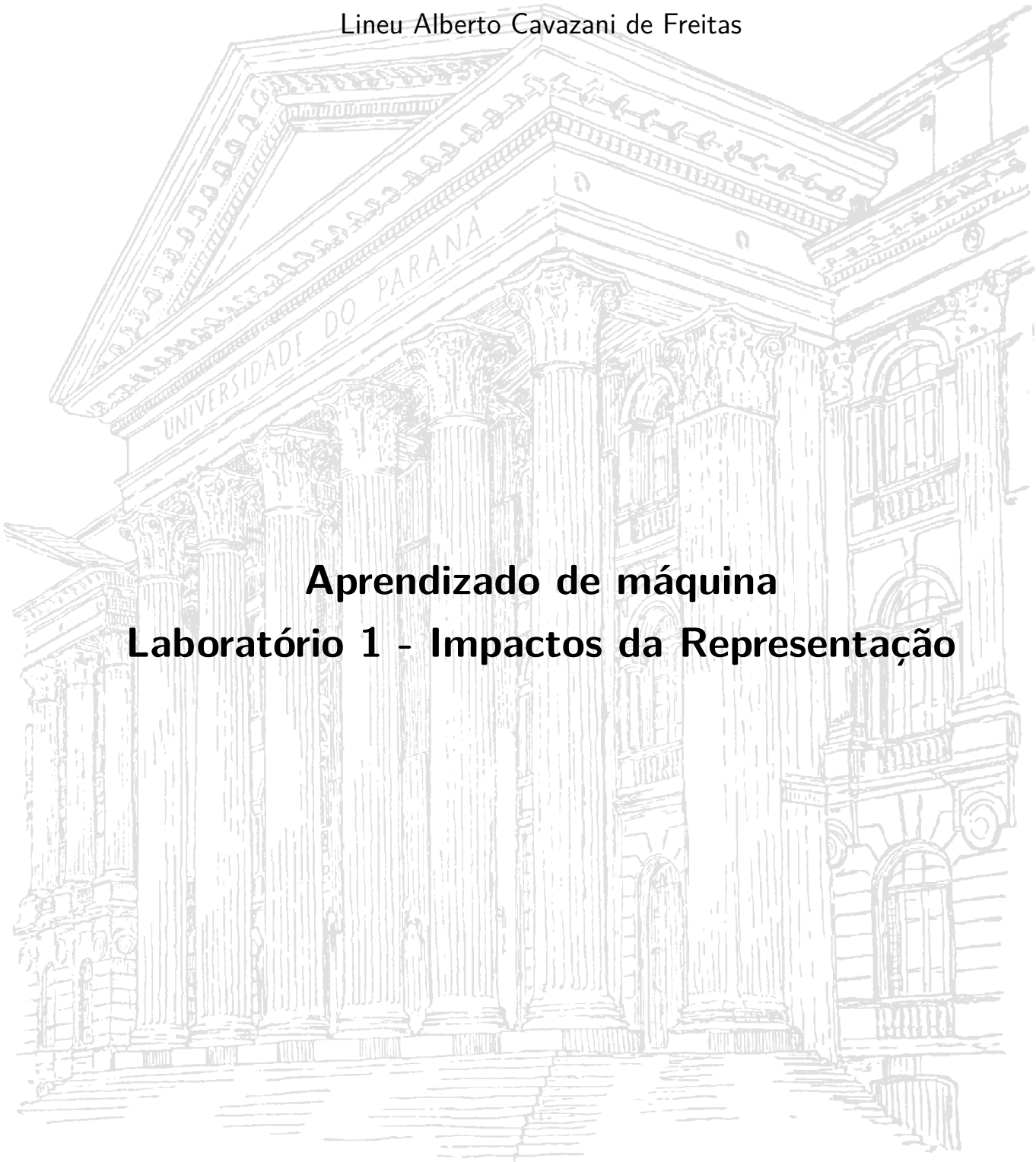
Universidade Federal do Paraná

Lineu Alberto Cavazani de Freitas



Aprendizado de máquina
Laboratório 1 - Impactos da Representação

Curitiba

2021

Lineu Alberto Cavazani de Freitas

Aprendizado de máquina
Laboratório 1 - Impactos da Representação

Relatório apresentado à disciplina Aprendizado de Máquina, ministrada pelo professor Luiz Eduardo Soares de Oliveira, no Programa de Pós Graduação em Informática da Universidade Federal do Paraná.

