说某一个特征不能有其他特征线性组合出来，一个列向量线性相关的方阵称为奇异矩阵。**

**范数：**

**有时我们需要衡量一个向量的大小。在机器学习中我们经常使用称为范数(norm)的函数来衡量向量的大小。形式上，LP 的范数定义**

****

**有时候我们也希望衡量矩阵的大小。在深度学习中，最常见的做法是使用Frobenius范数，即**

****

**计算向量之间的夹角**

****

**对角矩阵：**

****

**相当于广播**

**通过将一些矩阵限制为对角矩阵，我们可以得到计算代价较低的算法。**

**正交矩阵：**

**正交矩阵求逆代价很小。**

**特征分解：**

**许多数学对象可以通过将他们分解成多个组成部分或者找到他们的一些属性来更好的理解，比如分解质因数。**

****

**矩阵A的特征值和特征向量分别为https://images0.cnblogs.com/blog/650633/201407/141700231939476.png和**x**，记为https://images0.cnblogs.com/blog/650633/201407/141700235376905.png，该式子可理解为向量**x**在几何空间中经过**矩阵A**的变换后得到向量https://images0.cnblogs.com/blog/650633/201407/141700238651105.png。由此可知，向量**x**经过矩阵**A**变换后，方向并无改变（反方向不算方向改变），只是伸缩了https://images0.cnblogs.com/blog/650633/201407/141700244439760.png倍。**

**因此，将特征向量看成基向量，矩阵就是这些基向量向对应的特征值伸展所需的数学运算。给定一个矩阵，就可以找出对应的基（特征向量），及透过向量变换（矩阵），这些基的伸展（特征值**

**奇异值分解：**

**谱分析的基础是对称阵**[**特征向量**](https://baike.baidu.com/item/%E7%89%B9%E5%BE%81%E5%90%91%E9%87%8F/8663983)**的分解，而奇异值分解则是谱分析理论在任意矩阵上的推广。**

**Moore-Penrose 伪逆**

**迹运算：**

**迹运算返回的是矩阵对角元素的和：**

****

**行列式：**

**记作det(A),是一个将方阵A映射到实数的函数。行列式等于矩阵特征值的乘积。行列式的绝对值用来衡量矩阵参与矩阵乘法后空间扩大或者缩小了多少。如果行列式=0，那么空间至少沿着某一维完全收缩了，使其失去了所有的体积；如果行列式是1，那么这个转换保持空间体积不变。**

**实例分析：主成分分析（PCA）**

# 概率论篇

**概率分布：**

**用来描述随机变量或一簇随机变量在每一个可能取到的状态的可能性大小。我们描述概率分布的方式取决于随机变量是离散的还是连续的。**

1. **概率质量函数 数据类型是离散的**
2. **概率密度函数 数据类型是连续的**

**边缘概率分布**

**条件概率分布**

****

**条件概率的链式法则：**

**P(a,b,c) = P(a|bc) P(b|c) P(c)**

**期望、方差、协方差、协方差矩阵**

**常用的概率分布：**

1. **伯努利分布**
2. **高斯分布**
3. **高斯混合模型是概率密度的万能近似器，在这种意义下，任何平滑的概率密度都可以用具有足够多组件的高斯混合模型以任意精度来逼近。**

**贝叶斯规则：**

**我们经常会需要在已知P(Y|X)时计算P(X|Y)。幸运的是，如果还知道P(X),我们可以用贝叶斯规则来实现这一目的。**

**信息论：**

**主要使用信息论的一些关键思想来描述概率分布或者量化概率分布之间的相似性。**

**信息论的基本想法是一个不太可能的事件居然发生了，要比一个非常可能的事件发生提供更多的信息。**

1. **自信息：**
2. **香农熵：那些确定性的分布具有较低的熵；那些接近均匀分布的概率分布具有较高的熵。**
3. **如果对于同一个随机变量x有两个单独的概率分布P(x)和Q(x)，可以使用KL散度来衡量这两个分布的差异。**

