华中科技大学

课程报告

题目：图像分类：基与 KNN，SVN 或卷积神经网络等技术，进行 MNIST 图像数据的分类

院（系）：计算机学院
课程名称：人工智能导论（专选课）
学　　号：**I201920024**
姓　　名：木林
任课老师：魏老师
报告日期：**2020.05.26**

目录

滴要

本文存在的原因是调查和调查一些通常用于 AI 的 AI 计算，从而使 ML 模型在此点上围绕着为何和如何以 MNIST 数据库中的手写数字图片顺序来编写这些 ML 计算。

关键词：KNN（最近邻）计算，SVM（支持向量机），图像分类，MNIST 数据，欧几里得分离，交叉验证，零件工程。

1.绪论

如今，创新的发展速度，尤其是 PC 领域的发展速度令人震惊，甚至可以说是一个真正的进步阶段。如果一个人有时学习未知的方言并且他们对外来词没有最模糊的想法，则个人可以简单地使用手机上的摄像头在一秒钟内将非本地语言翻译成他们的母语。在另一种情况下，他们只需要打开电话并在解密应用程序中输入字符或单词，应用程序就会自然地识别出他们的手写，解释和表达。到处都是垃圾。如果不收集，将使环境状况更加令人担忧。为了有效地组织起来，我们必须集中精力于通常受到影响和污染的地区。在那里，图像分类被证明是有用的。最终模型之一是淘宝网上购物阶段。客户可以将他们想要的产品照片传输到淘宝。淘宝网将识别出所提供的图片以表征所提供的照片中包含的产品。届时，淘宝将提供产品摘要，以协调客户照片和许多其他有趣的事情。给定的模型具有真正的酷炫和超自然创新的所有特征，并且是所谓的相似图片分组之一。

MNIST 数据库（美国国家标准技术研究院的改进数据库）是一个大型的手写数字数据库，通常用于准备不同的图像处理框架。它具有不同的分类器，包括 KNN（K 最近邻）计算，SVM（支持向量机），CNN（卷积神经网络），线性分类器，Boost 树桩，非线性分类器，随机森林，DNN（深层神经网络），RMDL（随机多模型深度学习）(Wikipedia, 2020)。在本文中，我们将快速体验两种算法以及它们在 MNIST 数据库中的实现方式。

2. 机器学习算法

2.1. KNN（K-最近的邻居）算法

K 最近邻计算是分类和回归执行的最简单方法。为了简单起见，我们有两类数据正方形和三角形，如图 2.1 所示，它由 6 个正方形和 5 个三角形组成。假设我们需要获得有关新数据的答案，即绿色圆圈，该圆圈可能会将新数据分类为正方形乘员或三角形乘员。那就是 K-NN 计算出现的时候。 K 是最接近新信息的数据的估计。假设 k = 3，最接近的正方形= 1，三角形= 3，则投票的比例很大，因此新信息在三角形类别中占有一席之地。但是，当 K = 5 时，最接近的正方形= 3，而三角形= 2，因此新信息在类别正方形中具有位置。