Universidade Federal do Paraná

Curitiba
2021

Sumário

1 Introdução

Aprendizado de máquina consiste em dar ao computador habilidade de aprender. Em outras palavras, o aprendizado de máquina consiste em programar computadores de forma que eles aprendam a partir de dados. Segundo T. Michell "o aprendizado de máquina trata do projeto e desenvolvimento de algoritmos que imitam o comportamento de aprendizagem humano, com um foco principal em aprender automaticamente a reconhecer padrões complexos e tomar decisões". A disseminação das técnicas de aprendizado de máquina se dão hoje pela disponibilidade de dados, poder computacional e ferramentas disponíveis.

No cenário em que temos dados rotulados e uma variável alvo definida por categorias, recomenda-se o uso de técnicas de aprendizado supervisionado para fins de classificação, os chamados classificadores. Dentre os possíveis existem aqueles baseados em distâncias tal como o *k Nearest Neighbors* (kNN). A ideia geral o kNN consiste em, para uma unidade, encontrar os k mais próximos (similares) a ele na base rotulada de treinamento e atribuir a classe mais frequente.

Uma desvantagem deste classificador é que para identificar os mais próximos é necessário obter a distância de um ponto para os outros, logo, existe um alto custo computacional. Contudo, em alguns casos o kNN funciona melhor que os concorrentes mais complexos, como por exemplo quando existem múltiplas fronteiras de decisão, o que se torna difícil de detectar por algoritmos de fronteira. Além disso, o kNN não requer treinamento, apenas teste, pois necessita apenas de distâncias. Por outro lado, a fase de teste demanda tempo considerável.

Um conhecido problema de classificação é o reconhecimento de dígitos manuscritos. Trata-se de um problema de classificação em que a entrada é uma imagem. Nestes casos há a necessidade de converter a imagem em um vetor de atributos. Tal conversão é chamada de representação que, de forma mais geral, consiste em converter determinado objeto num vetor de características, isto é, numa forma que a máquina entenda. O objetivo da representação consiste em caracterizar um objeto através de medidas que sejam similares entre indivíduos de mesma classe e diferente para indivíduos de outras classes.

Em problemas em que a entrada é uma imagem existem diversas técnicas para se extrair características. A mais simples consiste em, para cada posição da imagem, verificar o valor de intensidade do pixel e classificar esta intensidade como 0 ou 1, gerando desta forma, para cada imagem, um vetor de características em que cada elemento do vetor representa a intensidade naquela partição da imagem.

O objetivo deste relatório é apresentar os resultados de um problema de classificação de dígitos manuscritos utilizando kNN. No trabalho testou-se variações de algoritmos kNN e a representação das imagens foi feita verificando a intensidade dos pixels em cada partição da imagem.

2 Descrição da atividade

A tarefa consiste em gerar diferentes vetores de características para imagens e aplicar diferentes algoritmos kNN, variando valores de k e métricas de distância a fim de verificar qual conjunto de características e combinação de parâmetros do classificador produz os piores e melhores resultados. Para o trabalho foi disponibilizado um conjunto de 2 mil imagens rotuladas, além de um script Python para extração da represenação: para cada posição da imagem, verifica-se o valor de intensidade do pixel e se esse valor for maior que 128, a característica é igual a 1, caso contrário 0.

