机器学习Project报告

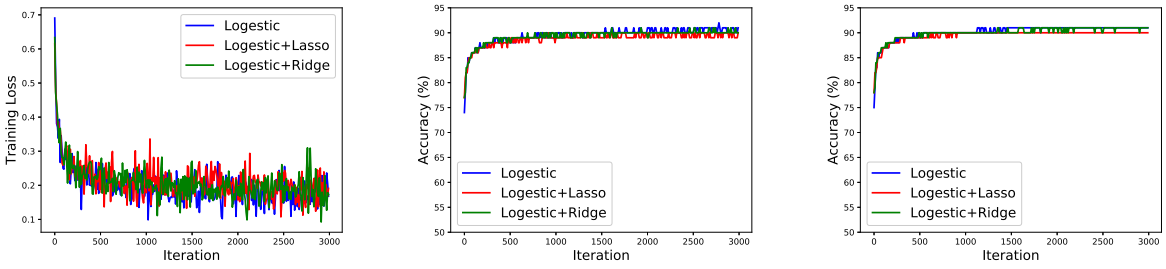
丁雨成 517021910416

1 Logestic Regression

1.1 实现

我首先实现了Logestic Regression及其带有Lasso，Ridge正则项的形式。实现的方式大致如下：首先利用torch.randn函数随机生成一个形状为$(10, 785)$的矩阵（其中785为图片的输入#784+biase项#1，每个参数的取样符合均值为0，方差为$1e^{-4}$的正态分布）。然后采用MSE的损失函数，对模型求导，得到grad矩阵。是否添加正则项的区别就在于grad矩阵的求法：如果添加正则项，则grad矩阵需要在未添加时的grad矩阵上添加一个对正则化的梯度：对model的每个元素x而言，Lasso的梯度是$\frac{x}{|x|}(x \neq 0)$，Ridge的梯度是$2x$。得到梯度后，利用梯度下降法更新模型参数，重复5个epoch。在实验中，我们设置$\text{train_batch} = 100$, $\text{learning\_rate} = 0.1$, 正则系数$\lambda = 0.001$。

在这里，绘制出训练损失值，训练准确率，测试准确率的变化情况，并给出最终的测试准确率。



(a) Training Loss曲线　(b) Train Accuracy曲线　(c) Test Accuracy曲线

图 1: Logestic Regression训练l曲线

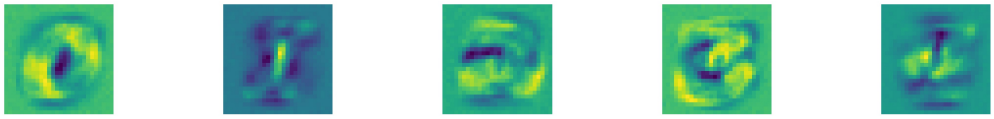
表 1: 不同损失函数下最终准确率。

Loss	logestic	logestic+lasso	logestic+ridge
Train Acc	91%	90%	91%
Test Acc	91%	90%	91%

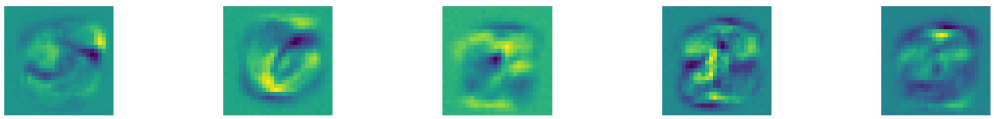
我观察到在三种不同方法下，无论是训练的损失变化，还是准确率变化都比较接近。我将在这一部分的第3小节研究lasso和ridge究竟带来了什么变化。

1.2 可视化与逻辑推断

我们将训练好的模型视为十个分类器，每个分类器为一个785维的向量，除去Bias项之外，剩余的784个维度分别对应着图片(28×28)的像素，我将其可视化为图像，进行模型的逻辑推断。



(a) 0 分类器 (b) 1 分类器 (c) 2 分类器 (d) 3 分类器 (e) 4 分类器



(f) 5 分类器 (g) 6 分类器 (h) 7 分类器 (i) 8 分类器 (j) 9 分类器

图 2: 分类器的可视化。

根据模型的可视化规则，色调较“暖”的位置对应的模型的参数值较大，为对识别该数字有正面作用的像素点。我的理解是分类器基本学习到了对应数字的分布，如果像素点分布在图中暖色调位置附近，则为对应数字的可能性较大。但即使是相同数字，不同图片的差异仍然较大，模型融合了这些图片的分布得到一个总体分布，因此显得较为模糊。

1.3 结论/假设及实验

1. 假设： Ridge会使得模型的l_2-norm 降低，Lasso会使得模型变得稀疏？

2. 实验设计：将正则化项钱的系数λ取不同的值，比较相同的λ下不加任何正则化项，加入Lasso正则化，Ridge正则化情况下，比较最终训练得到的模型的非0个数，l_2-模。值得注意的是，对比实验需要以相同的模型初始化。此外，由于训练过程中模型参数不会真正达到0，我采取的方式是绝对值大于0.001的参数视为“非0”参数。当某参数绝对值小于0.001时，我认为对结果影响很小，可忽略不计。

3. 实验结果：我按照设计的方法进行训练，总共训练了1个epoch，并将实验结果以表格的形式记录下来，如表 2 所示。可以看到，总体而言，lasso可以显著的减少模型中“非0”元素的个数。而ridge则对模型l_2-norm的减小很有帮助。并且，二者的效果随着λ的增大而愈发显著。但是，我观察到了一个异常数据，如表中加粗的数据显示，当λ取值为0.1时，lasso优化得到的非0个数远远高于λ去其他数值时的数量，甚至高于相同条件下的ridge优化。我从参数设置中找到了原因：在lasso优化过程中，对某一模型参数x而言，正则化项的梯度为$g = \pm\lambda$，而x的优化可表示为$x_{t+1} = x_t - \gamma g$。当我们取学习率$\gamma = 0.1$，正则系数$\lambda = 0.1$时，我们每次更新的量$\| \gamma g \| = 0.01 > 0.001$，无法通过对正则项的优化使得模型参数小于设定的阈值，即在该条件下，模型中“非0”参数个数很难减少。因此，lasso优化是的模型变得稀疏实际上需要一定的条件，我猜测条件之一是模型参数更新的量需要小于设定的阈值。为此，我补充了一个实验，同样取$\lambda = 0.1$，而设置学习率$\gamma = 0.01$，得到的结果在表格最下方，与