****

**结构化概率模型：**

**我们可以把概率分布分解成许多因子的乘积形式，而不是使用单一的函数来表示概率分布。当用图表示这种概率分布的分解时，我们把它称为结构化概率模型或者图模型。**

# 数值计算

**基于梯度的优化方法：**

**大多数深度学习算法都涉及某种形式的优化。优化指的是改变x以最小化或者最大化某个函数f(x)的任务。我们把最小化或者最大化的函数目标称为目标函数或准则。当我们对其进行最小化时，也把它称为代价函数、损失函数或误差函数。**

**这个函数的导数表明如何缩放输入的小变化才能在输出获得相应的变化:**

****

**因此导数对于最小化一个函数很有用，因为它告诉我们如何更改x来略微的更改y。**

**梯度之上：Jacobian 和 Hessian矩阵**

**有时我们需要计算输入和输出都为向量的函数的所有偏导数。包含这样的偏导数的矩阵称为Jacobian矩阵。**

**Hessian矩阵作矩阵分解(求特征向量)类似于求一维函数的二阶导数，可以用于判断临界点时局部最大值还是局部最小值。**

**牛顿法：**

**牛顿法基于一个二阶泰勒展开式来近似x(0)附近的f(x)：**

****

**当附近的临界点时最小点时牛顿法才适用，而梯度下降不会被吸引到鞍点。**

**约束优化：**

**我们希望在x的某些集合S中找到f(x)的最大值或者最小值。这称为约束优化。在约束优化中，集合S内的点x称为可行点。**

**KKT方法是针对约束优化非常通用的解决方案。（SVM）**

**广义拉格朗日乘子法。**

**实例：线性最小二乘**

**假设我们希望找到最小化下式的x的值**

****

****

# 机器学习基础

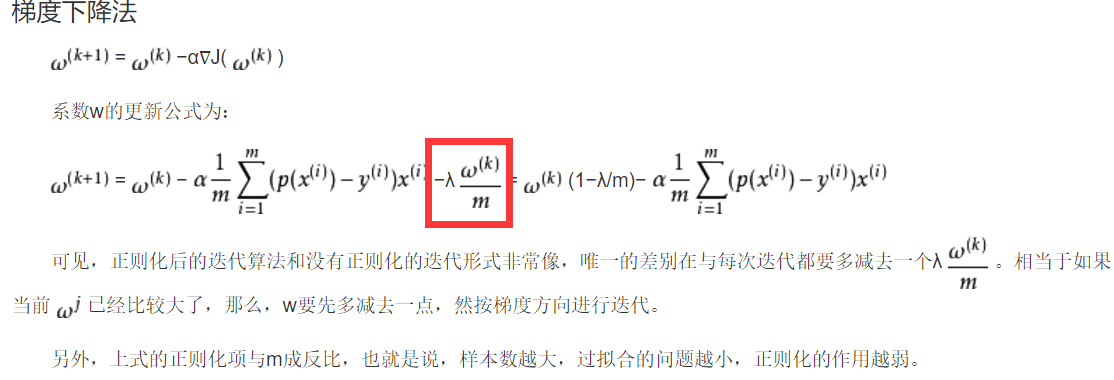
1. **分类**
2. **输入缺失分类**
3. **回归**
4. **转录**
5. **机器翻译**
6. **结构化输出**
7. **异常检测**
8. **合成和采样**
9. **缺失值填补**
10. **去噪**

**正则化：**

**算法的效果不仅很大程度上受影响于假设空间函数的数量，也取决于这些函数的具体形式。**

**权重衰减**

**可以加入权重衰减来修改线性回归的训练标准。带权重衰减的线性回归最小化训练集上的均方误差和正则化的和J（w），其偏好与平方L2 范数比较小的权重。**



**最大似然估计：根据样本分布估计参数。**

**贝叶斯统计：做预测时会考虑所有可能的参数。贝叶斯统计用概率反映知识状态的确定性程度。数据集能够被直接观测到，因此不是随机的。**

**示例：贝叶斯线性回归**

**比如，**

**最重要的是认为参数是服从某一个宽泛分布的。**

贝叶斯线性回归（Bayesian linear regression）是使用统计学中[贝叶斯推断](https://baike.baidu.com/item/%E8%B4%9D%E5%8F%B6%E6%96%AF%E6%8E%A8%E6%96%AD/833912)（Bayesian inference）方法求解的[线性回归](https://baike.baidu.com/item/%E7%BA%BF%E6%80%A7%E5%9B%9E%E5%BD%92/8190345)（linear regression）模型。

因此贝叶斯线性回归的模型本身至少包含2个[超参数](https://baike.baidu.com/item/%E8%B6%85%E5%8F%82%E6%95%B0/3101858)（hyperparameter）。