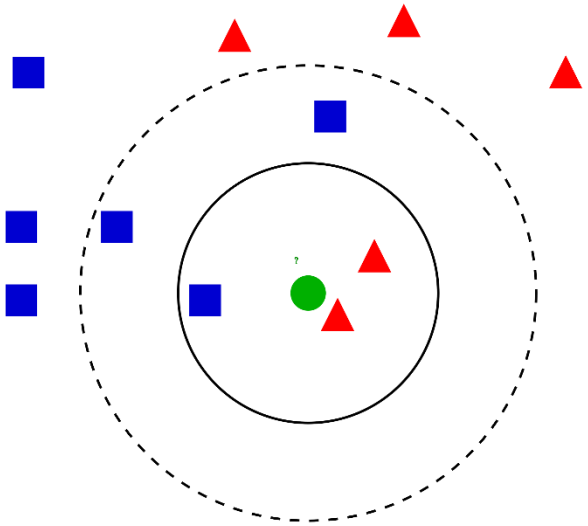


图 2.1

KNN 如何运作？按照 KNN 的含义，该测试通过确定自身与数据集中其他数据之间的距离来获得最近的邻居。因此，火车信息与测试信息之间的距离越短，将测试信息分类为已知数据类别的可能性就越大。要计算两个焦点之间的距离（火车数据与测试数据之间的距离），有很多方法，包括欧几里得，马哈拉诺比斯，曼哈顿，明可夫斯基，切比雪夫，余弦，相关，汉明，雅卡德，欧几里得和斯皮尔曼距离。使用独特的距离计算技术，KNN 的准确性可能会有所不同 (V. B. Surya Prasatha, 2019)。最近的邻居 k 值也会影响结果的准确性。因此，我们将如何选择 k 刺激来使预测更加准确。由于具有显着值 k 的最佳决策取决于给定的数据，因此不可能选择具有显着值 k 的最佳决策以获得更准确的结果，但是选择正确的 k 自尊有一些基本标准：

-较大的 k 估计值可以减少高噪声对表示的影响，但可以使类别之间

的界限更加清晰。由于结果是基于投票的较大部分，因此对 k 的较大估计可能会导致更准确的预测。在任何情况下，这都不是 100%来确保较大的 k 给出越来越准确的结果，这取决于所引用的数据。

-选择 k 作为奇数，因为避免平局

KNN 是基于时间的学习或懒惰学习器，因为它不会从准备信息中区分能力，但会“记住”准备数据集。使用 KNN 的优缺点：

优点：

-计算非常基础且易于实现

-没有令人信服的理由来组装模型

缺点：

-考虑到信息量度的扩展，由于它会计算测试信息与训练集中所有信息之间的距离，因此计算速度会变慢

2.2. SVM（支持向量机）

支持向量机 SVM 是用于 AI 领域中的分类，回归和异常发现的学习技术。 SVM 的目标是将新信息订购到已实现的数据类别中。 支持向量机将数据视为 n 维向量。 此时，SVM 形成一个（n-1）维超平面，将数据隔离为多个类。 分离的超平面和任何称为边缘的类的最接近数据之间的距离。 例如，在图 2.2 中，给出了支持向量机的可视化

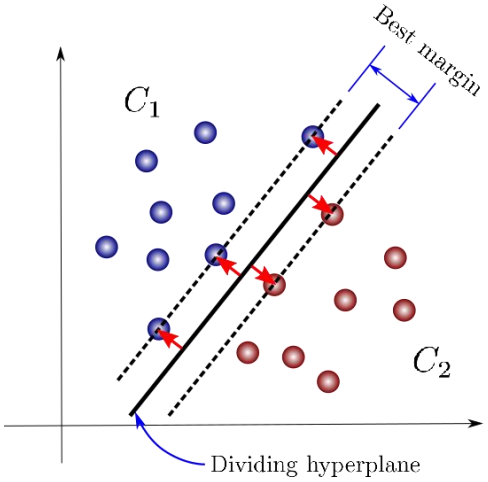


图 2.2

产生最大余量的超平面称为最大余量分类器。 总而言之，最大余量分类器是为数据顺序提供最佳结果的最佳分离超平面，因此余量越大，数据排列就越好。图 2.2 中的数据称为线性可分离数据。如果是图 2.3 中的数据，则称为非线性可分离数据。 通过利用假设为核函数的方程，将图 2.3 中二维空间中的数据映射到三维空间数据中图 2.4，以便更容易开发隔离超平面。 因此，使用核函数是，当无法通过线性分离将数据隔离为类时，它将数据映射到更高维的空间，找到边界超平面，然后进行分类。 核函数具有一些函数，包括线性核，多项式核，径向基函数核（RBF），高斯核，拉普拉斯 RBF 核，双曲正切核和 S 形核。

