PUCRS – FACIN
Inteligência Artificial – Engenharia da computação
Alunos: Guilherme Korol e Matheus Storck 03/07/2018
Prof.: Silvia Moraes

T4 – Aprendizagem de Máquina

1. Problema e objetivos

O problema proposto, conforme o enunciado, é o reconhecimento de dígitos manuscritos. Para tal, iremos explorar quatro técnicas de aprendizado de máquina. O *dataset* disponibilizado foi dividido em conjunto de treinamento e teste. Adicionalmente, o conjunto de teste foi incrementado com amostras editadas manualmente a fim de sofisticar a etapa de generalização. Com este trabalho, buscamos o aprofundamento prático no tópico de aprendizagem de máquina, complementando a fundamentação teórica vista durante as aulas.

2. Conjuntos de dados e Pré-processamento

Originalmente, o conjunto de dados foi gerado a partir da coleta de dígitos manuscritos de 43 pessoas, onde 30 delas contribuíram para o conjunto de treino e as restantes para o conjunto de teste. Cada amostra foi então pré-processada (por um filtro passa-baixas e *subsampling*) pelos autores do dataset de forma a extrair bitmaps normalizados em formato 32x32.

Para reduzir dimensionalidade, as amostras de 32x32 foram divididas em 64 janelas, sem sobreposição, de tamanho 4x4. Essa divisão gera amostras de tamanho 8x8, onde cada atributo é o número total de posições em um na respectiva janela 8x8 da matriz de entrada. Já para adaptação para o formato Weka, cada amostra de 8x8 atributos, mais a classe, foram formatadas de acordo. Formando, então, os arquivos de entrada para treinamento e teste.

Portanto, o conjunto total foi dividido em conjunto de treinamento e conjunto de teste. O primeiro, é constituído de 3823 amostras (68% do conjunto), enquanto o segundo possuí 1797 amostras (32%).

3. Algoritmos e parâmetros de treinamento

Como a tarefa proposta consiste em um problema de classificação, escolhemos os algoritmos Rede Multi-Layer Perceptron, Rede Bayesiana, Árvore de Decisão e K-Nearest neighbors (K-NN) para explorar neste trabalho.

a. Rede Multi-Layer Perceptron

A rede Multi-Layer Perceptron (MLP) é uma rede neural artificial do tipo *feedforward*, composta de três ou mais camadas (de entrada, saída e demais camadas ocultas).

As camadas ocultas são compostas de neurônios artificiais totalmente conectados as camadas vizinhas. Cada neurônio se resume a uma soma ponderada de todas suas entradas por determinados pesos, além de um termo bias. O resultado, então, é atribuído a uma função de ativação. Existem diversos tipos de funções de ativação, as mais usadas são sigmoide e tangente hiperbólica.

A aprendizagem em uma rede Multi-Layer Perceptron consiste em refinar os pesos de cada neurônio para que a camada de saída classifique a classe correta. O processo é realizado pelo algoritmo Error Backpropagation.

b. Redes Bayesianas

As redes Bayesianas mapeiam o relacionamento probabilístico entre os atributos de entrada em um grafo acíclico dirigido. Cada nodo da rede representa um evento, e as ligações entre os nodos são as probabilidades que relacionam tais eventos.

No caso mais simples, a rede pode ser construída por um especialista na área de aplicação e parte-se diretamente para generalização. Por outro lado, quando o método mais simples não é possível, deve-se usar algum algoritmo de aprendizado de máquina para aprender as relações entre atributos. Os algoritmos usados diferem quanto a forma e o objeto da aprendizagem. Em redes bayesianas, ora se tem a topologia do grafo e é preciso aprender os parâmetros, ora é preciso aprender ambos, topologia e parâmetros.

Iremos avaliar o desempenho de redes bayesianas com diferentes algoritmos de busca para encontrar a topologia. O algoritmo que encontra a topologia em redes bayesianas é o seguinte:

1. Lê entrada (dataset, Métricas de avaliação ou Scores e possíveis topologias).
2. Inicializa a estrutura (vazio, randomicamente, árvore, etc.)
3. Repete até que score(G_{t-1};D) = score(G_t;D)
4. Altere um vértice que resulte na melhor melhora de score possível entre todas operações possíveis.
5. $G_{final} = G_t$

Onde G é um grafo (rede bayesiana) e D é o *dataset* de entrada. Os algoritmos explorados neste trabalho são aplicados no passo 4.

c. Árvores de Decisão

Basicamente, o algoritmo parte de uma modelagem de um conjunto de decisões específicos (e suas consequências) em uma árvore. Árvores de decisão podem apresentar três tipos de nodos: nodos de decisão, de chance, e nodos finais (folhas).