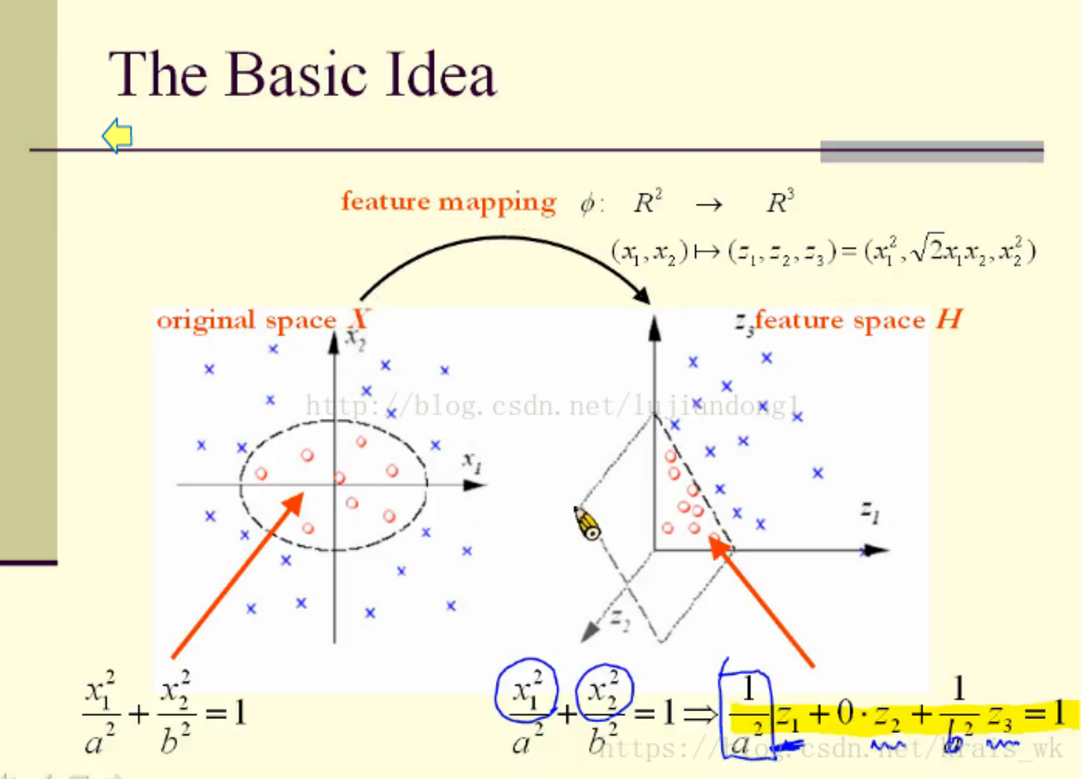
**补充：点估计（point estimation）是用样本统计量来估计总体参数，因为样本统计量为数轴上某一点值，估计的结果也以一个点的数值表示，所以称为点估计。**

**核技巧和流型学习：**

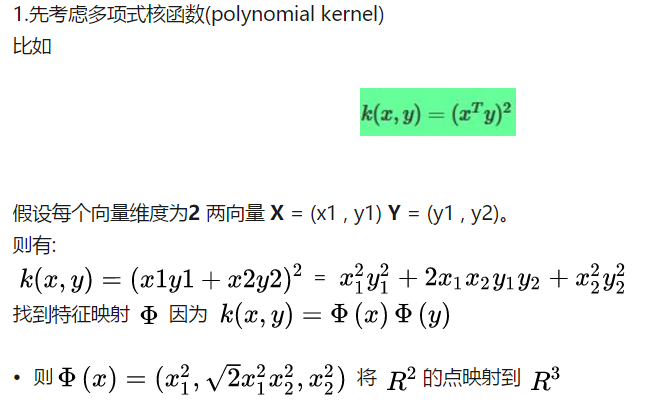
**Kernel的基本思想是，将低维空间不可分数据映射到高纬度的空间，比如说左图的数据是线性不可分的。**