表 2: 不同λ,，损失函数下模型的模。

$\lambda = 0.00001$	logestic	logestic+lasso	logestic+ridge
l_2-norm	8.38	8.36	8.36
#nonzeros	7684	7225	7670
$\lambda = 0.0001$	logestic	logestic+lasso	logestic+ridge
l_2-norm	8.39	8.18	8.19
#nonzeros	7647	5209	7636
$\lambda = 0.001$	logestic	logestic+lasso	logestic+ridge
l_2-norm	8.38	7.56	6.82
#nonzeros	7664	3776	7549
$\lambda = 0.01$	logestic	logestic+lasso	logestic+ridge
l_2-norm	8.37	5.96	3.23
#nonzeros	7666	3004	5940
$\lambda = 0.1$	logestic	logestic+lasso	logestic+ridge
l_2-norm	8.37	3.99	1.57
#nonzeros	7648	**7110**	5346
$\lambda = 0.1, \gamma = 0.01$	logestic	logestic+lasso	logestic+ridge
#nonzeros	-	**831**	-

预期相符。而ridge优化则具有天然的优势：当模型参数接近0的时候梯度会相应的减小，使得调控更加精细，而不会受参数的制约。实验中也反映了这一点，ridge优化得到的结果符合规律，无异常数据出现。

4. 实验结论：Rdige优化基本上可以保证模型的第二范数降低，Lasso则在满足一定条件（比如每次更新的步长较小时）使得大部分模型参数趋近于0。

2 LDA

2.1 实现及实验

我利用课本上所讲的内容实现了LDA解决二分类问题。我们每次这对一个数字进行二分类：取10000张训练图片，将标签是该数字的数据加入正样本，标签非该数字的加入负样本。利用课本上提供的最佳单位向量计算公式以及numpy提供的矩阵乘法得到用于分类的方向，求得正负样本在该向量上的投影的均值，记为μ_+, μ_-。对于每个测试数据，如果其投影值更接近μ_+，则分类为正样本；如果更接近μ_-，则分类为负样本。注意到由于原始测试数据中正负样本分布及其不均衡，如果分类器一般将样本分为负样本也会因为负样本数量更多而造成准确率很高的假象，因此我们根据样本标签，取相同数量的正负测试样本进行实验，得到如表 3 所示的结果。

表 3: 不同数字利用lda进行二分类的准确率

Target Number	0	1	2	3	4	5	6	7	8	9
Accuracy	81.9%	88.5%	89.1%	95.1%	88.9%	82.4%	95.1%	94.4%	92.3%	88.5%

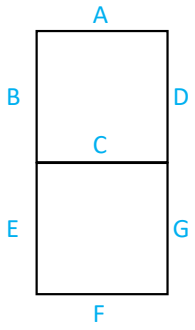


图 3: LED版本的数字显示示例

2.2 个人理解

为什么相同的方法在不同的数字上的表现差异如此之大呢？我猜想可能跟不同数字分布的相似程度有关。为了统计数字的分布情况，我统计了在LED显示数字的规则下，LED的每一条边都有哪些数字涉及，并将统计结果呈现在表 4 中。我们根据每条边上数字的数目对其“特异性进行”排序，数量越小的“特异性”越高，并依次给5个等级的特异性赋予从高到低的分数（从5到1）。我们统计了每个数字的平均得分，结果依次为：2.8，1.5，3.0，2.2，2.5，2.6，3.0，1.7，2.9，2.5。我将特异性平均结果和最终的准确率进行回归和假设检验，原假设为实验的准确率和特异性平均得分无关，得到了p-value结果为0.1061>0.05，因此无法拒绝原假设。可见在置信度为95%条件下，不能认为最后的准确率和我设定的特异性分数有关。可能的原因是：（1）手写的数字和LED中显示的数字分布差异较大，用这种方法无法正确的统计数字的分布；（2）我设定的特异性打分机制不够合理；（3）最后的准确率和这种特异性无直接关系。

3 Neural Networks

3.1 实现

我手动实现了简单的神经网络。主要的做法是首先实现不同的layers的正向、反向传播函数。在本次作业中，我实现了全连接层，Relu层和Sigmoid层的正向和反向传播函数。利用这些基本的函数，搭建了一个简单的神经网络，并且用基本的函数组合实现了其正向传播和反向传播的过程。该网络输入为784维，有两个隐藏层，隐藏神经元数目分别为300，100，最终输出一个10维向量。我在MNIST数据集上进行实验，取得了比较良好的效果。实验中，我采取的是MSE损失函数，梯度的求解参考了课本中的公式。

表 4: 不同数字边分布情况统计

Number	0	1	2	3	4	5	6	7	8	9	总数	特异性得分
A	1		1	1		1	1	1	1	1	8	2
B	1				1	1	1		1	1	6	4
C			1	1	1	1	1		1	1	7	3
D	1	1	1	1	1			1	1	1	8	2
E	1		1				1		1		4	5
F	1		1	1		1	1		1	1	7	3
G	1	1		1	1	1	1	1	1	1	9	1

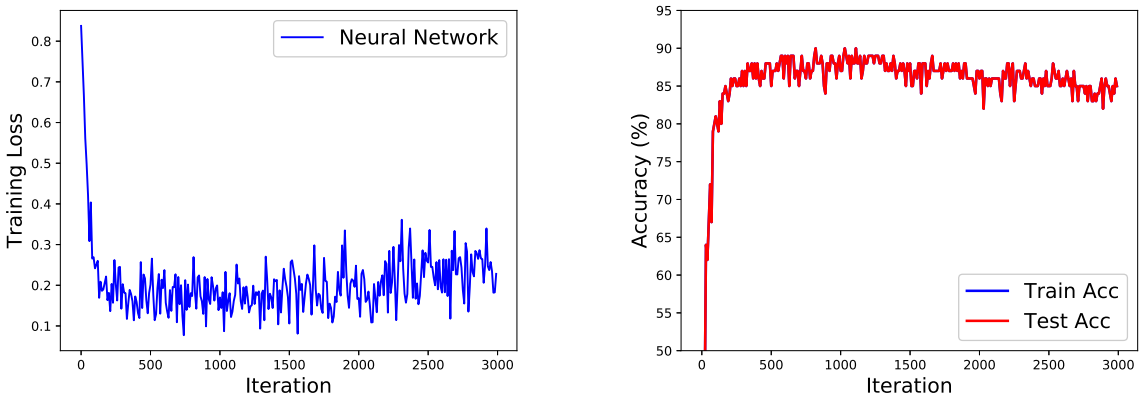


图 4: 简单神经网络的损失及准确率变化

3.2 实验

将batch size设置为100，学习设置为0.3，重复5个epoch，并记录损失函数值随训练代数变化的情况以及每个epoch训练结束时的测试准确率。结果如图 4 所示。我注意到了两点问题：（1）训练后期，损失函数值不降反升；（2）训练的准确率和测试的准确率变化曲线完全一致。针对第一点，我的猜想是学习率设置的过大，具体原因将在Section 4.2.2 中进行详细分析。针对第二点，我首先想到的是程序中存在bug，将测试集错误的当成了训练集进行准确度测试，后来发现并不存在这个错误。所以我认为的可能的原因是网络结构太过简单，丝毫没有过拟合现象的发生。