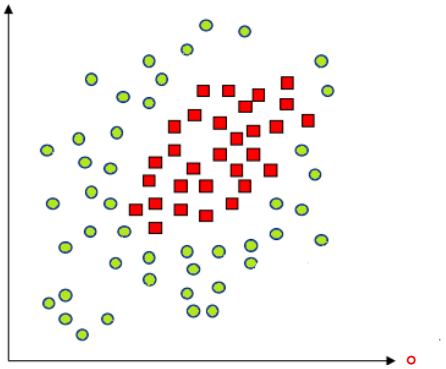


图 2.3

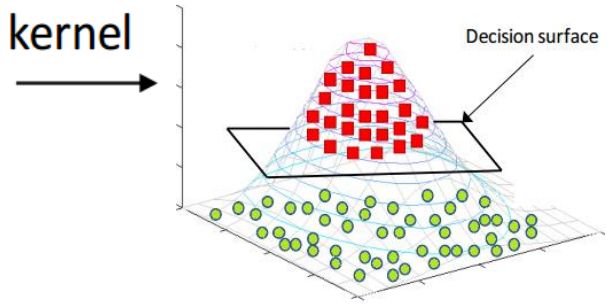


图 2.4

3. MNIST 修改后的美国国家标准技术研究院数据库

引用了该术语，并在演示中给出了简短定义。 MNIST 代表修改后的美国国家标准技术研究院数据库，如上一篇文章中所引用的，是一个庞大的手写数字数据库，通常用于准备不同的图片处理框架。MNIST 是对 NIST（国家标准与创新组织）准备数据集的测试的转载。NIST 的第一批高对比度图片不适用于 AI。通过处理像素质量的焦点，并将图片解密以将该点定位在 28x28 场的焦点处，这些图片被打包为 28x28 像素图片，每个图片都包含一个单独的灰度。 MNIST 包含 60,000 张从 0 到 9 的所有数字的手写数字图片。每张图片的标签标记编号都与图片中的编号相对应。例如，在图 3.1 中，第一个图片的名称为 4，第二个图片的名称为 1，第三个图片的名称为 8，依此类推。 MNIST 数据库包含 6 万张准备图片和 10000 张要测试的图片。训练集的一半和测试集的一半是从 NIST 的训练集获得的，另一半是从 NIST 的测试集获得的。

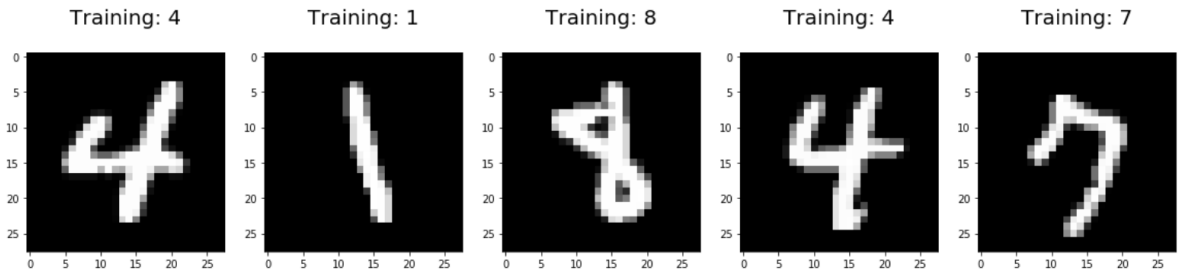


图 3.1 数据及其标签

表征手工数据集的 MNIST 的计算有很多，包括线性分类器，非线性分类器，K 最近邻，增强树桩，SVM，神经网络和卷积网络。 独特的计算具有不同的精确度。 以下是 MNIST 数据库合法站点的概述，该概述专门介绍了用于表征 MNIST 及其表示形式的每个分组策略(Yann LeCun, 1998)，如图 3.2 所示：