Como as imagens tem tamanho variável e os classificadores precisam de um vetor de tamanho fixo, as imagens são normalizadas para gerar o conjunto de dados no formato conveniente para utilização de classificadores. Além disso foi disponibilizado outro script Python com um exemplo de tamanho de imagem e parametrização de um kNN utilizando a biblioteca *scikit-learn* em que a base de dados foi dividida em 50% para treinamento e 50% para validação. Este script foi usado como base para realização do trabalho.

Foi realizado um experimento no qual variou-se o tamanho das imagens (que define o tamanho do vetor de características), o número de vizinhos do kNN, as métricas de distância e os algoritmos. As possibilidades testadas para cada uma destas variáveis foram:

- Tamanhos de imagem: 5x5, 10x10, 20x20, 30x30, 40x40.
- Número de vizinhos (k): 1, 5, 10, 20, 50.
- Métricas de distância: Manhattan, Euclidean, Chebyshev.
- Algoritmos: auto, ball_tree, kd_tree, brute.

O número de combinações únicas possíveis combinando cada nível é igual a 300. Cada combinação diz respeito a um modelo ajustado. Para cada uma destas combinações verificou-se o *precision*, *recall* e *F1-score*. O *precision* é indicado em casos que os falsos positivos são considerados mais prejudiciais que os falsos negativos. Em contrapartida, o *recall* é indicado a situações em que falsos negativos são mais prejudiciais. O *F1-score* considera ambas as métricas no cálculo. Quando o *F1-score* é baixo é um indicativo de que *precision* ou *recall* são baixos.

Através de análise gráfica do *F1-score* buscou-se verificar quais combinações apresentavam ajustes melhores ou piores. Após esta análise gráfica foram selecionados

os modelos em que, para cada dígito, o *F1-score* estava acima do quantil 75%. Destes modelos foram selecionados aqueles com o menor tamanho de imagem e menor número de vizinhos para explorar a matriz de confusão e acurácia. Estas análises foram realizadas no software R.

3 Resultados obtidos

A Figura 1 mostra os resultados dos experimentos para cada combinação possível entre tamanho da imagem (5x5, 10x10, 20x20, 30x30, 40x40), número de vizinhos (k=1, k=5, k=10, k=20, k=50), métrica de distância (Manhattan, Euclidean, Chebyshev) e algoritmo (auto, ball_tree, kd_tree, brute).

Ao analisar as quatro primeiras colunas da matriz de gráficos podemos observar que os resultados mostram um desempenho consideravelmente inferior das especificações considerando distância de Chebyshev quando comparada às distâncias Manhattan e Euclidean.

Analisando as linhas da matriz é possível avaliar o impacto do número de vizinhos em cada combinação. Nota-se, para as distâncias Euclidean e Manhattan, uma queda de desempenho conforme aumenta-se o número de vizinhos.

Em cada quadro são mostrados o desempenho para cada algoritmo, nota-se uma sobreposição dos pontos de diferentes cores, indicando que o desempenho entre algoritmos é similar.

Quanto ao tamanho das imagens, é possível observar que para poucos vizinhos parece haver uma pequena melhora de desempenho conforme aumenta-se o número de elementos no vetor de características. Contudo, há indícios de que tamanhos superiores a 20x20 não incrementam substancialmente o desempenho dos modelos.

De forma geral, os indícios apontam que um cenário com poucos vizinhos e tamanho de imagem a partir de 20 devem ser os melhores. Como não há diferença aparente entre algoritmos, deve-se optar pela mais rápida. O tempo de execução dos modelos não foi computado mas sabe-se a priori que os algoritmos que usam *spation partition tree* são mais velozes. Quanto a distância, Manhattan e Euclidean mostram desempenho similares, enquanto que a distância Chebyshev é bastante inferior às demais.

Com o objetivo de selecionar dentre as 300 possibilidades aquelas com o melhor desempenho, foram selecionados os modelos em que, para cada dígito, o F1-score estava acima do quantil 75%. A Tabela 1 apresenta os valores do quantil utilizado pra seleção.