为了验证最后损失值的上升是否是由于学习率太大引起的，我尝试着加入learning decay：即在训练的过程中逐渐减小学习率。我设置初始学习率仍然为0.3，但每个epoch后，学习率变为原来的0.9倍。补充实验的结果如图 5 所示，可以看到训练曲线最后虽然还是在震荡，但并没有图 4 中明显的的上升趋势，这一点在准确率曲线上可以更清晰的观察到。

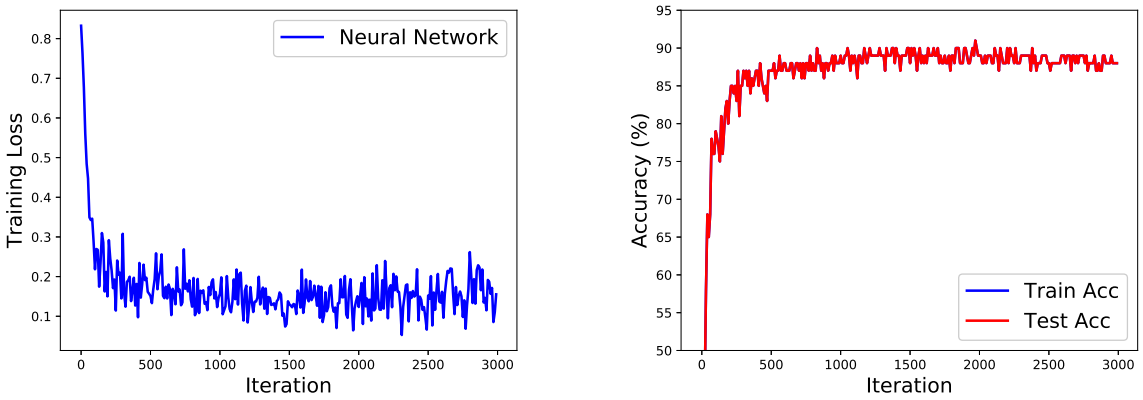


图 5: 加入学习率衰减后神经网络的损失及准确率变化

4 Convolutional Neural Networks

4.1 实现

我首先通过Pytorch实现了一个卷积神经网络。网络的结构依次为卷积层1，卷积核大小为5，通道数为1，数量为6；Relu层；卷积层2，卷积核大小为5，通道数为6，数量为16；紧接着是Relu层加最大池化层；三个全连接层，输出神经元数分别为120，84，10，前两个配合Relu激活函数，最后一个输出直接连接softmax层。

程序主要由训练和测试两部分组成，每次从训练集中选取一定数量的图片，利用提供的库Pytorch的反向传播算法求取梯度，采取交叉熵损失函数，利用梯度下降法优化模型。优化过程中记录训练损失，优化完成后利用测试集进行模型评估。

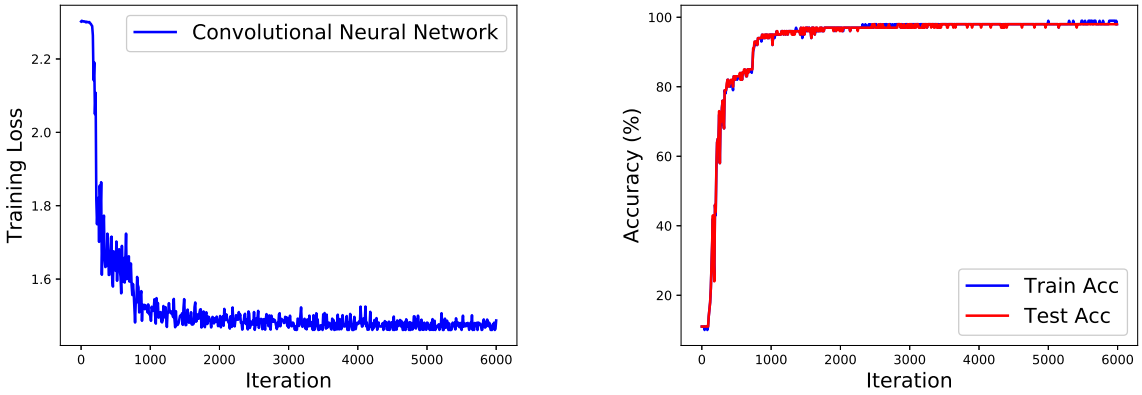
我的基本实验超参数设置如下：学习率设置成0.3，训练batch size设为100，一共进行10个epoch的训练，训练结束后进行测试。实验中训练损失的变化情况如图 6 所示，最终测试准确率为98%。此时过拟合现象依然不太严重，训练准确率曲线和测试准确率曲线大致重合。

4.2 学习率与优化速度

我比较了梯度下降法在不同学习率设置下的优化速度，将结果绘制在图 7 中，并尝试给出理论上的解释。因为此处主要是针对优化结果进行比较，而优化的目标是最小化训练损失函数，所以仅仅绘制出训练损失的变化曲线，而忽略准确率等因素来更方便的观察优化结果。

4.2.1 实验设置

我们比较了学习率为0.01, 0.05, 0.1, 0.5情况下训练损失的变化情况。图 7 显示了完整的训练损失变化曲线，为了更清晰地看到模型训练初期和后期的损失情况，我在图 8 中分别显示了前10个epoch和最后10个epoch的损失变化。可以看到，在模型训练初期，模型的收敛速度随着学习率的增大而增加，而在训练后期，学习率为0.5的曲线出现了令人意外的变化，损失值不降反升，而其余三个学习率条件下训练的得到的模型的损失都很低。