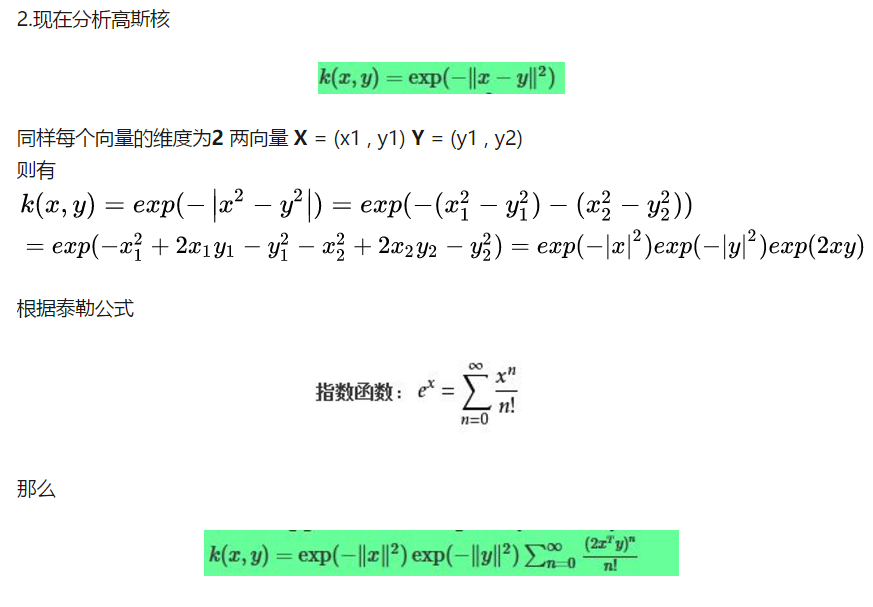
**核技巧观察到许多机器学习算法都可以写成样本间点积的形式。**

**如图所示：**



**运用核函数等价于经过映射函数 Φ(x)Φ(x) 将输入空间的内积 xi⋅xjxi⋅xj 变换为特征空间的内积 Φ(x)⋅Φ(z)Φ(x)⋅Φ(z) ，当映射函数是非线性函数时，学习到的模型就是非线性模型。**





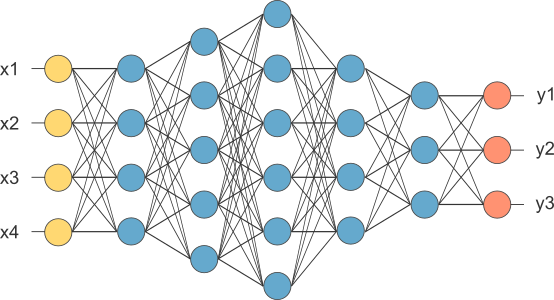
**流形学习：**

**流形指连接在一起的区域。数学上，它是指一组点，且每个点都有其邻域。每个点周围邻域的定义暗示着存在变换能从一个位置移动到其邻域位置。**

**一个流形好比是一个 d 维的空间，在一个 m 维的空间中 (m > d) 被扭曲之后的结果**。

# 神经网络

**目前最广泛使用的定义是Kohonen于1988年的描述，神经网络是由具有适应性的简单单元组成的广泛并行互连的网络，它的组织能够模拟生物神经系统对真实世界物体所做出的交互反应。**

**我总是听到CNN到底有多优秀。但是他们什么时候会失败呢？**

**CNN在解决物体识别和分类问题方面确实取得了巨大的成功。然而，它们并不完美。如果输入CNN的对象处于一个CNN不熟悉的方向，或是出现在CNN从未见过的地方，那么它的预测任务很可能失败。**

**例如，如果你把一张脸颠倒过来，网络将不再能够识别眼睛、鼻子、嘴巴以及它们之间的空间关系。类似地，如果改变面部的特定区域（即切换眼睛和鼻子的位置），网络也可以识别脸部，但是它已经不是真正的脸部了。CNN只学习到了图像中的统计信息，但是他们没有学习基本的思维，即到究竟什么样子才会被称作是脸。**

