最近在刷cs231n的课程和作业，在这里分享下自己的学习过程，同时也希望能够得到大家的指点。

写在前面：
1. 这仅仅是自己的学习笔记，如果侵权，还请告知;
2. 代码是参照[lightaime的github](https://github.com/lightaime/cs231n/blob/master/assignment1/svm.ipynb)，在其基础之上做了一些修改;
3. 在我之前，已有前辈在他的[知乎](https://www.zhihu.com/people/wu-sang-91/pins/posts)上分享过类似内容;
4. 讲义是参照[杜客](https://zhuanlan.zhihu.com/p/21930884)等人对cs231n的中文翻译。

温馨提醒：
1. 这篇文档是建立在你已经知道softmax线性分类器、随机梯度下降法（SGD）的基本原理，如果在阅读本篇文档过程中有些原理不是很清楚，请移步温习下相关知识（参阅cs231n讲义）;
2. 文档的所有程序是使用python3.5实现的，如果你是python2的用户，可能要对代码稍作修改；
3. 文章频繁提到将代码保存到```.py```中，是为了方便接下来的模块导入，希望读者可以理解。

注意：softmax线性分类器大体和svm线性分类器内容相同，不过是损失函数和梯度更新是不一样的，所以我们这里只实现了这一部分（好吧，其实我太懒）
使用线性softmax分类器完成cifa10的分类工作，主要有以下几个内容：
1. 模型构建；
2. 数据处理；
3. 梯度计算和检验；
4. 训练和预测；
5. 权重可视化。

#1 softmax线性分类器构建
对该分类器的构建，主要涉及梯度和损失函数计算、训练和预测模型这两个方面，我们分别来分别对其进行实现。

##1.1 损失函数和梯度的计算
损失函数的计算的公式如下（这里是没有加入正则项的公式，正则项加入同svm线性分类器）：

$$Li = -log(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}})$$

对该式的解释，我更喜欢以概率论的角度：给定图像数据xi，以W为参数，分配给正确分类标签yi的归一化概率。
损失函数和梯度的计算也是有数值和梯度两种实现方式，代码如下：
```#python
import numpy as np
from random import shuffle

def softmax_loss_naive(W,X,y,reg):
    loss=0.0
    dW=np.zeros_like(W)
    num_classes=W.shape[1]
    num_train=X.shape[0]

    for i in range(num_train):
        scores=X[i].dot(W)
        shift_scores=scores-max(scores)
        loss_i=-shift_scores[y[i]]+np.log(sum(np.exp(shift_scores)))
        loss+=loss_i
        for j in range(num_classes):
            softmax_out=np.exp(shift_scores[j])/sum(np.exp(shift_scores))
            if j==y[i]:
                dW[:,j]+=(-1+softmax_out)*X[i]
            else:
                dW[:,j]+=softmax_out*X[i]
    loss/=num_train
    loss+=0.5*reg*np.sum(W*W)
    dW=dW/num_train+reg*W
    return loss,dW

def softmax_loss_vectorized(W,X,y,reg):
```

```
    loss=0.0
    dW=np.zeros_like(W)
    num_classes=W.shape[1]
    num_train=X.shape[0]
    scores=X.dot(W)


    shift_scores=scores-np.max(scores,axis=1).reshape(-1,1) #先转成(N,1) python广播机制

    softmax_output=np.exp(shift_scores)/np.sum(np.exp(shift_scores),axis=1).reshape((-1,1))

    loss=-np.sum(np.log(softmax_output[range(num_train),list(y)])) #softmax_output[range(num_train),list(y)]计算的是正确分类y_i的损失
    loss/=num_train
    loss+=0.5*reg*np.sum(W*W)

    dS=softmax_output.copy()
    dS[range(num_train),list(y)]+=-1
    dW=(X.T).dot(dS)
    dW=dW/num_train+reg*W
    return loss,dW
```

这份代码被保存到了```softmax.py```中。

##1.2 分类器的构建

构建的分类器模型除了损失函数和梯度外，还需要构建训练和预测模型，其中训练模型是采用随机梯度下降法来进行梯度更新的。

代码如下：

```#python
import numpy as np
import sys
sys.path.append("..")
from softmax.~~softmax~~ import *


class LinearClassifier:
    def __init__(self):
        self.W = None

    def train(self, X, y, learning_rate=1e-3, reg=1e-5, num_iters=100,
              batch_size=200, verbose=False):

        num_train,dim = X.shape
        num_classes = np.max(y) + 1  # assume y takes values 0...K-1 where K is number of classes
        if self.W is None:
            # lazily initialize W
            self.W = 0.001 * np.random.randn(dim, num_classes)

        # Run stochastic gradient descent to optimize W
        loss_history = []
        for it in range(num_iters):
            X_batch = None
            y_batch = None

            batch_idx = np.random.choice(num_train, batch_size,replace=True)
            X_batch = X[batch_idx]
            y_batch = y[batch_idx]

            # evaluate loss and gradient
            loss, grad = self.loss(X_batch, y_batch, reg)

            loss_history.append(loss)

            self.W += -1 * learning_rate * grad
```

```
            if verbose and it % 100 == 0:
                print('iteration %d / %d: loss %f' % (it, num_iters, loss))

        return loss_history

    def predict(self, X):

        y_pred = np.zeros(X.shape[1])
        scores = X.dot(self.W)
        y_pred = np.argmax(scores, axis = 1)
        return y_pred

    def loss(self, X_batch, y_batch, reg):
        pass

class Softmax(LinearClassifier):
  """ A subclass that uses the Softmax + Cross-entropy loss function """

  def loss(self, X_batch, y_batch, reg):
    return softmax_loss_vectorized(self.W, X_batch, y_batch, reg)
```

剩下的内容就是和*svm*线性分类器一毛一样了，各位理解就好！