(a) 训练损失变化的曲线　　(b) 训练/测试准确率变化曲线

图 6: 基本设置下卷积神经网络的训练曲线

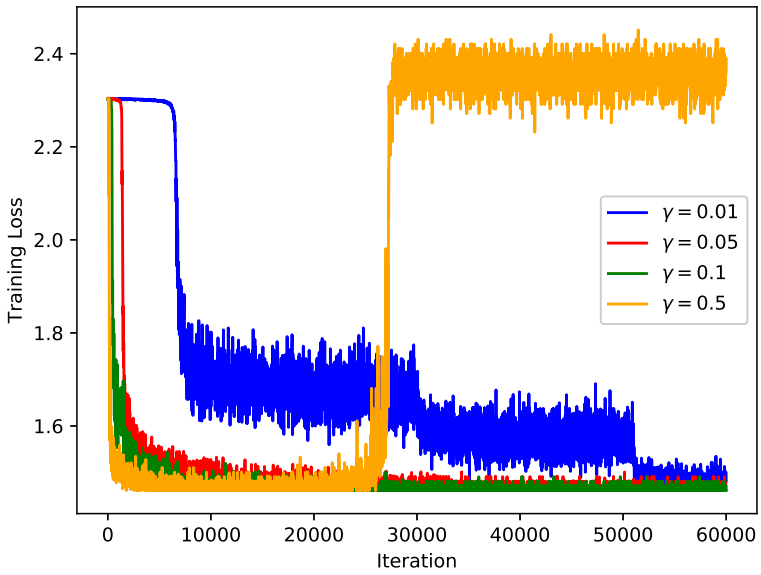


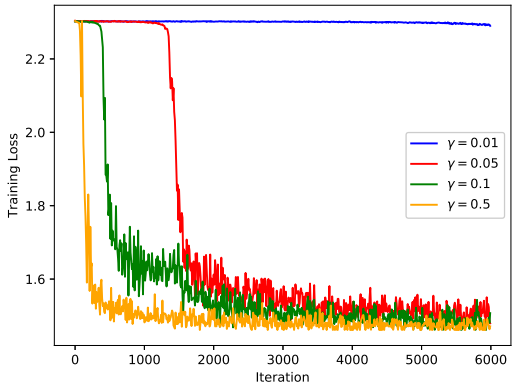
图 7: 不同学习率下神经网络损失变化的完整曲线

4.2.2 理论解释

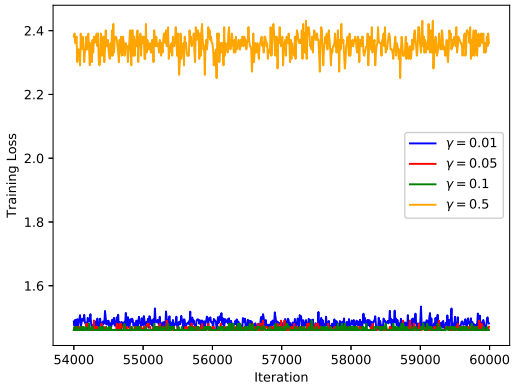
通过查阅有关优化的相关资料并结合我自己做过的研究工作，我尝试对优化结果进行解释。首先做出如下两个一般假设：

1. 损失函数的期望（记为f）满足L-smooth性质，即对于模型$\forall \mathbf{x}, \mathbf{y}$:

$$f(\mathbf{x}) \leq f(\mathbf{y}) + \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle + \frac{L}{2} \parallel \mathbf{x} - \mathbf{y} \parallel^2$$



(a) 前10个epoch中损失的变化



(b) 最后10个epoch中损失的变化

图 8: 不同学习率下神经网络的损失变化

2. 每次随机取样计算得到的梯度的方差满足如下条件：

$$\forall t:\ \mathbb{E}\left[\| \nabla f\left(\mathbf{x}_t\right)-\nabla F\left(\mathbf{x}_t, \xi_t\right) \|^2\right] \leq \sigma^2.$$

其中$F\left(\mathbf{x}_t, \xi_t\right)$模型在取到的样本集合为$\xi_t$时的损失函数，简化记为$\mathbf{g}_t$。

我将迭代过程表示为：

$$\mathbf{x}_{t+1}=\mathbf{x}_t-\gamma \mathbf{g}_t$$

则利用假设1，相邻代的模型满足如下条件：

$$\begin{aligned} f(\mathbf{x}_{t+1}) \leq & f(\mathbf{x}_t)+\langle \nabla f(\mathbf{x}_t), \mathbf{x}_{t+1}-\mathbf{x}_t\rangle+\frac{L}{2}\| \mathbf{x}_{t+1}-\mathbf{x}_t \|^2 \\ = & f(\mathbf{x}_t)-\gamma\left\langle \nabla f(\mathbf{x}_t), \nabla f(\mathbf{x}_t)+\mathbf{g}_t-\nabla f(\mathbf{x}_t)\right\rangle+\frac{\gamma^2 L}{2}\| \nabla f(\mathbf{x}_t)+\mathbf{g}_t-\nabla f(\mathbf{x}_t) \|^2 \\ \leq & f(\mathbf{x}_t)-\gamma\left\langle \nabla f(\mathbf{x}_t), \nabla f(\mathbf{x}_t)+\mathbf{g}_t-\nabla f(\mathbf{x}_t)\right\rangle+\gamma^2 L\| \nabla f(\mathbf{x}_t) \|^2+\gamma^2 L\| \mathbf{g}_t-\nabla f(\mathbf{x}_t) \|^2 \end{aligned}$$

注意到$\mathbb{E}[\mathbf{g}_t]=\nabla f(\mathbf{x}_t)$, 如果我们对两边同时求期望，可以得到如下不等式：

$$\mathbb{E}\left[f(\mathbf{x}_{t+1})\right] \leq \mathbb{E}\left[f(\mathbf{x}_t)\right]-(\gamma-\gamma^2 L)\mathbb{E}\left[\| \nabla f(\mathbf{x}_t) \|^2\right]+\gamma^2\sigma^2 L$$

对$t=1,2,\cdots,T$,进行累加并移项：

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\| \nabla f(\mathbf{x}_t) \|^2\right] \leq \frac{\mathbb{E}\left[f(\mathbf{x}_1)-f(\mathbf{x}_{T+1})\right]}{(\gamma-\gamma^2 L)T}+\frac{\gamma\sigma^2 L}{\gamma^2-\gamma^2 L} \leq \frac{\mathbb{E}\left[f(\mathbf{x}_1)\right]-f(\mathbf{x}^*)}{(\gamma-\gamma^2 L)T}+\frac{\gamma^2\sigma^2 L}{\gamma-\gamma^2 L}$$