CLASSIFIER	PREPROCESSING	TEST ERROR RATE (%)	Reference
Linear Classifiers			
linear classifier (1-layer NN)	none	12.0	LeCun et al. 1998
linear classifier (1-layer NN)	deskewing	8.4	LeCun et al. 1998
pairwise linear classifier	deskewing	7.6	LeCun et al. 1998
K-Nearest Neighbors			
K-nearest-neighbors, Euclidean (L2)	none	5.0	LeCun et al. 1998
K-nearest-neighbors, Euclidean (L2)	none	3.09	Kenneth Wilder, U. Chicago
K-nearest-neighbors, L3	none	2.83	Kenneth Wilder, U. Chicago
K-nearest-neighbors, Euclidean (L2)	deskewing	2.4	LeCun et al. 1998
K-nearest-neighbors, Euclidean (L2)	deskewing, noise removal, blurring	1.80	Kenneth Wilder, U. Chicago
K-nearest-neighbors, L3	deskewing, noise removal, blurring	1.73	Kenneth Wilder, U. Chicago
K-nearest-neighbors, L3	deskewing, noise removal, blurring, 1 pixel shift	1.33	Kenneth Wilder, U. Chicago
K-nearest-neighbors, L3	deskewing, noise removal, blurring, 2 pixel shift	1.22	Kenneth Wilder, U. Chicago
K-NN with non-linear deformation (IDM)	shiftable edges	0.54	Keysers et al. IEEE PAMI 2007
K-NN with non-linear deformation (P2DHMDM)	shiftable edges	0.52	Keysers et al. IEEE PAMI 2007
K-NN, Tangent Distance	subsampling to 16x16 pixels	1.1	LeCun et al. 1998
K-NN, shape context matching	shape context feature extraction	0.63	Belongie et al. IEEE PAMI 2002
Boosted Stumps			
boosted stumps	none	7.7	Kegl et al., ICML 2009
products of boosted stumps (3 terms)	none	1.26	Kegl et al., ICML 2009
boosted trees (17 leaves)	none	1.53	Kegl et al., ICML 2009
stumps on Haar features	Haar features	1.02	Kegl et al., ICML 2009
product of stumps on Haar f.	Haar features	0.87	Kegl et al., ICML 2009
Non-Linear Classifiers			
40 PCA + quadratic classifier	none	3.3	LeCun et al. 1998
1000 RBF + linear classifier	none	3.6	LeCun et al. 1998
SVMs			
SVM, Gaussian Kernel	none	1.4	
SVM deg 4 polynomial	deskewing	1.1	LeCun et al. 1998
Reduced Set SVM deg 5 polynomial	deskewing	1.0	LeCun et al. 1998
Virtual SVM deg-9 poly [distortions]	none	0.8	LeCun et al. 1998
Virtual SVM, deg-9 poly, 1-pixel jittered	none	0.68	DeCoste and Schelkopf, MLJ 2002
Virtual SVM, deg-9 poly, 1-pixel jittered	deskewing	0.68	DeCoste and Schelkopf, MLJ 2002
Virtual SVM, deg-9 poly, 2-pixel jittered	deskewing	0.56	DeCoste and Schelkopf, MLJ 2002
Neural Nets			
2-layer NN, 300 hidden units, mean square error	none	4.7	LeCun et al. 1998
2-layer NN, 300 HU, MSE, [distortions]	none	3.6	LeCun et al. 1998
2-layer NN, 300 HU	deskewing	1.6	LeCun et al. 1998
2-layer NN, 1000 hidden units	none	4.5	LeCun et al. 1998
2-layer NN, 1000 HU, [distortions]	none	3.8	LeCun et al. 1998
3-layer NN, 300+100 hidden units	none	3.05	LeCun et al. 1998
3-layer NN, 300+100 HU [distortions]	none	2.5	LeCun et al. 1998
3-layer NN, 500+150 hidden units	none	2.95	LeCun et al. 1998
3-layer NN, 500+150 HU [distortions]	none	2.45	LeCun et al. 1998
3-layer NN, 500+300 HU, softmax, cross entropy, weight decay	none	1.53	Hinton, unpublished, 2005
