Ai Lab

Name:张展翔
Student Number: PB20111669

part_1

贝叶斯网络手写数字识别

实验原理

数字识别贝叶斯网络主要采用了像素值作为特征值来进行预测图片所对应的数字

在 `fit` 函数中，需要完成对先验概率和条件概率的计算,其中， `labels_prior` 为先验概率，

```
    def fit(self, pixels, labels):
        n_samples = len(labels)
        # Calculate prior probability
        labels_counts = np.bincount(labels, minlength=self.n_labels)
        self.labels_prior = labels_counts / n_samples

        # Calculate conditional probability
        for pixel in range(self.n_pixels):
            for value in range(self.n_values):
                for label in range(self.n_labels):
                    pixel_value_mask = (pixels[:, pixel] == value)
                    label_pixel_mask = (labels == label)
                    count = np.sum(pixel_value_mask & label_pixel_mask)
                    self.pixels_cond_label[pixel, value, label] = count / labels_counts[label]
```

然后通过遍历每一个数字每个像素的值来计算对应的条件概率

在 `predict` 函数中，首先获取待处理图片的像素值，然后根据学习的概率来预测最高可能性的值。

```
    def predict(self, pixels):
        n_samples = len(pixels)
        labels_pred = np.zeros(n_samples, dtype=np.int8)

        for i in range(n_samples):
            probabilities = np.zeros(self.n_labels)
            for label in range(self.n_labels):
                label_prob = self.labels_prior[label]
                for pixel in range(self.n_pixels):
                    pixel_value = pixels[i, pixel]
                    pixel_prob = self.pixels_cond_label[pixel, pixel_value, label]
                    label_prob *= pixel_prob
                probabilities[label] = label_prob

            labels_pred[i] = np.argmax(probabilities)

        return labels_pred
```

实验结果

准确率是84.38%

```
C:\ProgramData\Anaconda3\python.exe F:\Desktop\exp2\src_exp2_ai2023sp_ustc\part_1\src\Bayesian-network.py
test score: 0.8438
```

利用K-means实现图片压缩

实验原理

利用KMeans算法将图像的每一个像素（R，G，B）作为数据点并对所有点进行聚类。完成聚类之 后，将所有数据点的值替换成其聚类中心的值，这样仅需保留几个聚类中心的数据点值以及其他 点的类别索引，从而达到压缩图片的目的。

在`assign_points`函数中，我们需要计算每个点和中心点的距离并把它们分配给最近的中心点，因此，我们可以使用如下方法进行计算

```
        for i in range(n_samples):
            distances = np.linalg.norm(points[i] - centers, axis=1)
            labels[i] = np.argmin(distances)
```

在`update_centers`函数中，我们需要根据新的数据点来更新中心点

```
        for k in range(self.k):
            cluster_points = points[labels == k]
            if len(cluster_points) > 0:
                centers[k] = np.mean(cluster_points, axis=0)
```

在`fit`函数中，通过迭代更新中心直到收敛或者达到最大迭代次数来执行，可以用np.allclose来检查收敛性

```
 centers = self.initialize_centers(points)
        for _ in range(self.max_iter):
            labels = self.assign_points(centers, points)
            new_centers = self.update_centers(centers, labels, points)
            if np.allclose(centers, new_centers):
                break
            ceters = new_centers
```

在`compress`函数中，我们需要将像素值替换为对应的附近聚类中心值。可以用如下方式实现

```
points = img.reshape((-1, img.shape[-1]))
        centers = self.fit(points)
        labels = self.assign_points(centers, points)
        compressed_points = centers[labels.astype(int)]
        compressed_img = compressed_points.reshape(img.shape)
```

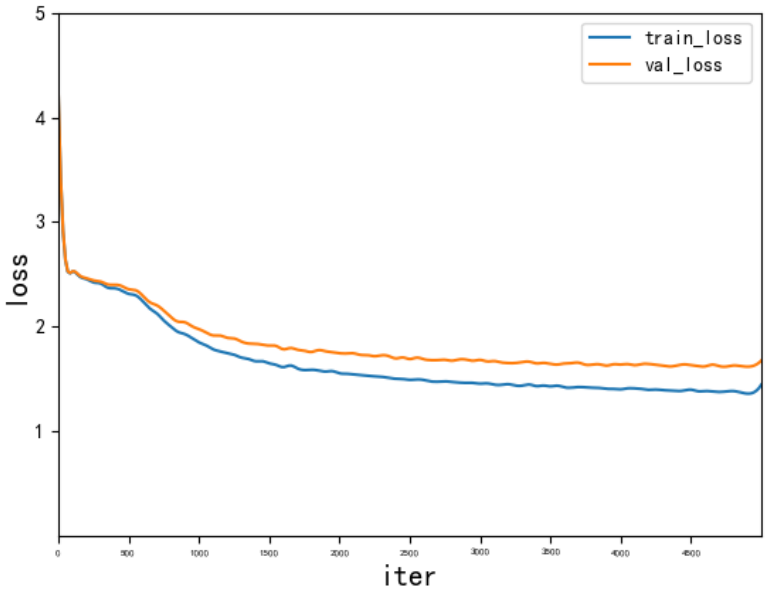
实验结果

Compressed Image

Compressed Image

part_2

误差曲线如下所示：



输入：

To be or not to be: that is the

结果(为什么会这样呜呜呜)：

To be or not to be: that is the shamest of the stand.

Citizens:
I would you the say the same of your souls and the seast
That the shall be the shall be the seast of the state.

Second Citizen:
I will the shall be the shall be the shall be the shall
That the that the to the stand of the world,
And the to the to the stand the father of the
That the that the the stand of the world to the stand
The to the that the to the to the to the to thee.

The would the the shame of the world to the stand the seat of the seat
That the the t