当$\gamma L \leq \frac{1}{2}$时，可以简化成

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\| \nabla f(\mathbf{x}_t) \|^2\right] \leq 2\frac{\mathbb{E}\left[f(\mathbf{x}_1)\right]-f(\mathbf{x}^*)}{\gamma T}+2\gamma\sigma^2 L$$

等式左边是对于非凸优化收敛的一般表示形式。当$T \to +\infty$且$\gamma \to 0$时，等式右边趋近于0，说明左边的梯度基本为0，意味着模型接近了一个鞍点。

从最后的结果来看，收敛的速度主要取决于等式右边的两项。在训练的初始阶段，第一项起主要作用，增大学习率有助于快速的降低损失值，对应着我们在实验中观察到的收敛速度随学习率在一定范围内的增大而加快。随着训练的进行，第二项开始起到主要作用，此时，学习率过大可能以两种方式导致模型效果变差：（1）学习率太大导致第二项太大，进而引起模型质量的下降；（2）学习率太大导致模型更新过大，进而导致模型直接跳出了这个鞍点所在的区域。这就对应着我们在图 4 和 7 中观察到的训练损失增大的情况。在实际部署中，可以通过学习率递减的方式，在初始阶段使用较大的学习率，在后续阶段使用较小的学习率，来达到既能增加收敛速度，又能精细调节模型使得最终效果更好的目的。

4.3 网络结构实验

我更改了网络结构，并比较了在不同网络结构下训练的损失和准确率变化情况。网络的结构的具体设置呈现在表 5 中。我设置了三个拥有不同结构的网络，其卷积层的数量依次增多，目的是探究网络的复杂性对于优化速度以及训练效果的影响。图 9 反映了训练损失和准确率的变化情况。通过观察我发现，结构较为简单的网络net1在训练开始阶段效果提升的很快，但经过一段时间的训练后会到达一个瓶颈，在瓶颈期间效果远不如更为复杂的网络net2和net3；而结构较为复杂的网络net3在初始阶段提升速度较慢，但其瓶颈期较短，且度过瓶颈期后上升的速度很快。而net2的变化则介于而这中间。

4.4 有关瓶颈期的猜想

为什么会有这种瓶颈期呢？我的猜想是这段时间是属于卷积层提升特征提取能力的阶段，而前后则是全连接层利用特征进行分类的阶段。具体过程如下：在开始时，卷积层对于图像特征的提取还处在比较初级的阶段，此时结构较复杂的网络net3由于卷积层太多，不但没有进行有效的特征提取反而阻碍了全连接层对于图像的分类；当全连接层更新完毕时，卷积层还处在特征提取的学习阶段，此时net1由于卷积层数目太少，学习的十分缓慢，对用着瓶颈期；最终卷积层学会如何提取特征后，全连接层进行进一步的更新，学会用提取的特征进行分类，对应着最后网络从瓶颈期的飞跃。

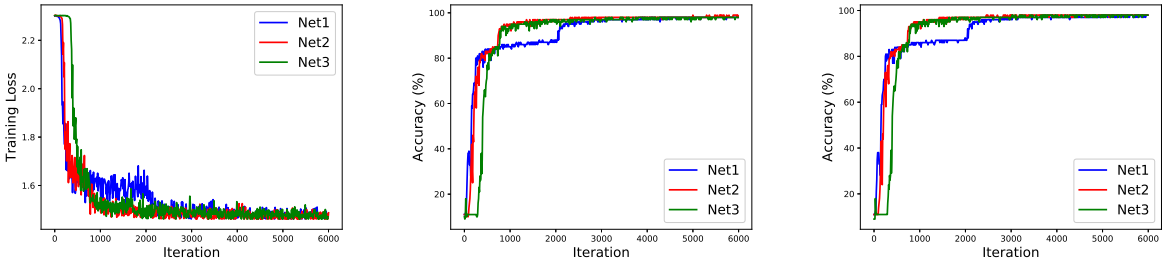
我尝试着做了实验验证：绘制出网络在刚刚到达瓶颈期、刚刚结束瓶颈期[1]、以及最终的卷积层可视化结果进行比较，具体的方法为首先得到卷积层经过Relu激活的结果（因为Relu才是直接与全连接层相连的部分），利用sigmoid归一化到[0, 1]区间，然后进行可视化。因为net1的瓶颈效果最显著，我采用net1进行实验。通过观察实验数据，我设定训练准确率达到82%为瓶颈期开始，达到95%为瓶颈期结束。如果我的假设正确，则在瓶颈期开始阶段的特征应该和瓶颈期刚结束时的特征图有很大不同，而最终的特征图应该与瓶颈期结束时的特征图无明显差别。

最后的结果如图 10 所示，可以看到，在部分通道上，瓶颈期开始阶段提取的特征和瓶颈期结束后以及训练结束后提取的特征有很大不同，瓶颈开始阶段的卷积层往往将整个数字所在区域全部激活，而瓶颈结束后以及训练结束后得到卷积层仅仅将部分区域激活，提取的是局部特征，并且二者的特征图比较接近。但在剩余的通道上，三者提取的特征有无明显差别。可见我的猜想有一定的合理性，但是否正确还有待更多实验验证。

[1]按照我的假设，此处应该为即将结束瓶颈期，但考虑到“即将结束”不好判定，以及从即将结束瓶颈期到完成飞跃结束瓶颈期用时很短，可认为模型参数几乎不变，我采用了更加容易判定的刚刚结束瓶颈期代替。

表 5: 不同的网络结构

Model	#Conv1 filters	#Conv2 filters	# Conv3 filters	#FC layers
Net1	6	-	-	3
Net2	6	16	-	3
Net3	6	16	32	3



(a) Training Loss曲线　(b) Train Accuracy曲线　(c) Test Accuracy曲线

图 9: 不同网络结构的训练曲线

4.5 可视化与逻辑推断

我采取了和作业2中相同的Grad-Cam可视化方法。我认为这种方法的主要目的是通过找到卷积层后的特征图中对某个类别分类有正面作用的区域来表现神经网络所学到的知识，并用热力图将其可视化。