**Hinton 和 Sabour 从神经科学中借用了一些想法，即认为大脑被组织成了叫做胶囊 的模块。这些胶囊特别擅长处理物体的姿态（位置、大小、方向）、变形、速度、反照率、色调、纹理等特征。**

**他们推测，大脑肯定有一种机制，将低层次的视觉信息传递到它认为能最好的处理这些信息的胶囊。针对卷积神经网络模型性能不足的领域问题，人们提出了胶囊网络和动态路由算法**

**ConvNet as fixed feature extractor.： 其实这里有两种做法：**

1. **使用最后一个fc layer之前的fc layer获得的特征，学习个线性分类器(比如SVM)**
2. **重新训练最后一个fc layer**

# 深度前馈网络

**前馈神经网络或者多层感知机，是典型的深度学习模型。当前馈神经网络被扩展成包含反馈连接时，他们被称为循环神经网络。**

**万能近似定理**

**万能近似定理意味着无论我们试图学习什么函数，我们知道一个大的MLP一定能够表示这个函数。即使MLP能够表示该函数，学习也可能因两个不同的原因而失败。**

1. **用于训练的优化算法可能找不到用于期望函数的参数值。**
2. **训练算法可能由于过拟合而选择了错误的函数。**

**计算图**

**Pytorch计算图学习**

**class** Net(nn.Module):  
 **def** \_\_init\_\_(self):  
 super(Net, self).\_\_init\_\_()  
 self.fc1 = nn.Linear(2, 10)  
 self.fc2 = nn.Linear(10, 10)  
 self.fc3 = nn.Linear(10, 1)  
 **def** forward(self, x):  
 x = F.relu(self.fc1(x))  
 x = F.relu(self.fc2(x))  
 x = self.fc3(x)  
 **return** x

# 深度学习中的正则化

**许多策略被显式的设计来减少测试误差（可能会增大训练误差为代价）。这些策略被称为正则化。**

1. **参数范数惩罚**
2. **L2** 参数正则化（岭回归或Tikhonov正则）
3. **L1** 正则化
4. **作为约束的范数惩罚**

**数据集增强**

**增加噪音数据、对于图像的旋转平移。**

**噪音鲁棒性**

**向隐藏单元添加噪音是值得单独讨论的重要话题。**

**标签平滑技术（向输出目标注入噪音）**

**半监督学习（深度学习）**

**在深度学习的背景下，半监督学习通常指的是学习一个表示H（x）=F（x）。学习表示的目的是使相同类中的样本有类似的表示。**

**Salakhutdinov and Hinton（2008）描述了一种学习回归核机器中核函数的方法，其中建模P（x）时使用的未标记样本大大提高了P（y | x）的效果。**

**多任务学习**

**提前终止**

**当训练有足够的表示能力甚至会过拟合的大模型时，我们经常观察到，训练误差会随着时间的推移逐渐降低但验证集的误差会再次上升。**

**参数绑定和参数共享**

**模型的参数存在一定的相关性，我们想要表达的一种常见依赖是某些参数应该彼此接近。**

**最长用的方法是参数共享，可以显著的减少模型所占的内存。**

**卷积神经网络（参数共享）**

**稀疏表示**

1. **参数稀疏**
2. **X具有稀疏表示h的线性回归**
3. **OMP-1可以成为深度学习框架中非常有效的特征提取器。**

**-----------------------------------------------------------------------------**

1. **import numpy as np**
2. **import math**
3. **from PIL import Image**
4. **#读取图像，并变成numpy类型的 array**
5. **im = np.array(Image.open('lena.bmp')) #图片大小256\*256**
6. **#生成高斯随机测量矩阵**
7. **sampleRate=0.7 #采样率**
8. **Phi=np.random.randn(256\*sampleRate,256)**
9. **#生成稀疏基DCT矩阵**
10. **mat\_dct\_1d=np.zeros((256,256))**
11. **v=range(256)**
12. **for k in range(0,256):**
13. **dct\_1d=np.cos(np.dot(v,k\*math.pi/256))**
14. **if k>0:**
15. **dct\_1d=dct\_1d-np.mean(dct\_1d)**
16. **mat\_dct\_1d[:,k]=dct\_1d/np.linalg.norm(dct\_1d)**
17. **#随机测量**
18. **img\_cs\_1d=np.dot(Phi,im)**
19. **#OMP算法函数**
20. **def cs\_omp(y,D):**
21. **L=math.floor(3\*(y.shape[0])/4)**
22. **residual=y #初始化残差**
23. **index=np.zeros((L),dtype=int)**
24. **for i in range(L):**
25. **index[i]= -1**
26. **result=np.zeros((256))**
27. **for j in range(L): #迭代次数**
28. **product=np.fabs(np.dot(D.T,residual))**
29. **pos=np.argmax(product) #最大投影系数对应的位置**
30. **index[j]=pos**
31. **my=np.linalg.pinv(D[:,index>=0]) #最小二乘,看参考文献1**
32. **a=np.dot(my,y) #最小二乘,看参考文献1**
33. **residual=y-np.dot(D[:,index>=0],a)**
34. **result[index>=0]=a**
35. **return result**
36. **#重建**
37. **sparse\_rec\_1d=np.zeros((256,256)) # 初始化稀疏系数矩阵**
38. **Theta\_1d=np.dot(Phi,mat\_dct\_1d) #测量矩阵乘上基矩阵**
39. **for i in range(256):**
40. **print('正在重建第',i,'列。')**
41. **column\_rec=cs\_omp(img\_cs\_1d[:,i],Theta\_1d) #利用OMP算法计算稀疏系数**
42. **sparse\_rec\_1d[:,i]=column\_rec;**
43. **img\_rec=np.dot(mat\_dct\_1d,sparse\_rec\_1d) #稀疏系数乘上基矩阵**
44. **#显示重建后的图片**
45. **image2=Image.fromarray(img\_rec)**
46. **image2.show()**
47. **--------------------------------------------------------------------------**