Dígito	0	1	2	3	4	5	6	7	8	9
Quantil 75%	0.96	0.81	0.91	0.91	0.88	0.92	0.95	0.81	0.88	0.85

Tabela 1 – Quantil 75% do F1-score para cada dígito.

Das 300 combinações, 32 estavam dentro do critério. Uma análise mais cuidadosa

destes modelos mostrou que apenas modelos considerando número de vizinhos entre 1 e 5 e distância Manhattan ou Euclidean atendiam ao critério utilizado. Além disso metade dos 32 modelos possuiam vetor de características provenientes de imagens 20x20.

Destes 32 modelos foram selecionados aqueles com o menor tamanho de imagem e menor número de vizinhos para explorar a matriz de confusão e acurácia. Restaram 8 modelos com desempenho extremamente similar. Estes modelos foram especificados com 1 vizinho, distâncias Manhattan ou Euclidean e vetor de características vindos de imagens 20x20. Deste modo a tabela 2 mostra a matriz de confusão do modelo considerado o melhor: algoritmo kd_tree, 1 vizinho, distância Euclidean e imagem 20x20. Este modelo apresentou acurácia de 0.92. Das 1000 imagens usadas para avaliação do modelo apenas 81 foram classificadas de forma equivocada, mais da metade dos erros entre os dígitos 1 e 7.

	0	1	2	3	4	5	6	7	8	9
0	**94**	1				2				
1		**93**				1		1		
2	1	3	**101**	1				4		1
3				**97**		1		2	3	
4		8			**83**	1	1			2
5	1			5		**90**	1			
6	2	5			1		**98**			
7		3	1		1			**87**		5
8		4		1		4		2	**75**	1
9		1			2			8		**101**

Tabela 2 – Matriz de confusão do modelo final.