4.5.1 Grad-Cam 简介

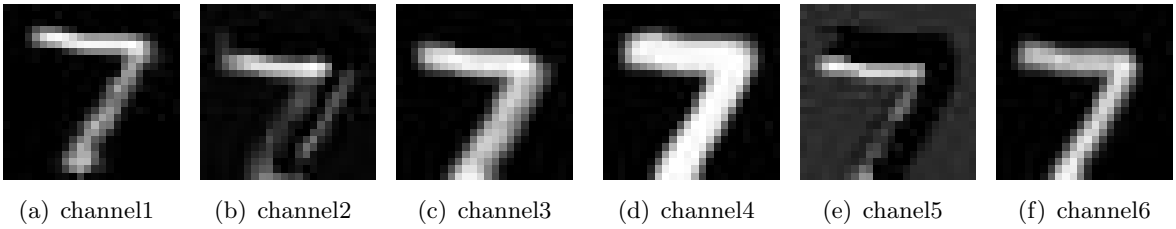
Grad-Cam是一种基于Cam改进，能够在不改变网络结构的条件下可视化对某一类别的分类有较大正面作用的特征。

首先介绍一下Cam [Zhou et al.(2015)Zhou, Khosla, Lapedriza, Oliva, and Torralba]，我的理解是最后一层卷积层的任一张输出特征图主要提取了和某一类别相关的部分特征。如果对于这个特征图取均值，则反映了输入图片在这一特征上的表现水平。再将这些特征图的平均值输入全连接层，通过全连接层反映这些特征最后对于某一类别分类的贡献。假设最后一个卷积层会输出n个特征图，我们将其记作$A_1, A_2, \cdots, A_n$，进行平均后的输出分别为$a_1, a_2, \cdots, a_n$，对于类别c的贡献可以表示为如下形式：

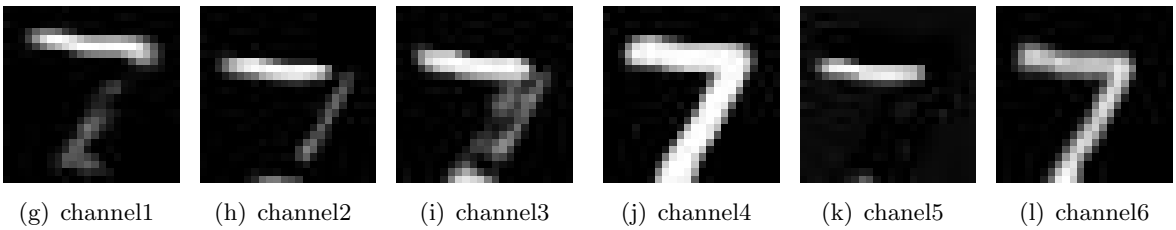
$$S^c = \sum_{i=1}^{n} w_i^c a_i.$$

其中w_i为训练得到的模型参数。$w_1, w_2, \cdots, w_n$可以认为是$A_1, A_2, \cdots, A_n$对于类别c的贡献权重，如果按照这个权重融合特征图$A_1, A_2, \cdots, A_n$，可以将对类别c有显著作用的特征可视化：

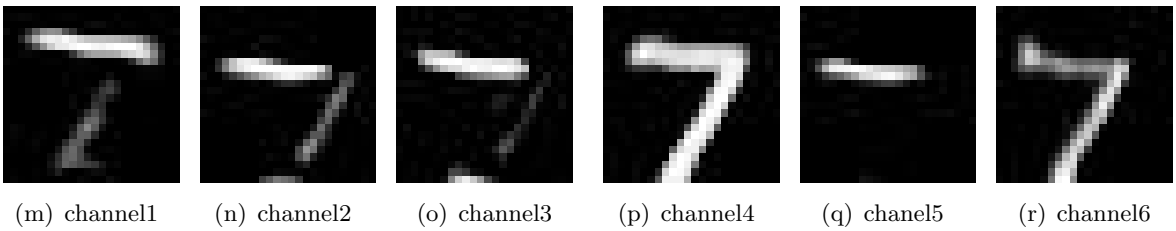
$$L^c = \sum_{i=1}^{n} w_i^c A_i.$$



(a) channel1 (b) channel2 (c) channel3 (d) channel4 (e) chanel5 (f) channel6



(g) channel1 (h) channel2 (i) channel3 (j) channel4 (k) chanel5 (l) channel6



(m) channel1 (n) channel2 (o) channel3 (p) channel4 (q) chanel5 (r) channel6

图 10: 不同阶段特征图的可视化，三排分别对应瓶颈期开始阶段，瓶颈期刚结束阶段，训练结束阶段。

但Cam方法存在必须要改变网络结构的问题，对于一个已经训练好的模型这难免会极大地增加计算时间开销。因此， [Selvaraju et al.(2016)Selvaraju, Das, Vedantam, Cogswell, Parikh, and Batra]提出了Grad-Cam算法，利用梯度的反向传播，求得了w_i而不需要改变网络结构重新训练。注意到w_i^c可以通过$\partial S_c/\partial a_i$求得，而这个偏导有可以表示为：

$$w_i^c = \frac{\partial S_c}{\partial a_i} = \frac{\frac{\partial S_c}{A_i^{j,k}}}{\frac{\partial a_i}{\partial A_i^{j,k}}} = Z\frac{\partial S_c}{\partial A_i^{j,k}}.$$

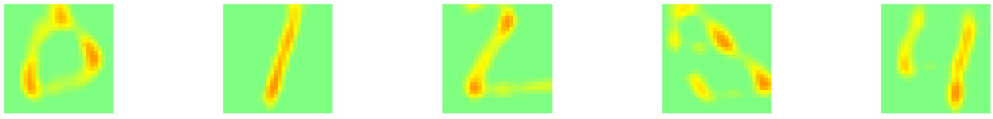
其中Z为特征图A_i的元素个数。且等式对任何j,k均成立。如果对所有像素点累加，则有

$$Zw_i^c = \sum_{j,k} w_i^c = \sum_{j,k} Z\frac{\partial S_c}{\partial A_i^{j,k}}.$$

因此，$w_i^c = \sum_{j,k}\frac{\partial S_c}{\partial A_i^{j,k}}$。加上归一化步骤，可以将$l_c$表示为如下形式：

$$L^c = \frac{1}{Z}\sum_{i=1}^{n}\sum_{j,k}\frac{\partial S_c}{\partial A_i^{j,k}}A_i.$$

