

Trabalho Final de Machine Learning: Reconhecimento Interativo de Dígitos

Aluno : Evandro Lucas Figueiredo Teixeira

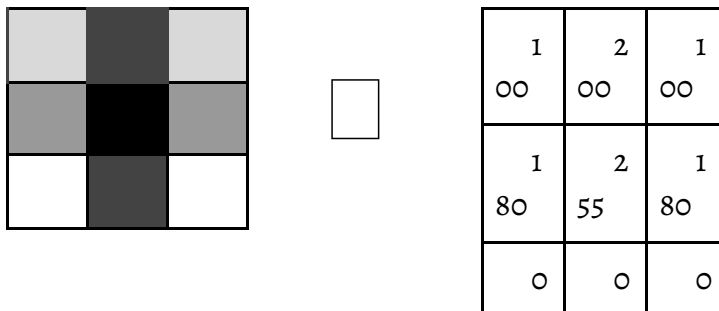
Matrícula : 2016006379

Introdução e Objetivos

O objetivo deste projeto é desenvolver uma aplicação que possibilite ao usuário desenhar um dígito na tela e verificar, em tempo real, qual número um modelo treinado irá palpar. A ideia deste trabalho é não só realizar experimentos com a MNIST Database, mas permitir a interação com o modelo treinado.

Dados e treinamento do modelo

Foi utilizada a base de dados MNIST, correspondente à mais de 60.000 entradas, cada entrada definida como uma matriz quadrada de 28 pixels de altura/largura. Cada pixel pode receber um valor entre 0 e 255, onde 0 corresponde a um pixel completamente branco e 255 a um pixel completamente preto. Segue um exemplo abaixo, utilizando uma grade 5x5:



As entradas do MNIST seguem a mesma lógica, porém com uma grade maior, de 28 x 28 pixels. Cada entrada é representada como uma linha de uma tabela, composta por :

- Uma coluna “label” indicando qual o dígito a ser representado;
- 784 colunas indicando a cor de cada pixel. As colunas são nomeadas indicando os índices da tabela, ou seja, “1x2” representa a cor do pixel situado na primeira linha e na segunda coluna da grade.

A base de dados do MNIST pode ser encontrada em seu formato original neste endereço:

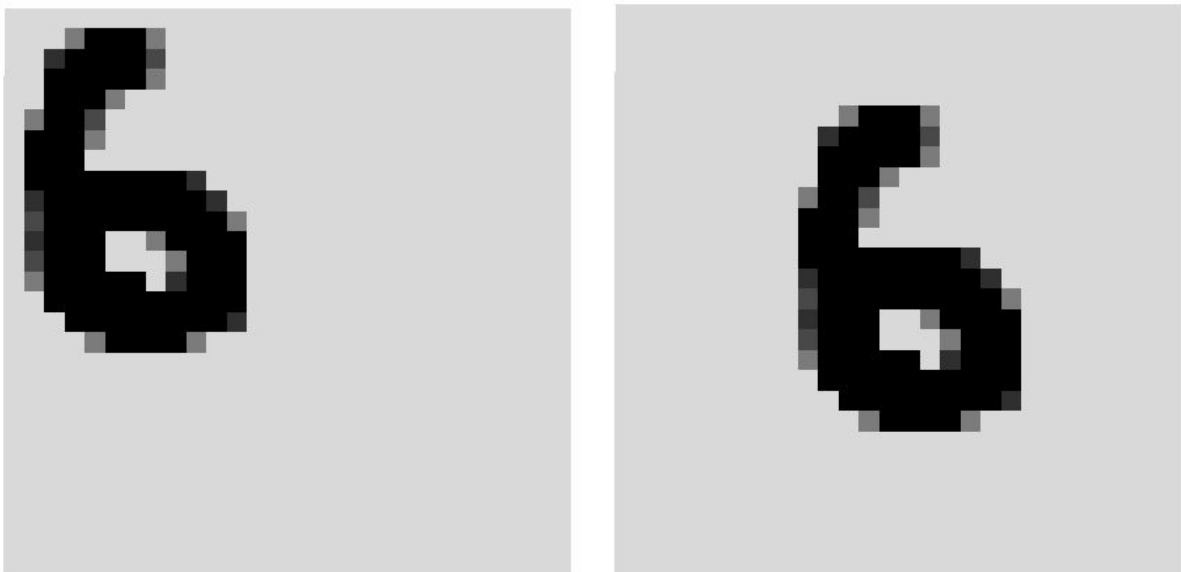
- <http://yann.lecun.com/exdb/mnist/>

Para este trabalho, foi utilizada a versão já processada destas imagens, convertendo a grade de pixels para uma linha de uma tabela, como descrito acima. Tanto a base de dados como o algoritmo para a transformação das mesmas se encontram nos links:

- <https://www.kaggle.com/oddrational/mnist-in-csv>
- <https://pjreddie.com/projects/mnist-in-csv/>

Apesar de já possuir os dados tratados no momento do treino (graças ao dataset) é necessário replicar o processo de tratamento das imagens para que o input não seja distorcido. Por exemplo, ao enviar a grade de pixels para a predição, ela deve centralizar o dígito inserido utilizando o centro de massa de cada pixel não-branco.

No que diz respeito à interface, o usuário não precisa se preocupar com a centralização pois ela ocorre de forma implícita logo antes da predição. Mesmo assim, para fins didáticos, é possível visualizar este alinhamento ao se pressionar a tecla `SHIFT` logo após desenhar um dígito:



Persistência e predições em tempo-real

A base de dados do MNIST é bem grande, com ,mais de 60.000 linhas, cada uma contendo 785 colunas. O treino de um modelo robusto de machine learning (como o Gradient Tree Boosting) pode durar dezenas de minutos para ser processado. Certamente seria mais simples baixar esta tabela utilizando um notebook e se aproveitar da persistência de dados para treinar o modelo apenas uma vez e executar predições em células separadas. Porém, como estamos desenvolvendo uma aplicação gráfica em tempo real, não podemos utilizar o notebook. Com isto em mente, esta aplicação utiliza-se da persistência, oferecida pelo módulo nativo `joblib` .

O processo de pré-execução da aplicação envolve três processos principais:

- Executar o script `fit.py` , que irá treinar algum modelo;
- O estado do modelo será salvo em um arquivo do tipo `joblib`;
- Ao se executar o script `main.py` , este estado será carregado para o novo processo.

A aplicação demora apenas de 1 à 5 segundos para ser inicializada.

Interface gráfica

Evidentemente, como se trata de uma aplicação interativa, foi necessário se implementar uma interface gráfica. Para este trabalho, foi utilizado a implementação mais comum do OpenGL em Python, PyOpenGL, e um motor gráfico para o OpenGL, que no caso foi o PyGame.

De forma simplificada, o OpenGL foi utilizado para mapear a posição do ponteiro do mouse à um pixel específico do canvas da aplicação, preenchendo esse pixel (e os pixels ao redor dele) com valores entre 0 e 255, de forma análoga ao dado fornecido do MNIST. A grade da aplicação também possui 28 x 28 quadrados. Esta matriz, então, é transformada em uma lista e é fornecida ao modelo treinado para se obter a predição.