Para nosso problema de classificação, a árvore deve conter dez folhas (uma para cada classe). Um dos algoritmos mais populares para aprendizagem em árvores de decisão é o de Partição Recursiva (Recursive Partioning). Em linhas gerais, o algoritmo reparte recursivamente o conjunto de dados isolando variáveis independentes até que o conjunto retorne (ou classifique) a mesma classe para todos os seus indivíduos. Dessa forma, o algoritmo aprende os "critérios de decisão" que devem estar nos vértices para classificar corretamente.

d. K-Nearest neighbors (K-NN)

O algoritmo K-NN é um algoritmo simples de um único parâmetro, K. Consiste em classificar uma determinada amostra baseada na classe mais comum entre os K vizinhos mais próximos.

Sua execução inicia a partir do cálculo da distância entre a amostra e seus vizinhos. Para este cálculo, diversas métricas podem ser utilizadas. As mais usuais são a distância Euclidiana (para dados contínuos) e a distância de Hamming (para dados discretos).

Neste trabalho, nos dedicamos a seleção do valor K, o qual define a quantidade de vizinhos participantes da votação da classe da amostra, que resulte na melhor classificação. A escolha deste parâmetro está diretamente ligada a natureza dos dados. Valores pequenos de K resultam em classificações mais flexíveis, porém com alta variância. Valores altos de K resultam em pouca variância, mas grande propensão ao erro dada a maior quantidade de votantes.

4. Análise dos resultados

a. Rede Multi-Layer Perceptron

Para rede MLP, treinamos com diversos números de épocas e para dois valores de taxa de aprendizagem. O conjunto de teste foi utilizado para avaliar o erro em todos os resultados reportados nesse trabalho. A Figura 1 apresenta as taxas da Raiz do Erro Médio Quadrático (RMSE em inglês). Usamos RMSE por se tratar de uma métrica que informa "quão concentrado" as predições estão entorno da classe correta (acurácia).

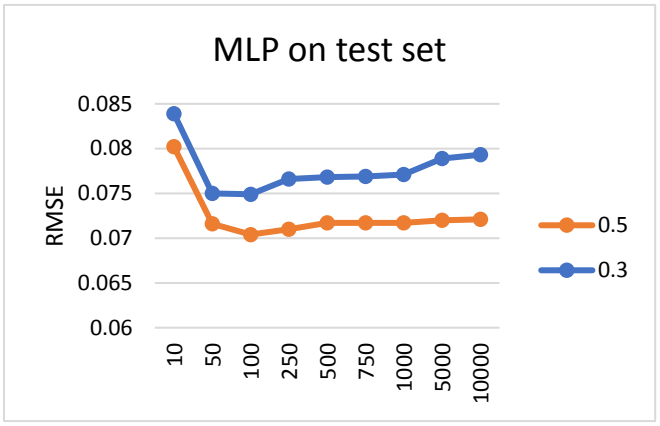

Figura 1. RMSE para taxas de aprendizagem de 0.3 e 0.5.

Podemos observar que os melhores resultados foram obtidos com taxa de 0.5 e que a partir de, aproximadamente, 500 épocas o erro estabiliza para a mesma taxa de aprendizagem. Porém, com apenas 100 épocas atinge o menor valor. Ao treinar para número de épocas maiores que 100, observamos o efeito de *overfitting* do modelo. Por esse motivo, iremos apresentar na Tabela 1 precisão, abrangência e F1 para o modelo com 100 épocas e taxa de aprendizagem de 0.5 (modelo de menor RMSE).

Classe	Precisão	Abrangência	F1
0	0.994	0.989	0.992
1	0.948	0.995	0.971
2	0.978	0.994	0.986
3	1	0.951	0.975
4	0.973	0.983	0.978
5	0.937	0.984	0.96
6	1	0.989	0.994
7	0.982	0.922	0.951
8	0.975	0.908	0.94
9	0.917	0.978	0.946
Média:	0.97	0.969	0.969

Tabela 1 - Valores de Precisão, Abrangência e F1 por classe. 100 Épocas e 0.5 de taxa de aprendizagem.

b. Redes Bayesianas

Na classificação com redes bayesianas, exploramos diferentes algoritmos de busca para encontrar a topologia da rede. Iremos comparar os algoritmos K2, Hill Climber, Repeated Hill Climber e LAGD Hill Climber. Outros algoritmos que poderiam ser usados, como Simulated Annealing e Algoritmos Genéticos, não executaram nas máquinas que utilizamos.