即为Grad-Cam的表达形式。



(a) 0特征 (b) 1 特征 (c) 2 特征 (d) 3 特征 (e) 4 特征

(f) 5 特征 (g) 6 特征 (h) 7 特征 (i) 8 特征 (j) 9 特征

图 11: Grad-Cam可视化特征热力图

4.5.2 Grad-Cam实现与可视化

我们在模型的最后一层卷积层输出的特征图上实现了Grad-Cam可视化，得到图 11 所示的结果。其中，每张图表示特征图中对于对应数字有较大正面作用的区域，色调越暖则表示正面作用越大。可以看到，特征热力图中有较为清晰的数字轮廓，但每个数字颜色最深的区域仅仅在其中的一部分，可见神经网络是一般是通过学习到图片的局部空间特征来识别数字的。当然，对于1这种非常简单的数字，神经网络学习到了它的全部空间特征，对于整个数字都很敏感。

4.6 结论/假设及实验

1. 假设：我发现一些不同的数字其实也是有相同之处的，比如数字8去掉某些笔画就与数字5类似。那么不同的数字所包含的信息是否有所交叉？

   实验设计：训练好的模型可以看作对数字的像素分布的知识，我将用这种知识进行该假设的检验。将数字5作为目标类别，对于不同类别的输入图片，进行Grad-Cam可视化，观察其特征图是否有对数字5敏感的部分。

   实验结果：不同类别图片对于数字5的敏感部分如图 12 所示。可以看到，除去数字5本身外，1，3，4，5，8，9也有一些对于5的识别敏感的部分，这些部分大多数是相应数字和5的重叠区域。因此，我认为不同的数字所包含的信息确实有所交叉。

   实验结论：这个假设比较合理。

2. 假设：既然一个数字可能包含有其他数字的分布信息，这种信息是否是有价值的？我们是否可以通过这种信息加速神经网络的训练？

   实验设计：首先对一个教师网络进行训练，训练完成后，这个教师网络可以看作已经学习到了数字的信息和数字之间的交叉信息。接下来同时训练两个网络，一个网络以教师网络的输出作为标准，一个网络以标签作为标准，采取相同的图片顺序，bathc size，学习率，优化方法等进行训练，采取MSE损失函数，比较损失值下降和测试准确率上升的速度。注意到为公平起见，我们记录的损失值都是对于标签的损失值。

   实验结果：batch size被调整为500。两种方法的训练损失和准确率变化如图 13 所示。可以看到，基于教师网络的学习在训练的开始阶段有着更快的收敛速度，可见这种