2-layer NN, 800 HU, Cross-Entropy Loss	none	1.6	Simard et al., ICDAR 2003
2-layer NN, 800 HU, cross-entropy [affine distortions]	none	1.1	Simard et al., ICDAR 2003
2-layer NN, 800 HU, MSE [elastic distortions]	none	0.9	Simard et al., ICDAR 2003
2-layer NN, 800 HU, cross-entropy [elastic distortions]	none	0.7	Simard et al., ICDAR 2003
NN, 784-500-500-2000-30 + nearest neighbor, RBM + NCA training [no distortions]	none	1.0	Salakhutdinov and Hinton, AI-Stats 2007
6-layer NN 784-2500-2000-1500-1000-500-10 (on GPU) [elastic distortions]	none	0.35	Ciresan et al. Neural Computation 10, 2010 and arXiv 1003.0358, 2010
committee of 25 NN 784-800-10 [elastic distortions]	width normalization, deslanting	0.39	Meier et al. ICDAR 2011
deep convex net, unsup pre-training [no distortions]	none	0.83	Deng et al. Interspeech 2010
Convolutional nets			
Convolutional net LeNet-1	subsampling to 16x16 pixels	1.7	LeCun et al. 1998
Convolutional net LeNet-4	none	1.1	LeCun et al. 1998
Convolutional net LeNet-4 with K-NN instead of last layer	none	1.1	LeCun et al. 1998
Convolutional net LeNet-4 with local learning instead of last layer	none	1.1	LeCun et al. 1998
Convolutional net LeNet-5, [no distortions]	none	0.95	LeCun et al. 1998
Convolutional net LeNet-5, [huge distortions]	none	0.85	LeCun et al. 1998
Convolutional net LeNet-5, [distortions]	none	0.8	LeCun et al. 1998
Convolutional net Boosted LeNet-4, [distortions]	none	0.7	LeCun et al. 1998
Trainable feature extractor + SVMs [no distortions]	none	0.83	Lauer et al., Pattern Recognition 40:6, 2007
Trainable feature extractor + SVMs [elastic distortions]	none	0.56	Lauer et al., Pattern Recognition 40:6, 2007
Trainable feature extractor + SVMs [affine distortions]	none	0.54	Lauer et al., Pattern Recognition 40:6, 2007
unsupervised sparse features + SVM, [no distortions]	none	0.59	Labusch et al., IEEE TNN 2008
Convolutional net, cross-entropy [affine distortions]	none	0.6	Simard et al., ICDAR 2003
Convolutional net, cross-entropy [elastic distortions]	none	0.4	Simard et al., ICDAR 2003
large conv. net, random features [no distortions]	none	0.89	Ranzato et al., CVPR 2007
large conv. net, unsup features [no distortions]	none	0.62	Ranzato et al., CVPR 2007
large conv. net, unsup pretraining [no distortions]	none	0.60	Ranzato et al., NIPS 2006
large conv. net, unsup pretraining [elastic distortions]	none	0.39	Ranzato et al., NIPS 2006
large conv. net, unsup pretraining [no distortions]	none	0.53	Jarrett et al., ICCV 2009
large/deep conv. net, 1-20-40-60-80-100-120-120-10 [elastic distortions]	none	0.35	Ciresan et al. IJCAI 2011
committee of 7 conv. net, 1-20-P-40-P-150-10 [elastic distortions]	width normalization	0.27 +-0.02	Ciresan et al. ICDAR 2011
committee of 35 conv. net, 1-20-P-40-P-150-10 [elastic distortions]	width normalization	0.23	Ciresan et al. CVPR 2012