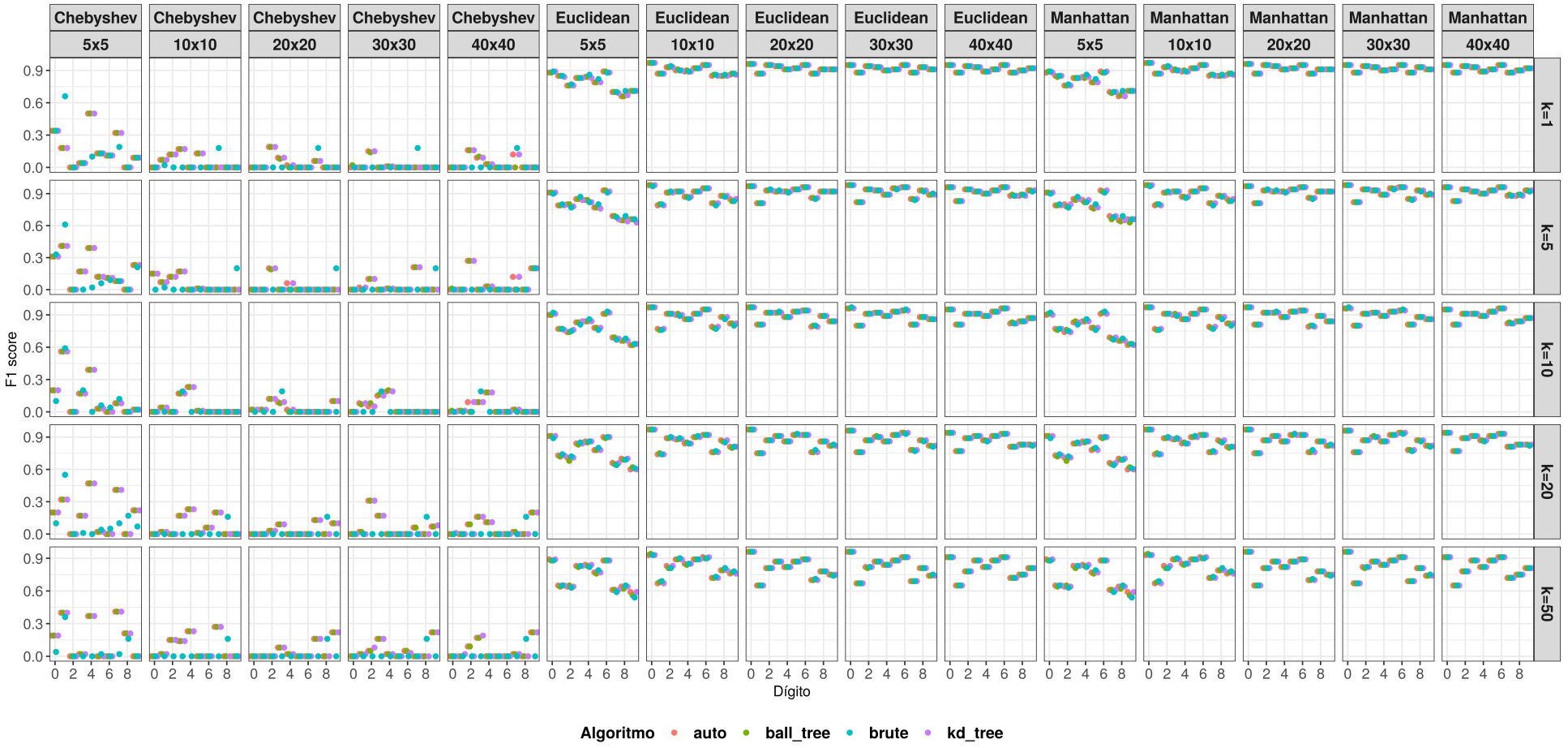


Figura 1 – F1-score para cada combinação do experimento.

4 Considerações finais

Os resultados mostraram que os melhores modelos eram aqueles com número reduzido de vizinhos, com distância Manhattan ou Euclidean e vetor de características vindos de imagens 20x20. Deste modo, não foi observado um ganho substancial ao aumentar-se o número de vizinhos ou o tamanho do vetor de características. As distâncias Manhattan ou Euclidean apresentaram resultados bastante similares e igualmente satisfatórios sendo que, em geral, opta-se pela distância Euclidean. Algo esperado e confirmado pelo experimento foi um desempenho similar entre diferentes algoritmos mostando que, para esse caso, a escolha pode ser simplesmente o mais rápido.

O experimento da forma que foi realizado apresenta algumas restrições: os resultados mostram que entre 1 e 5 vizinhos, a melhor performance é apresentada com apelas 1. Contudo em um experimento melhor delineado poderia ser investigado qual o melhor número de vizinhos testando uma quantidade maior de possibilidades, provavelmente seria um valor entre 1 e 5.

Como outra restrição pode-se citar o tamanho do vetor de características. Foram testados 5 tamanhos, todos quadrados, em que o melhor foi aquele de tamanho 20x20. Os resultados mostraram em algum momento não faz mais diferença o acrécimo de características ao vetor de entrada, sendo que, provavelmente um vetor entre 100 e 400 características é bastante razoável.

Inicialmente havia interesse em testar também como se saía a distância de Mahalanobis, contudo esta apresentou problemas e foi abandonada no começo dos experimentos.

Quanto aos algoritmos, não foi constatada uma diferença grande de desempenho entre as possibilidades, levando a crer que a escolha do mais rápido não traria prejuízos. Uma melhora neste experimento pode ser computar o tempo de execução nos cenários a fim concluir qual o mais rápido.

Por fim, vale ressaltar que esta análise foi meramente exploratória. Uma possibilidade seria, para cada cenário, gerar réplicas variando o conjunto de dados e analisar os resultados do experimento utilizando alguma metodologia estatística que permita estimar o efeito de cada fator sobre as métricas de qualidade de ajuste, tal como modelos de regressão.