**Bagging技术（模型平均技术）**

**Dropout**

**Dropout 提供了正则化一大类模型的方法，计算方便但功能强大。**

**具体而言，Dropout训练的集成包括所有从基础网络除去非输出单元后形成的子网络。**

**对抗训练**

**人类观察者不会察觉原始样本和对抗样本之间的差异，但是网络会做出非常不同的预测。**

**GAN（对抗生成网络）**

**GAN属于无监督学习，由两个神经网络组成。生成器（generator）从潜在空间中随机获取样本，并与真实数据一起作为判别器（discriminator）的输入。判别器是一个经典分类器，作用是把真实数据和生成数据尽量分开。判别器对生成数据的判别结果会返回给生成器，训练生成器生成更多能够成功“骗过”判别器的数据。训练的最终目的是使生成样本的分布与真实数据达到一致，判别器完全不能区分真伪。**

**import** torch  
**import** torch.nn **as** nn  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
  
*# torch.manual\_seed(1) # reproducible  
# np.random.seed(1)  
  
# Hyper Parameters*BATCH\_SIZE = 64  
LR\_G = 0.0001 *# learning rate for generator*LR\_D = 0.0001 *# learning rate for discriminator*N\_IDEAS = 5 *# think of this as number of ideas for generating an art work (Generator)*ART\_COMPONENTS = 15 *# it could be total point G can draw in the canvas*PAINT\_POINTS = np.vstack([np.linspace(-1, 1, ART\_COMPONENTS) **for** \_ **in** range(BATCH\_SIZE)])  
  
*# show our beautiful painting range  
# plt.plot(PAINT\_POINTS[0], 2 \* np.power(PAINT\_POINTS[0], 2) + 1, c='#74BCFF', lw=3, label='upper bound')  
# plt.plot(PAINT\_POINTS[0], 1 \* np.power(PAINT\_POINTS[0], 2) + 0, c='#FF9359', lw=3, label='lower bound')  
# plt.legend(loc='upper right')  
# plt.show()***def** artist\_works(): *# painting from the famous artist (real target)* a = np.random.uniform(1, 2, size=BATCH\_SIZE)[:, np.newaxis]  
 paintings = a \* np.power(PAINT\_POINTS, 2) + (a-1)  
 paintings = torch.from\_numpy(paintings).float()  
 **return** paintings  
  
G = nn.Sequential( *# Generator* nn.Linear(N\_IDEAS, 128), *# random ideas (could from normal distribution)* nn.ReLU(),  
 nn.Linear(128, ART\_COMPONENTS), *# making a painting from these random ideas*)  
  
D = nn.Sequential( *# Discriminator* nn.Linear(ART\_COMPONENTS, 128), *# receive art work either from the famous artist or a newbie like G* nn.ReLU(),  
 nn.Linear(128, 1),  
 nn.Sigmoid(), *# tell the probability that the art work is made by artist*)  
  
opt\_D = torch.optim.Adam(D.parameters(), lr=LR\_D)  
opt\_G = torch.optim.Adam(G.parameters(), lr=LR\_G)  
  
plt.ion() *# something about continuous plotting***for** step **in** range(1000):  
 artist\_paintings = artist\_works() *# real painting from artist* G\_ideas = torch.randn(BATCH\_SIZE, N\_IDEAS) *# random ideas* G\_paintings = G(G\_ideas) *# fake painting from G (random ideas)* prob\_artist0 = D(artist\_paintings) *# D try to increase this prob* prob\_artist1 = D(G\_paintings) *# D try to reduce this prob* D\_loss = - torch.mean(torch.log(prob\_artist0) + torch.log(1. - prob\_artist1))  
 G\_loss = torch.mean(torch.log(1. - prob\_artist1))  
  
 opt\_D.zero\_grad()  
 D\_loss.backward(retain\_graph=**True**) *# reusing computational graph* opt\_D.step()  
  
 opt\_G.zero\_grad()  
 G\_loss.backward()  
 opt\_G.step()  
 print(**'step = {}'**.format(step))  
  