图 3.2

3.1. 使用 KNN 算法对 MNIST 数据库的手写数字进行分类

尽管 KNN 计算是这两种选择的计算中最困难的一个，但对于这样的基本订单计算，KNN 的排列精度并不差。 像所引用的一样，有不同的方法来确定焦点和数据之间的分离，但是在此部分中，将使用欧几里得距离。 我们将利用 Anaconda 飞行员来协助跳入 Jupyter Notebook，并且将使用 python 编程语言进行编码。

首先让我们导入必要的库，加载 MNIST 数据集，然后将其打印出来。 科学工具

```
In [11]: from sklearn.neighbors import KNeighborsClassifier
         import numpy as np
         import matplotlib.pyplot as plt
         from tensorflow.keras.datasets import mnist

In [12]: (train_data, train_labels), (test_data, test_labels) = mnist.load_data()

In [13]: print('train data: ' + str(train_data.shape))
         print('train lables: ' + str(train_labels.shape))
         print('test data:  '  + str(test_data.shape))
         print('test labels:  '  + str(test_labels.shape))
```

输出：

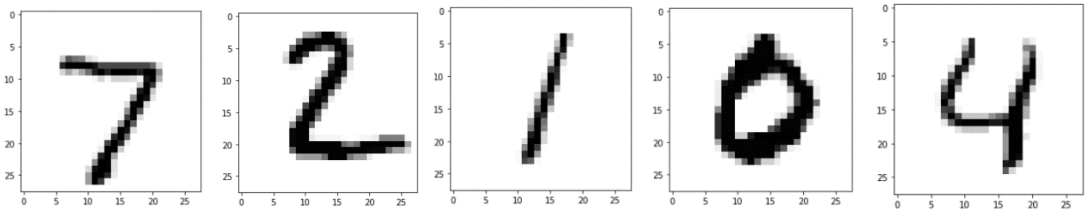
```
MNIST Dataset Shape:
train data: (60000, 28, 28)
train lables: (60000,)
test data:  (10000, 28, 28)
test labels:  (10000,)
```

显示一些 MNIST 手写图像：

```
In [4]: def displayImage(i):
            plt.imshow(test_data[i], cmap='Greys')
            plt.show()

In [137]: for i in range (0, 5):
              displayImage(i)
```

输出：



及其标签：

```
In [138]: for b in range (0, 5):
              print(test_labels[b])

7
2
1
0
4
```

通过为相应的火车数据分配火车标签来训练我们的 KNN 模型。 为了缩短运行时间，仅使用火车的前 10000 个火车数据：

In [15]:
```
test_data_range=[]
for test in range (0, 10000):
    test_datanp=np.array(test_data[test])
    test_datan=test_datanp.reshape(1, -1)
    test_data_range.append(test_datan)
```

In [9]:
```
train_data_range=[]
for train in train_data:
    train=train.flatten()
    train_data_range.append(train)
train_data_range=np.array(train_data_range, dtype=np.float32)
```

In [10]:
```
model = KNeighborsClassifier(1)
model.fit(train_data_range[:10000], train_labels[:10000])
```

Out[10]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

我们将对测试数据的前 5 个在 k = 1 进行预测，并查看其是否与上述结果匹配，并显示我们的 KNN 模型的准确性：

In [13]:
```
for r in range (0, 5):
    prediction=int(model.predict(test_data_range[r]))
    print("test data prediction")
    print(prediction)
success=0
for a in range (0, 10000):
    if int(model.predict(test_data_range[a]))==test_labels[a]:
        success+=1
accuracy=(success/10000)*100
print("accuracy=%.2f%%" % (accuracy))
```

```
test data prediction
7
test data prediction
2
test data prediction
1
test data prediction
0
test data prediction
9
accuracy=94.63%
```

现在我们看到我们的预测结果与 In [137]和 In [138]中的结果匹配。对 10000 个训练集的 10000 个测试数据集进行测试大约需要 2 分钟，并且给出近 95%的准确结果。 请记住，精确度和时间可能会有所不同，具体取决于训练集和测试集的数量。

现在，让我们看看 k 值如何影响模型的准确性：

In [148]:
```
for kvals in range(1, 16, 2):
    true=0
    KNN = KNeighborsClassifier(kvals)
    KNN.fit(train_data[:10000], train_labels[:10000])
    for b in range (0, 10000):
        if int(KNN.predict(test_data_range[b]))==test_labels[b]:
            true+=1
    print("k=%d, accuracy=%.2f%%" % (kvals, true*100/10000))
```

```
k=1, accuracy=94.63%
k=3, accuracy=94.63%
k=5, accuracy=94.42%
k=7, accuracy=94.43%
k=9, accuracy=94.08%
k=11, accuracy=93.95%
k=13, accuracy=93.78%
k=15, accuracy=93.48%
```

我们可以看到 k = 1 和 k = 3 提供了最佳结果。

3.2. 使用 SVM 对 MNIST 数据库的手写数字进行分类

首先让我们导入必要的库，加载 MNIST 数据集，然后将其打印出来。 科学工具

```
In [11]: from sklearn.neighbors import KNeighborsClassifier
         import numpy as np
         import matplotlib.pyplot as plt
         from tensorflow.keras.datasets import mnist

In [12]: (train_data, train_labels), (test_data, test_labels) = mnist.load_data()

In [13]: print('train data: ' + str(train_data.shape))
         print('train lables: ' + str(train_labels.shape))
         print('test data:  '  + str(test_data.shape))
         print('test labels:  '  + str(test_labels.shape))
```