Os algoritmos testados não apresentaram diferenças nos resultados da classificação com redes bayesianas. Podemos concluir que todos encontraram a mesma topologia de grafo. Na Tabela 2, está representado precisão, abrangência e índice F1 para todas classes na classificação por rede bayesiana.

Classe	Precisão	Abrangência	F1
0	0.989	0.978	0.983
1	0.833	0.824	0.829
2	0.895	0.87	0.883
3	0.94	0.858	0.897
4	0.887	0.95	0.917
5	0.961	0.94	0.95
6	0.989	0.956	0.972
7	0.922	0.927	0.925
8	0.843	0.833	0.838
9	0.783	0.883	0.83
Média:	0.904	0.902	0.903

Tabela 2 - Valores de Precisão, Abrangência e F1 por classe para rede bayesiana.

c. Árvores de Decisão

Executamos o algoritmo *Random Tree*, que implementa o algoritmo de árvores de decisão, porém sem realizar a "poda" nos nodos de chance. Notamos que ao realizar apenas o treinamento alcançamos 0% de RMSE, o que indica que o modelo é capaz de classificar corretamente todas as instâncias do conjunto de treino. Em contrapartida, quando realizamos o teste (generalização) após treinamento, os resultados foram muito discrepantes com o esperado, conforme Figura 2. O que nos indica uma dificuldade que o modelo de árvore de decisão tem em generalizar os parâmetros aprendidos para amostras que não foram "vistas" durante o treino.

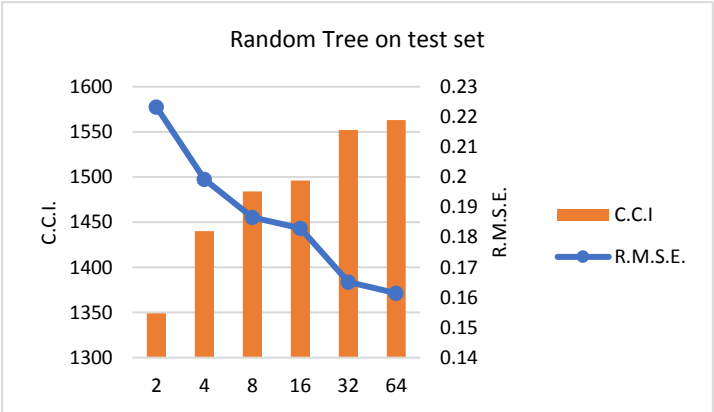

Figura 2. Instâncias corretamente classificadas (CCI) e RMSE para valores randômicos K.

A etapa de treino foi executada para um valor fixo de *seed* e com diversos valores de K (de 2 até 64). O primeiro parâmetro indica a semente para o mecanismo de randomização e o segundo, indica a quantidade de atributos randômicos que serão escolhidos. O fato de não ser possível alcançar valores próximos aos obtidos na etapa de treinamento se dá pelo

algoritmo não utilizar o mecanismo de "poda", o que acaba causando *overfitting* e aumentando o erro de classificação na etapa de teste.

Como o valor de K igual a 64 resultou no melhor número de instâncias corretamente classificadas, decidimos utilizar este como métrica para precisão, abrangência e índice F1, conforme Tabela 3:

Classe	Precisão	Abrangência	F1
0	0.983	0.949	0.966
1	0.8	0.857	0.828
2	0.949	0.847	0.896
3	0.804	0.831	0.817
4	0.79	0.934	0.856
5	0.921	0.896	0.908
6	0.94	0.945	0.942
7	0.922	0.793	0.853
8	0.814	0.805	0.809
9	0.821	0.839	0.83
Média:	0.874	0.87	0.87

Tabela 3. Valores de Precisão, Abrangência e F1 por classe para Random Tree (K = 64 e Seed = 64).

d. K-Nearest neighbors

A partir da análise do algoritmo K-NN, fica claro que o grande desafio é encontrar um valor de K que resulte na melhor classificação.

Primeiro, além de realizamos uma comparação entre oito valores de K arbitrariamente escolhidos (1, 2, 4, 8, 16, 32, 64), confirmamos a melhor opção de K baseada na técnica de *Cross-validate*. Esta técnica realiza a escolha de K realizando validação cruzada no conjunto de treino para todos valores entre uma faixa escolhida (de 1 à 64, no nosso caso).

A Figura 3 apresenta RMSE e a quantidade de Instâncias Classificadas Corretamente (CCI em inglês) para alguns valores de K. Verifica-se que dentre as opções para K, o valor 4 é o que garante o menor RMSE, porém, o que melhor classificou as instâncias (acurácia, precisão e abrangência) foi o valor 1.