G\_paintings = G(torch.tensor([-0.9, -0.5, 0, 0.1, 0.1]))  
print(G\_paintings)

**切面距离、正切传播和流形正切分类器**

# 深度模型中的优化

**本章主要关注这一类的优化问题：寻找神经网络上的一组参数，它能显著地降低代价函数，该代价函数通常包括整个训练集上的性能评估和额外的正则化项。**

**1.代理损失函数和提前终止**

**我们通常优化代理损失函数，代理损失函数作为原目标的代理，还具备一些优点，例如正确类别的负对数似然通常作为0-1损失的替代。**

1. **批量算法和小批量算法**
2. **病态**
3. **局部极小值**
4. **高原、鞍点和其他平坦区域 Hessian矩阵在局部极小点处只有正特征值。而在鞍点处，Hessian矩阵则同时具有正负特征值。**
5. **对于牛顿法而言，鞍点显然是一个问题。梯度下降旨在朝‘下坡’移动，而非明确寻求临界点。而牛顿法的目标是寻求梯度为0的点，如果没有适当的修改，牛顿法就会跳进一个鞍点。（二阶优化的无鞍牛顿法比较有希望改进）**
6. **悬崖和梯度爆炸，多层神经网络通常存在像悬崖一样的斜率较大的区域，这是由于几个较大的权重相乘导致。遇到斜率较大的悬崖结构时，梯度更新会很大程度地改变参数值，通常会完全跳过这类悬崖结构。（参数弹射）梯度阶段可以干涉步长。**
7. **长期依赖，由于变深的网络模型丧失了学习到先前信息的能力，让优化变得极其困难。（循环神经网络容易出现梯度爆炸的现象）**
8. **非精确梯度，这种情况下我们只能近似梯度。**
9. **局部和全局结构间的弱对应，最终的观点还是建议在传统优化算法上研究怎么选择更加的初始化点，以此来实现目标更切实可行。**
10. **补充（病态条件数表明函数相对于输入的微小变化而变化的快慢程度，如果之前就存在误差的话，矩阵会放大原来的误差。因为条件数大，所以矩阵对输入的误差很敏感）**
11. **优化的理论限制，研究优化算法更现实的性能上界仍然是学术界的一个重要目标。**

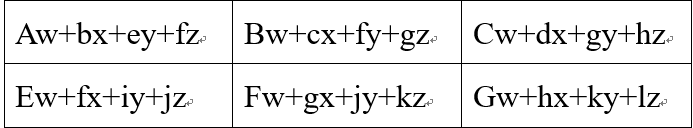
**基本算法**

1. **随机梯度下降（SGD,Bottou 1998）**
2. **动量（Nesterov动量）**
3. **参数初始化策略，初始分布的大小确实对优化的过程的结果和网络泛化能力都有很大的影响。**
4. **自适应学习率算法(AdaGrad, RMSProp, Adam)**
5. **二阶近似方法(牛顿法、共轭梯度、BFGS)**
6. **优化策略与元算法，批标准化是优化深度神经网络中最激动人心的最新的创新方法，他是一个自适应的重参数化的方法。而在神经网络中，每一层的输入在经过层内操作之后必然会导致与原来对应的输入信号分布不同，并且前层神经网络的增加会被后面的神经网络不断的累积放大。这个问题的一个解决思路就是根据训练样本与目标样本的比例对训练样本进行一个矫正，而BN算法（批标准化）则可以用来规范化某些层或者所有层的输入，从而固定每层输入信号的均值与方差。**
7. **坐标下降，在某些情况下，将一个优化问题分解成几个部分，可以很快地解决原问题。如果我们相对于某一个单一变量xi 最小化f(x)，然后相对于另一个变量xj等，反复循环所有的变量，我们保证到达极小值。**
8. **Polyak平均**
9. **监督预训练，有时训练一个较简单的模型来求解问题，然后使模型更复杂会更有效。这些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称为预训练。（贪心监督预训练、FitNets）**
10. **设计有助于优化的模型**
11. **延拓法和课程学习**