输出：

```
MNIST Dataset Shape:
train data: (60000, 28, 28)
train lables: (60000,)
test data:  (10000, 28, 28)
test labels:  (10000,)
```

使用 FIT 功能 10000 个列车数据和 10000 个列车标签以缩短运行时间，然后，首先测试测试数据集中的 10000 个数据。 我们可以看到精度为 95.73，比上一节中的 KNN 模型高约 1%。 还请记住，svm.SVC 中的参数也会影响预测准确性：

```
In [6]: train_data_range=[]
        for train in train_data:
            train=train.flatten()
            train_data_range.append(train)
        train_data_range=np.array(train_data_range, dtype=np.float32)

In [7]: test_data_range=[]
        for test in test_data:
            test=test.reshape(1, -1)
            test_data_range.append(test)

In [9]: svm_model = svm.SVC(gamma=0.05 , C=5, kernel='poly')
        svm_model.fit(train_data_range[:10000], train_labels[:10000])

Out[9]: SVC(C=5, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
            decision_function_shape='ovr', degree=3, gamma=0.05, kernel='poly',
            max_iter=-1, probability=False, random_state=None, shrinking=True,
            tol=0.001, verbose=False)

In [11]: true=0
         for b in range (0, 10000):
             if int(svm_model.predict(test_data_range[b]))==test_labels[b]:
                 true+=1
         accuracy=true/10000*100
         print(accuracy)

         95.73
```

预测测试数据集中的前 10 个数据：

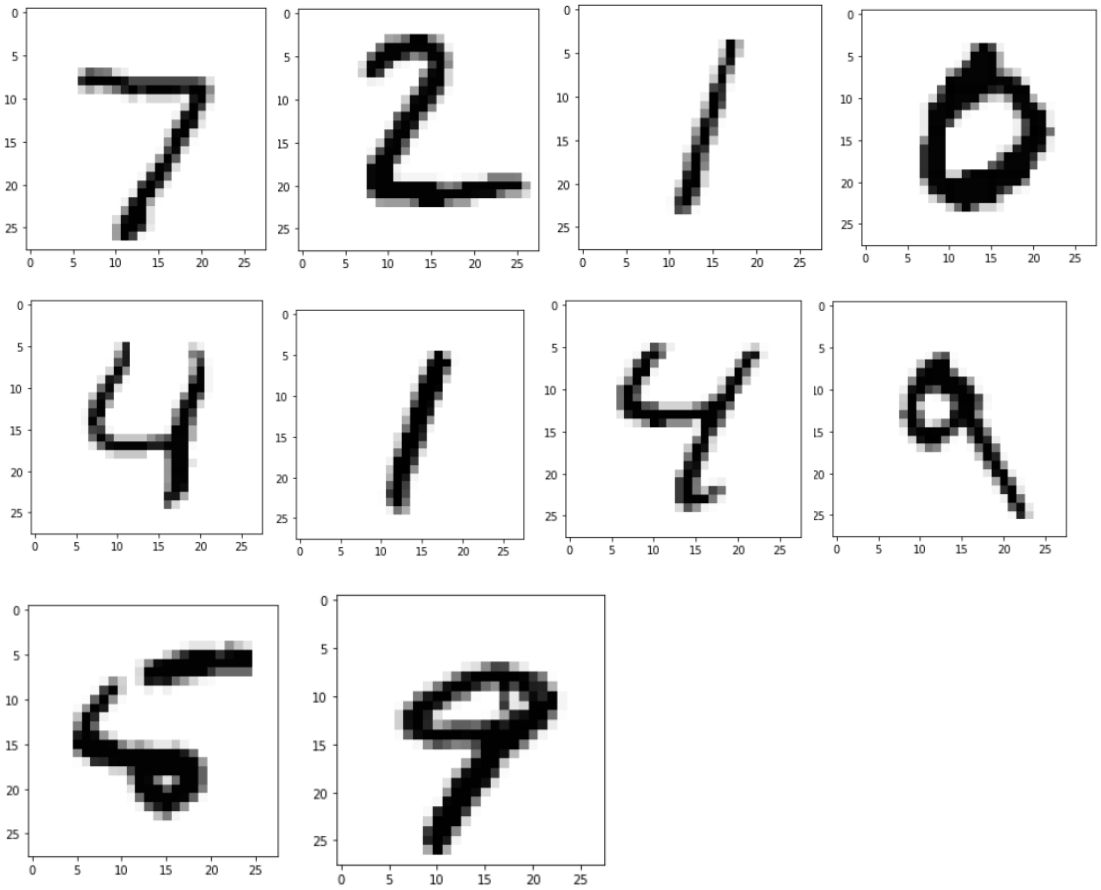
```
In [15]: for a in range (0, 10):
             prediction=svm_model.predict(test_data_range[a])
             print(prediction)

         [7]
         [2]
         [1]
         [0]
         [4]
         [1]
         [4]
         [9]
         [6]
         [9]
```