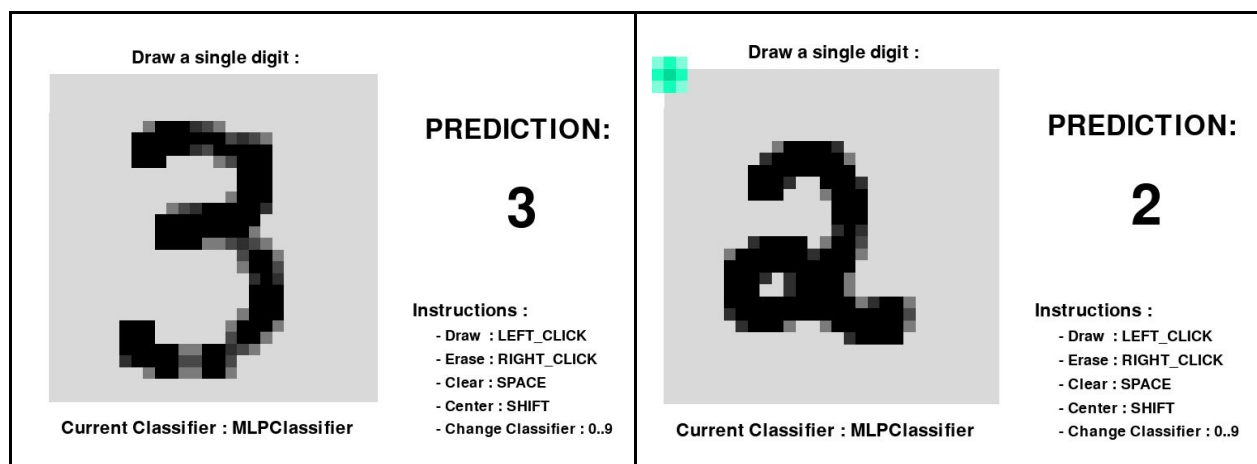
Classificadores

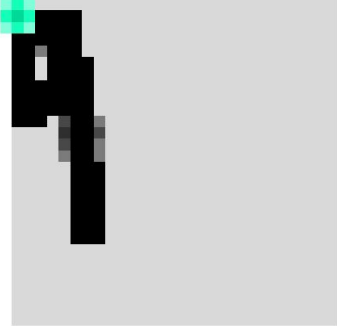
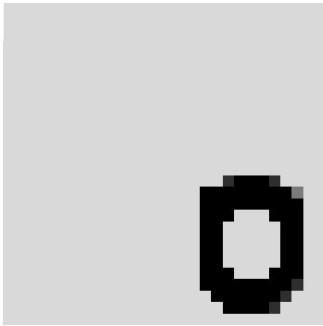
Foram utilizados 10 classificadores disponibilizados pelo sklearn, são eles:

- GaussianNB,
- Linear SVM,
- MLPClassifier,
- RBF SVM,
- DecisionTreeClassifier,
- GradientBoostingClassifier,
- AdaBoostClassifier,
- Perceptron,
- PassiveAggressiveClassifier,
- NearestCentroid,
- RandomForestClassifier

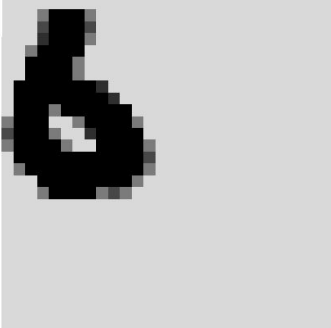
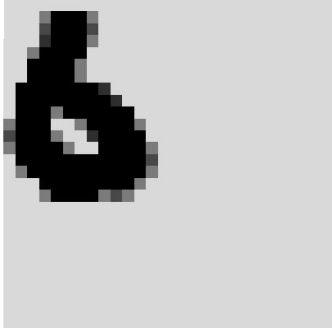

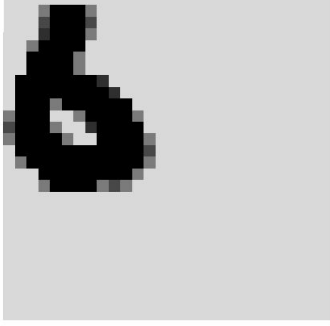
Cada um desses classificadores terá seu estado salvo no diretório `pre`, e podem ser utilizados a qualquer momento durante a execução do aplicativo através das teclas `0...9`.

Resultados (MLPClassifier)



<p>Draw a single digit :</p>  <p>PREDICTION:</p> <p>9</p> <p>Instructions :</p> <ul style="list-style-type: none"> - Draw : LEFT_CLICK - Erase : RIGHT_CLICK - Clear : SPACE - Center : SHIFT - Change Classifier : 0..9 <p>Current Classifier : MLPClassifier</p>	<p>Draw a single digit :</p>  <p>PREDICTION:</p> <p>7</p> <p>Instructions :</p> <ul style="list-style-type: none"> - Draw : LEFT_CLICK - Erase : RIGHT_CLICK - Clear : SPACE - Center : SHIFT - Change Classifier : 0..9 <p>Current Classifier : MLPClassifier</p>
--	---

Resultados (Vários Classificadores)

<p>Draw a single digit :</p>  <p>PREDICTION:</p> <p>9</p> <p>Instructions :</p> <ul style="list-style-type: none"> - Draw : LEFT_CLICK - Erase : RIGHT_CLICK - Clear : SPACE - Center : SHIFT - Change Classifier : 0..9 <p>Current Classifier : GaussianNB</p>	<p>Draw a single digit :</p>  <p>PREDICTION:</p> <p>6</p> <p>Instructions :</p> <ul style="list-style-type: none"> - Draw : LEFT_CLICK - Erase : RIGHT_CLICK - Clear : SPACE - Center : SHIFT - Change Classifier : 0..9 <p>Current Classifier : SVC</p>
<p>Draw a single digit :</p>  <p>PREDICTION:</p> <p>4</p> <p>Instructions :</p> <ul style="list-style-type: none"> - Draw : LEFT_CLICK - Erase : RIGHT_CLICK - Clear : SPACE - Center : SHIFT - Change Classifier : 0..9 <p>Current Classifier : MLPClassifier</p>	<p>Draw a single digit :</p>  <p>PREDICTION:</p> <p>3</p> <p>Instructions :</p> <ul style="list-style-type: none"> - Draw : LEFT_CLICK - Erase : RIGHT_CLICK - Clear : SPACE - Center : SHIFT - Change Classifier : 0..9 <p>Current Classifier : AdaBoostClassifier</p>