# 卷积神经网络

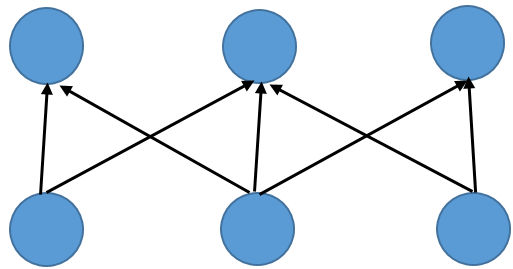
1. 卷积运算 在卷积神经网络的术语中，卷积的第一个参数通常叫做输入，第二个参数叫做核函数。输出有时称为特征映射。输入与核作移动点积运算。

|  |  |  |  |
| --- | --- | --- | --- |
| a | b | c | d |
| e | f | g | h |
| i | j | k | l |



|  |  |
| --- | --- |
| w | x |
| y | z |

1. 动机 卷积运算通过三个重要的思想帮助改进机器学习系统：稀疏交互、参数共享、等变表示。
2. 稀疏连接



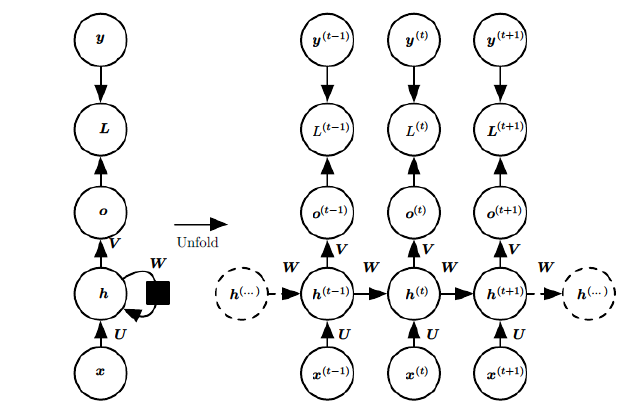
1. 参数共享是指在一个模型的多个函数中使用相同的参数。卷积运算中的参数共享保证了我们只需要一个参数集合，而不是对每一个位置都需要学习一个单独的参数集合。这意味着在卷积网络中尽管直接连接都是很稀疏的，但处在更深的层中的单元可以间接地连接到全部或者大部分输入图像。
2. 池化，对于感兴趣特征的位置运行聚类算法，这种方法对于每幅图像产生一个不同的池化区域集合，另一个是先学习一个单独的池化结构，在应用到全部的图像中。
3. 卷积与池化作为一种无限强的先验
4. 基本卷积函数的变体
5. 结构化输出，可以输出高维的结构化向量。
6. 卷积神经网络提取特征，<https://blog.csdn.net/weixin_42078618/article/details/83895555>
7. 卷积神经网络的神经科学基础

# 序列建模：循环与递归网络

1.展开计算图

计算图是一种形式化一组计算结构的方式。很多循环神经网络使用以下公式定义隐藏单元的值。





1. 基于编码-解码的序列到序列框架
2. LSTM

# 线性因子模型（稀疏编码）

在本章中描述了一些基于潜变量最简单的概率模型：线性因子模型，线性因子模型通过随机线性解码器函数来定义，该函数通过对H的线性变换以及添加噪音生成x。

# 自编码器

自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。

h

X

f g

r

自编码器具有两个组件：编码器f（将x映射到h）和解码器g（将h映射到r）

欠完备的自编码器

自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编码器称为欠完备自编码器。学习过程可以简单描述为最小化一个损失函数。当解码器是线性的且L是均方误差，欠完备的自编码器会学习出与PCA相同的生成子空间。

正则自编码器

1. 结合稀疏惩罚
2. 去噪自编码器
3. 收缩自编码器
4. 随机编码器和解码器，将编码函数推广到编码分布，再输入输出加入噪音。

去噪自编码器（DAE）

使用自编码器学习流形

# 表示学习

一般而言一种好的表示可以使后续的学习任务更容易。表示学习特别有趣，因为它提供了进行无监督学习和半监督学习的一种方法。半监督学习通过进一步学习未标注的数据，来解决过拟合的问题。具体地，我们可以从未标注数据上学习很好的表示，然后利用这些表示来解决监督学习问题。

深度学习网络的表达方式有局部表达、分布式表达和稀疏表达3种。

1. 贪心逐层无监督预训练，尝试获取输入的分布。
2. 贪心逐层无监督预训练依赖于单层表示学习算法，如RBM、单层自编码器、稀疏编码模型。
3. 无监督预训练在处理单词时特别有用。然而在处理图像时是不太有用的，可能是因为图像已经在一个很丰富的向量空间中，其中的距离只能提供低质量的相似度。
4. 监督预训练