显示测试的前 10 个数据，现在我们可以看到数据与上面的结果相匹配：

```
In [4]: def displayImage(i):
            plt.imshow(test_data[i], cmap='Greys')
            plt.show()

In [16]: for a in range (0, 10):
             displayImage(a)
```



4. 总结与展望

4.1。总结

在过去的两周的深度学习研究中，我对这个领域的兴趣比以往任何时候都大。总而言之，尽管有很多方法可以解释和实现图像分类，但它仍然是具有挑战性的问题之一。 由于其在各个领域中的应用，近年来受到了广泛的关注。 尽管在此领域做出了巨大努力，但图像分类系统在完全实现地球上所有已知条件方面还远非理想。 本文简要回顾了机器学习领域，实现和应用中的几种算法。

4.2。展望

在过去的几年中，我经常梦想有一天我可以自己建立一个AI，它可以像真正的人类一样理解我的语言和情感，因此我可以与它进行交流并与朋友和伴侣一起生活。 因此，在过去的近两周的机器学习算法研究中，尽管该研究与图像校准无关，但与我的兴趣无关，但我也获得了另一种认识，即该研究确实在推动着我的雄心。 活的。 有时候，对于真正智能的机器计算机而言，数据分类很难进行分类，有时甚至是人类都会发生。在未来的研究中，将围绕以下方面进行工作：

-如何提高每种算法的预测准确性以进行更好的分类

-像提到的那样实现我的AI野心，在以后的工作中，我将研究如何构建这样的AI

参考文献

References

1.Carrasco, O. C. (2019, July 8). *Support Vector Machines for Classification*. Retrieved from towardsdatascience: https://towardsdatascience.com/support-vector-machines-for-classification-fc7c1565e3

2.Harrison, O. (2018, Sep 11). *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. Retrieved from towardsdatascience: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

3.pyimagesearch.guru. (N/A, N/A N/A). *k-Nearest Neighbor classification*. Retrieved from pyimagesearch.guru: https://gurus.pyimagesearch.com/lesson-sample-k-nearest-neighbor-classification/#

4.scikit-learn.org. (N\A, N\A N\A). *1.4. Support Vector Machines*. Retrieved from scikit-learn.org: https://scikit-learn.org/stable/modules/svm.html

5.scikit-learn.org. (N\A, N\A N\A). *sklearn.neighbors.KNeighborsClassifier*. Retrieved from scikit-learn.org: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

6.Sharma, S. (2019, Nov 29). *Kernel Trick in SVM*. Retrieved from medium: https://medium.com/analytics-vidhya/how-to-classify-non-linear-data-to-linear-data-bb2df1a6b781

7.V. B. Surya Prasatha, H. A. (2019, Sep 29). *1708.04321.pdf*. Retrieved from arxiv.org: https://arxiv.org/pdf/1708.04321.pdf

8.Wikipedia. (2020, May 28). *K-nearest neighbors algorithm*. Retrieved from Wikpedia.org: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

9.Wikipedia. (2020, Jun 11). *MNIST database*. Retrieved from Wikipedia.org: https://en.wikipedia.org/wiki/MNIST_database

10.Wikipedia. (2020, May 18). *Support vector machine*. Retrieved from Wikipedia.org: https://en.wikipedia.org/wiki/Support_vector_machine

11.Yann LeCun, C. C. (1998, Nov). *THE MNIST DATABASEof handwritten digits*. Retrieved from yann.lecun: http://yann.lecun.com/exdb/mnist/