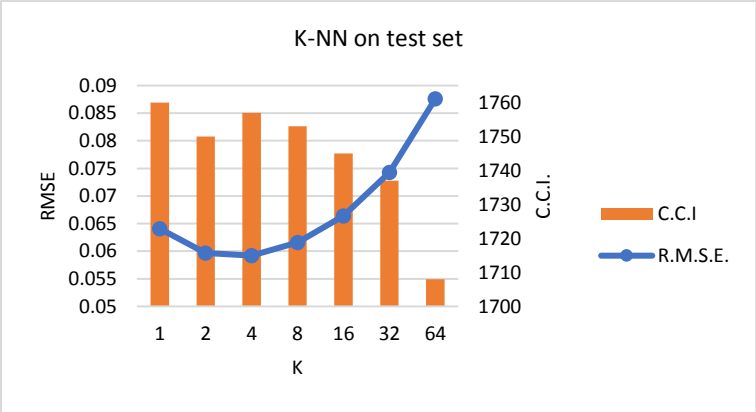

Figura 3. CCI e RMSE para valores de K.

Acreditamos que o fato do melhor desempenho estar em K = 1 se dá por conta da distribuição dos dados. Pois, para valores altos de K, seria muito difícil distinguir entre os limites das classes. A Tabela 4 exibe os valores de precisão, abrangência e índice F1 para todas classes.

Classe	Precisão	Abrangência	F1
0	1	1	1
1	0.933	0.995	0.963
2	1	0.989	0.994
3	0.978	0.978	0.978
4	0.983	0.983	0.983
5	0.989	0.984	0.986
6	1	1	1
7	0.989	0.983	0.986
8	0.97	0.937	0.953
9	0.955	0.944	0.95
Media:	0.98	0.979	0.979

Tabela 4. Valores de Precisão, Abrangência e F1 por classe para K-NN (K = 1).

5. Conclusões

Ao final, propomos uma comparação entre as técnicas apresentadas. Usaremos um arquivo de generalização criado artificialmente a partir do conjunto de teste. O arquivo conta com 20 amostras para cada classe. No conjunto, foram escolhidas, randomicamente, as posições e a quantidade de "ruído" acrescentado nos atributos de entrada. Por exemplo:

```
0,0,5,13,9,1, ... ,0,0,5,0,0,3,15,2,0,11,8,0,0,4,12,0, ,...,0,6,13,10,0,0,0,0
0,0,5,7 ,9,1, ... ,0,0,5,0,0,3,15,2,0,11,8,6,0,4,12,0, ,...,0,6,13,14,0,0,0,0
```

Onde três atributos, randomicamente escolhidos, foram modificados por quantias randomicamente escolhidas. Assim, estaríamos alterando a quantidade de pixels em valor um na imagem normalizada de entrada. Representando, então, um ruído que avaliaria de forma mais exigente as técnicas em uso.

A Tabela 5 apresenta o resultado da comparação.

Técnica	Precisão Média	Abrangência Média	F1 Médio
MLP	0.956	0.955	0.955
Rede Bayesiana	0.876	0.850	0.845
Árvore de Decisão	0.907	0.885	0.885
K-NN	0.977	0.970	0.969

Tabela 5 - Resultado da comparação entre as quatro técnicas utilizando arquivo de generalização.

Frente ao conjunto de generalização criado, a técnica K-NN mostrou os melhores resultados. Entendemos que o K-NN se mostrou menos suscetível aos "ruídos" criados, devido a forma de com que o algoritmo classifica as amostras. Alterando o valor de atributos, estamos deslocando a amostra no plano. No entanto, esse pequeno deslocamento parece não afetar a classificação por K-NN que avaliaria a classe das amostras "vizinhas".

Por outro lado, os piores resultados apareceram na técnica de rede bayesiana. Como explicamos anteriormente, acreditamos que a técnica apresenta dificuldades para expressar um relacionamento estatístico nos dados. Este comportamento foi acentuado pela adição de ruído no conjunto de generalização e o uso de algoritmos de busca não sofisticados.

Por fim, ressaltamos que se poderia criar um conjunto de generalização mais realista a partir de novas imagens onde poderíamos alterar orientação, brilho, contraste, cor, etc. E a partir delas, pré-processa-las e chegar em um dataset de generalização mais robusto que o utilizado.

6. Referências

E. Alpaydin, C. Kaynak. Optical Recognition of Handwritten Digits Data Set. Disponível em: <https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits> Acesso em: 29 de Junho de 2018