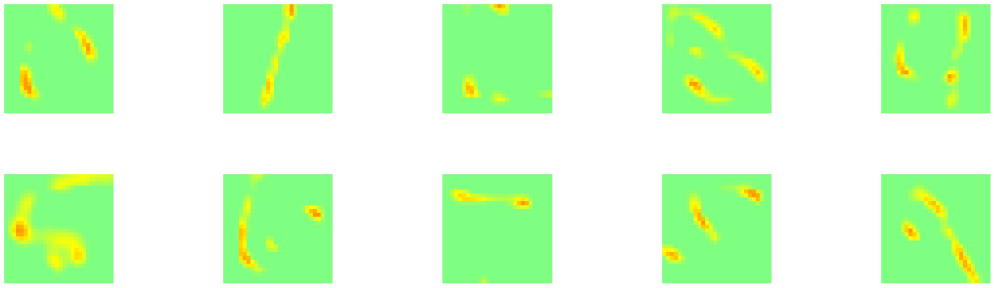


图 12: Grad-Cam：对于数字5的敏感程度可视化特征热力图

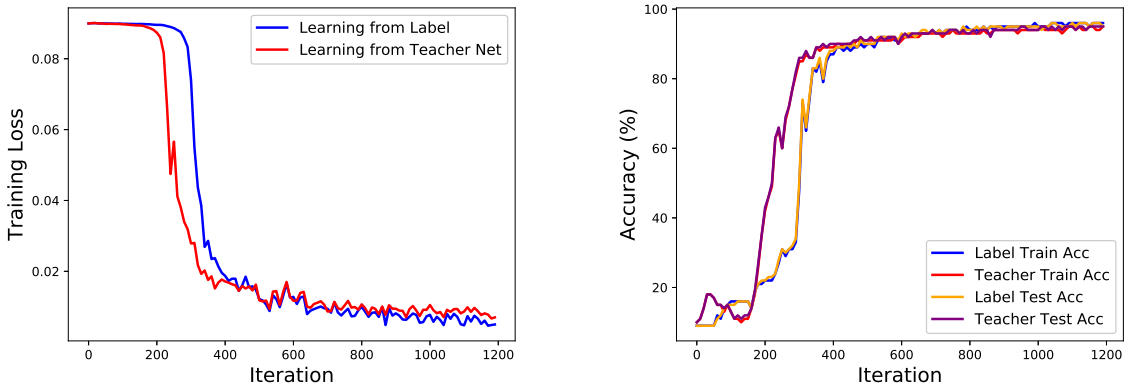


图 13: 基于标签/基于教师网络学习训练损失和准确率变化对比曲线

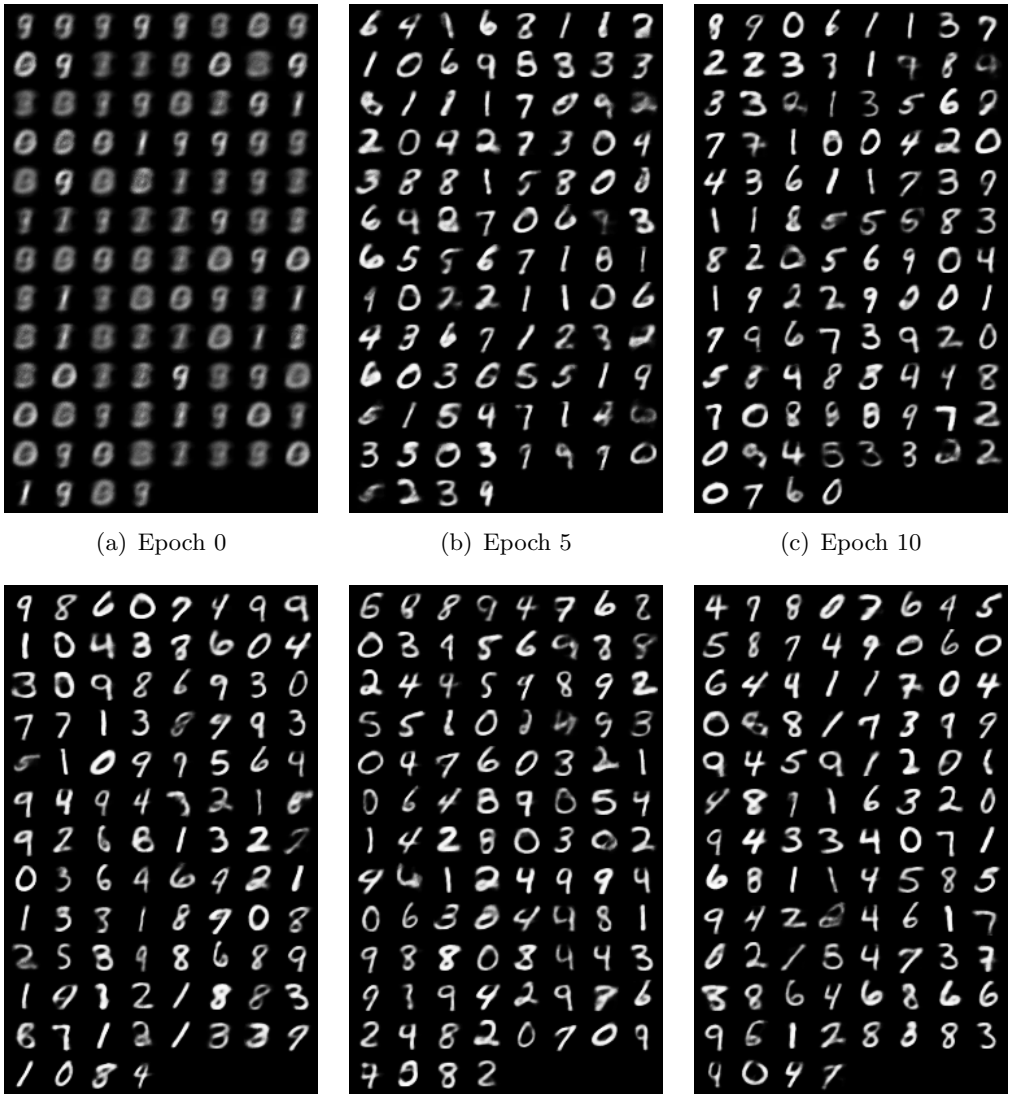
数字间的交叉信息有一定的价值。但随着训练的进行，基于标签的学习的效果最后超过了基于教师网络的学习，因为基于教师网络的学习终究只是在学习教师所拥有的知识，而基于标签的学习则是在学习“真理”。尽管教师网络所拥有的数字间的交叉信息有助于学习速度的提升，但最终达到的效果还是不如基于标签的学习。但如果将二者结合起来，可能即能兼顾收敛速度，又可达到很好的效果。准确率方面的变化和损失值的变化比较一致，且几乎不存在过拟合现象。

实际应用场景：一般来说，如果已有一个效果很好的网络，是不需要再训练一个网络的。但在某些场景中，我想这也是很有意义的。比如出于隐私保护和知识产权的原因 [European Parliament and Council of the European Union(2016)]，在某些场景下，不能共享模型的参数，但可以看到模型的输出结果。在这种场景下，可以适当的利用已有网络的输出，加快训练的过程，提升效率。

实验结论：这个假设比较合理，但是否为偶然情况还需进一步验证。因此我又补充做了三次实验，曲线的变化情况和图中都比较类似。如果以后有机会，我会做进一步的验证。

5 Generative Model: VAE

我尝试着实现了简单的VAE用于生成MNIST图像。实现的主要思路是：784维输入图



(a) Epoch 0 (b) Epoch 5 (c) Epoch 10

(d) Epoch 15 (e) Epoch20 (f) Epoch 25

图 14: 不同epoch时得到的重建图像

像经过encoder网络，得到两个10维向量，一个向量对应着编码结果（10维向量）的均值，一个向量对应着编码结果的方差。根据编码结果取样，再将取样结果输入到decoder网络，解码得到重建图像。我的实验中encoder和decoder都为全连接网络，其中encoder的神经元数量为

$$Input(784) \to 300 \to 100 \to 10.$$

其中，最后一个箭头处有两个线性模型，分别得到编码结果的均值和方差。根据均值、方差取样后，得到样本s，将s输入解码器，经过结构为

$$10 \to 100 \to 300 \to 784$$

的网络得到重建图像。

优化的目标由两部分组成，其一是最小化重建图像和原始图像的差距（我采用的损失函数是BCE），其二是最小化取样结果和标准正态分布的差距（用KL散度衡量，直接用均

值和方差计算）。将两项的损失函数相加，得到最终的损失函数。在实现时，我注意到了方差必须要为正数，而在初始阶段encoder神经网络的输出是随机的，无法保证这一点。因此，我将输出（记为x）的平方看作方差，以此为基础计算KL散度和取样。图 14 展示了训练的结果，可以清晰地看到VAE的重建效果越来越好。

6 总结

在这次的机器学习项目1中，我实现了多种机器学习方法，并用实验去验证这些方法的有效性。我的直观感受是较为简单的模型如Logestic，LDA等，虽然效果不如更为复杂的卷积神经网络，但却拥有更直观的可解释性，如Logestic得到的分类器可以直接看到每个像素点的贡献程度。较为复杂的模型则能提供更准确的预测，可解释性却远不如简单模型。我在本次实验中用一些可视化方法尝试着理解神经网络的分类逻辑，取得了一点点成果。当然，距离真正理解神经网络的内在逻辑还有很长的路要走。除此以外，因为我接触过一些关于优化的知识，所以借此机会尝试着对神经网络的优化过程进行理论解释。在Project 2中，我也在此基础上考虑了batch size的影响并给出了理论证明。

总的来说，这学期的机器学习过程让我受益匪浅，我不仅学习到了基本的机器学习方法，更对其背后深层次的统计学意义有了一定的理解。虽然这次实验还有许多不完善的地方，我也没有实现所有的方法并验证更多的结论，但还是加深了我对机器学习的理解。最后，十分感谢张老师一学期以来精彩的讲课与细致的答疑，以及助教一学期以来的帮助。预祝老师和助教工作顺利，做出更多有影响力的研究。

参考文献

[Zhou et al.(2015)Zhou, Khosla, Lapedriza, Oliva, and Torralba] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. 2015.

[Selvaraju et al.(2016)Selvaraju, Das, Vedantam, Cogswell, Parikh, and Batra] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. 2016.

[European Parliament and Council of the European Union(2016)] European Parliament and Council of the European Union. The General Data Protection Regulation (EU) 2016/679 (GDPR). `https://eur-lex.europa.eu/eli/reg/2016/679/oj`, April 2016. Took effect from May 25, 2018.