Passo-à-passo de execução

1- Antes de executar qualquer script, devemos importar os módulos abaixo com o pip. Alguns dos módulos abaixo são dependências de outros, então o download das dependências não deve demorar muito.

```
pip install PyOpenGL
pip install pygame
pip install numpy
pip install scikit-learn
pip install pandas
pip install matplotlib
pip install statistics
pip install joblib
```

2- Descompacte o arquivo `TP_FINAL.zip` em um diretório de sua escolha.

3 - (Opcional) - Verifique se o arquivo `model.joblib` se encontra no diretório `per` . Ele corresponde ao estado do modelo já treinado. Para treinar o modelo, execute o arquivo `fit.py` e aguarde até o fim de sua execução. Ele irá sobrescrever o arquivo `per/model.joblib` .

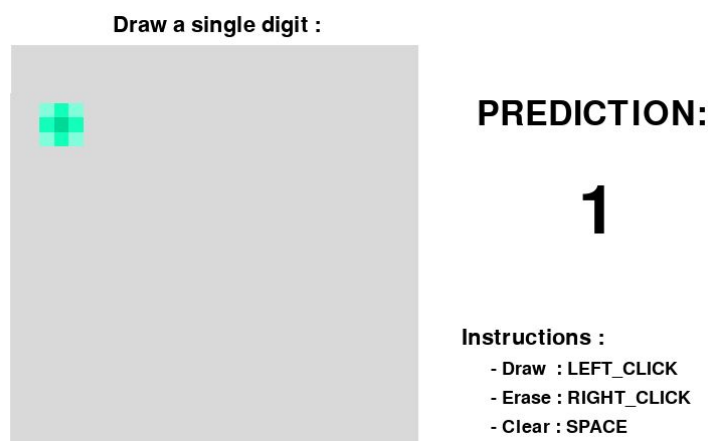
```
python3 fit.py
```

Possuir uma placa de vídeo em sua máquina irá acelerar bastante o processo, nesse caso ele deve durar alguns minutos para treinar o modelo com 50.000 entradas descritas no arquivo `mnist_train.csv` .

5- Execute a aplicação:

```
python3 main.py
```

Aguarde alguns segundos para que ela se inicialize. Você verá a tela abaixo com as instruções de uso da aplicação.



Instruções:

- Para desenhar, basta usar segurar o botão de clique do mouse e arrastar o ponteiro sobre o canvas acinzentado.
- Para apagar algum pedaço específico do desenho, usa-se o botão direito do mouse.
- Para limpar todo o canvas, basta apertar a barra de espaço.
- Para centralizar a imagem, aperte a tecla SHIFT
- Para alterar o classificador, basta utilizar as teclas numéricas de 0 à 9. O nome do classificador é exibido logo abaixo do canvas.
- Note que nem todos os classificadores possuem o mesmo desempenho ou acurácia. Alguns classificadores (como o KNN) performaram tão mal que tive de retirá-los da aplicação.

Considerações Finais

Creio que consegui aplicar conceitos vistos na matéria de uma maneira útil e interdisciplinar, principalmente no que diz respeito à modelagem de dados de entrada e entendimento do problema. Os primeiros testes que fiz me demonstraram a importância da normalização dos dados e das peculiaridades de cada algoritmo. Também foi uma ótima oportunidade para interagir com um classificador em tempo real e observar a diferença entre cada um deles.

Futuramente, pretendo adicionar mais funcionalidades à aplicação, como permitir a configuração de parâmetros dos modelos e classificar não só dígitos mas caracteres. Também pretendo aumentar a resolução do canvas e adicionar captura de pressão na entrada do mouse, para simular a escrita em uma mesa gráfica, por exemplo.

Referências

- <http://yann.lecun.com/exdb/mnist/>
- <https://www.kaggle.com/oddrational/mnist-in-csv>
- <https://pjreddie.com/projects/mnist-in-csv/>
- <https://stackoverflow.com/questions/59178717/is-k-nearest-neighbours-regression-inherently-slow>
- <https://stackoverflow.com/questions/41844311/list-of-all-classification-algorithms>

Repositório GitHub do código: <https://github.com/EvandroLucas/DigitDetector>