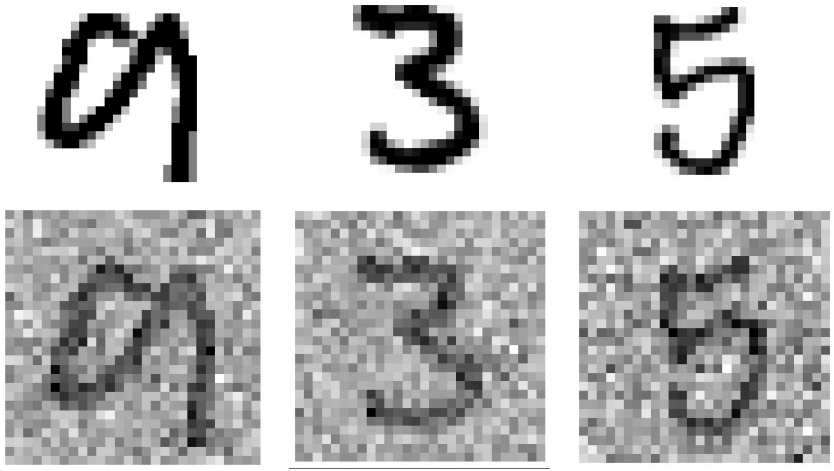
Inteligência Computacional
Projeto 1: Classificadores

João Ferreira, Ryan Oliveira e Valdinei Conceição

16 de Abril de 2024

Parte I - MNIST

- 70000 imagens de dígitos escritos a mão (0 a 9);
- Cada uma sendo 28x28 pixels (784 features);
- Cada feature varia de 0 (branco) a 255 (preto).



70% Treino [0:49000]
10% Validação [49000:56000]
20% Teste [56000:70000]

PSNR 10dB

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX^2}{MSE}\right)$$

$$\frac{PSNR}{10} = \log_{10}\left(\frac{MAX^2}{MSE}\right)$$

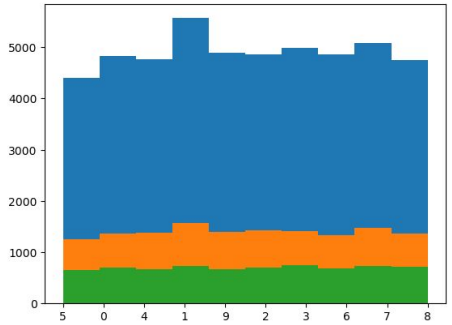
$$10^{\frac{PSNR}{10}} = \frac{MAX^2}{MSE}$$

$$MSE = \frac{MAX^2}{10^{\frac{PSNR}{10}}}$$

The value of MAX is 255 and we want a PSNR of 10dB, so the equation is:

$$MSE = \frac{255^2}{10^{\frac{10}{10}}} = 6502.5$$

```
def gen_10db_psnr():
    gaussian_noise = np.random.normal(0, np.sqrt(6502.5), X.shape[1])
    return gaussian_noise
```



Parte I - Classificadores

- Naive Bayes:

Acurácia	Teste sem ruído	Teste com ruído
Treinamento sem ruído	0.82	0.61
Treinamento com ruído	0.79	0.84

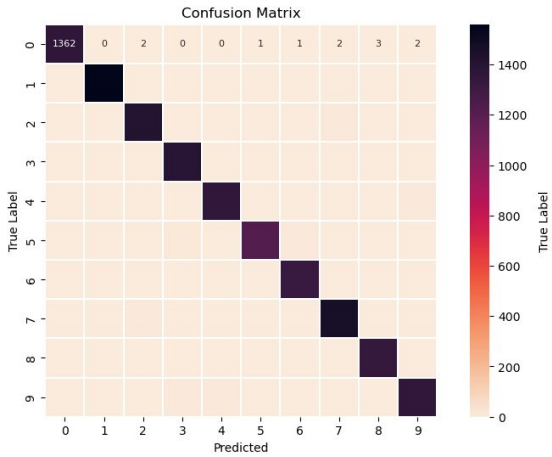
- Decision Tree:

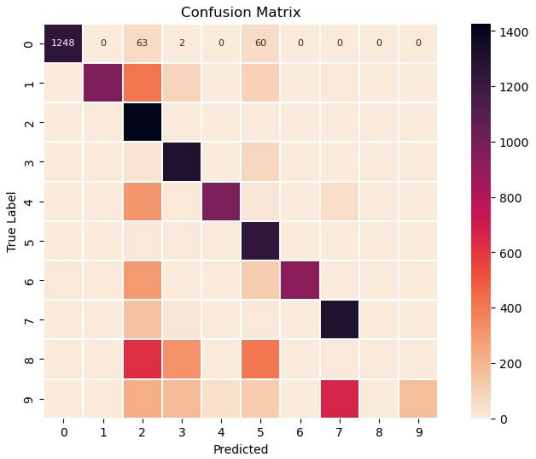
Acurácia	Teste sem ruído	Teste com ruído
Treinamento sem ruído	0.88	0.33
Treinamento com ruído	0.81	0.66

- SVM:

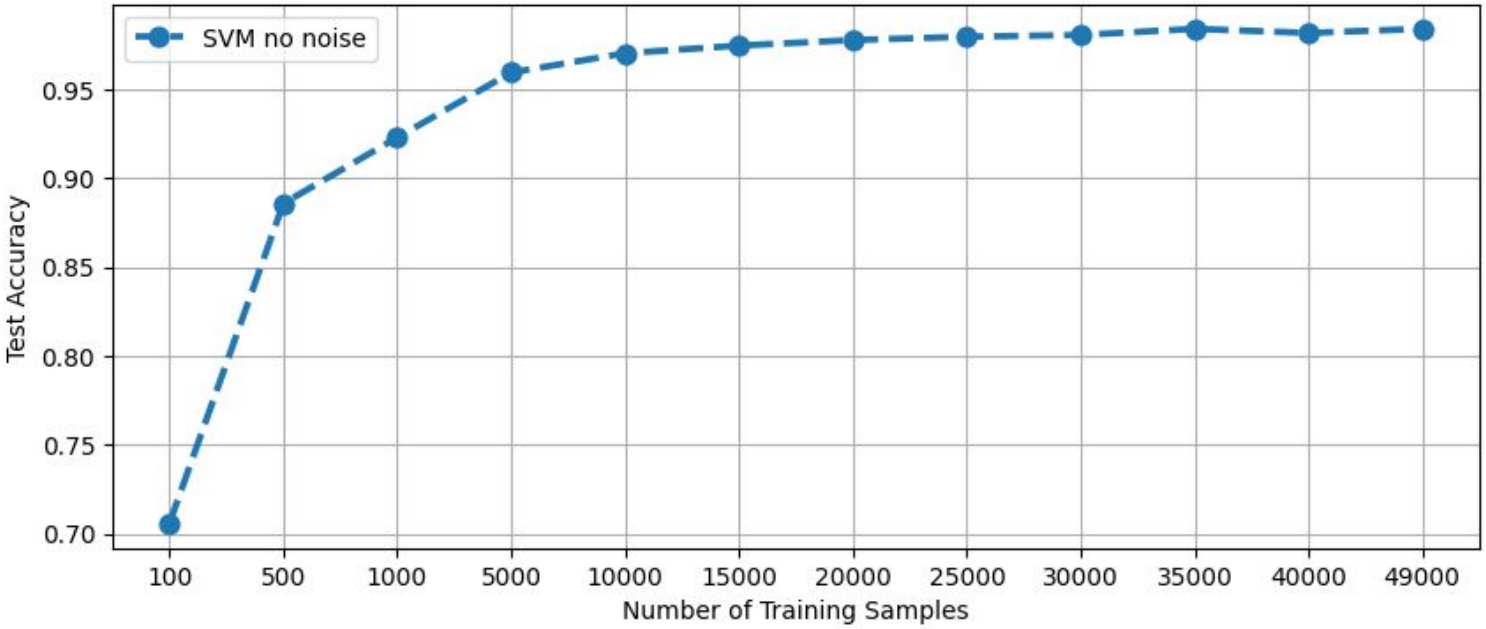
Acurácia	Teste sem ruído	Teste com ruído
Treinamento sem ruído	0.98	0.68
Treinamento com ruído	0.96	0.95

- Melhores hiperparâmetros SVM (treinamento com ruído e sem): C=10, gamma='scale', kernel='rbf'





Parte I - Sample Complexity



Parte 2 - Dataset

	clientid	income	age	loan	default
count	2000.000000	2000.000000	1997.000000	2000.000000	2000.000000
mean	1000.500000	45331.600018	40.807559	4444.369695	0.141500
std	577.494589	14326.327119	13.624469	3045.410024	0.348624
min	1.000000	20014.489470	-52.423280	1.377630	0.000000
25%	500.750000	32796.459717	28.990415	1939.708847	0.000000
50%	1000.500000	45789.117313	41.317159	3974.719419	0.000000
75%	1500.250000	57791.281668	52.587040	6432.410625	0.000000
max	2000.000000	69995.685578	63.971796	13766.051239	1.000000

Para a parte 2 do projeto escolhemos o dataset `credit_data.csv`[1] que contém colunas que simulam dados de agências de crédito. Os atributos são as colunas `income`, `age` ,`loan` e a classe é a coluna `default`, classificando quem paga os empréstimos com `1` e quem não paga com `0`.

[X] (https://www.kaggle.com/datasets/laotse/credit-risk-dataset [adaptado])

Decision Tree Vs SVM

Obtivemos resultados positivos acima dos 90% de acurácia nos 3 modelos testados, ficando mais ou menos empatado entre os modelos Decision Tree e SVM, então abaixo iremos analisar as estatísticas obtidas nos dois modelos e decidir qual o melhor:

Métrica	Árvore de Decisão	SVM
Acurácia	98%	99%
Precisão (Classe 0)	99%	99%
Precisão (Classe 1)	92%	99%
Revocação (Classe 0)	99%	100%
Revocação (Classe 1)	95%	91%
F1-Score (Classe 0)	99%	99%
F1-Score (Classe 1)	94%	95%

Parte 2 - Hiperparâmetros

```python
# Cria uma instância do classificador SVM
svm_credit = svm.SVC()

# Define os hiperparâmetros para testar
param_grid = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'rbf'],
    'gamma': ['scale', 'auto']
}

# Cria o objeto GridSearchCV
grid_search = GridSearchCV(estimator=svm_credit, param_grid=param_grid, cv=5)

# Treina o modelo com o Grid Search
grid_search.fit(X_credit_train, y_credit_train)

# Obtem os melhores hiperparâmetros encontrados
best_params = grid_search.best_params_
```

```
Melhores hiperparâmetros encontrados:
{'C': 100, 'gamma': 'auto', 'kernel': 'rbf'
```

[X] https://github.com/jvictorferreira3301/Inteligencia_Computacional/tree/main

Parte 2 - Matriz de Confusão e Sample Complexity