Extração de Textura com (LBP) Local Binary Pattern e Classificação

Gissely de Souza

Resumo—Este relatório apresenta o processo de extração de características de uma bases de dados, aplica vários classificadores combinados nas características extraída e depois comparar os resultados. Primeiramente nesse relatório é apresentado o método utilizado tanto para a extração como para classificação e execução dos classificados, em seguida os experimentos realizados e por fim os resultados obtidos.

Index Terms—Extração de Característica, Classificadores

1 Introdução

Técnicas de aprendizagem de máquina podem ser utilizadas para encontrar padrões em diversos domínios, inclusive em imagens. É neste ponto que a linha de pesquisa de aprendizagem de máquina, advinda da área de inteligência artificial, se encontra com a linha de pesquisa de reconhecimento de padrões, advinda da área de processamento de sinais. O fluxo padrão para soluções de reconhecimento de padrões consiste em três etapas:

- Filtragem e pré-processamento da entrada;
- Extração e seleção de características;
- Classificaçã;

Um problema de classificação consiste na determinação de regras e posterior classificação desses exemplos. Este conjunto de regras é criado por um classificador, que recebe como entrada um vetor de características e oferece como saída uma classe resultante para a instância que as características descrevem. Dos diferentes classificadores já existentes na literatura e como cada um destes classificadores apresenta vantagens e desvantagens, apresentando maior ou menor desempenho em determinados problemas torna-se necessário não apenas conhecer os classificadores e seu funcionamento, mas também ser capaz de analisar e comparar o desempenho dos mesmos e é este um dos objetivos principais deste trabalho, treinar classificadores e conjuntos de classificadores e comparar seus resultados para se descobrir qual deles obtém a melhor performance para um determinado problema de classificação de texturas.

Porém independente de qual classificador se esteja usando, nenhum deles seria capaz de obter um bom resultado sem a extração de boas características relevantes ao problema enfrentado, e é este o outro grande objetivo deste trabalho. Encontrar e extrair um conjunto de características que permita um melhor resultado para os vários classificadores utilizados.

2 Método

Durante a primeira etapa de realização deste trabalho foi feita a decisão de qual extrator de características que seria usado para extraír da base de imagens TrainVal fornecidas para a classificação, para este problema de texturas resolvi usar o local binary pattern (LBP) para testar como se compartaria, foi também considerado a utilização de um histograma de níveis de cinza além do LBP, porém este histograma, após testes preliminares não resultou em nenhuma mudança notável nos resultados obtidos.

Foi preparada a base de dados das imagens para a leitura, para extração de texturas e racterísticas de cada imagem, foram escolhidos os classificadores que seriam utilizados. Uma vez as características extraídas a base foi divida em duas partes; em treinamento e validação, conforme a especificação. Para os testes realizados a base foi divida em uma proporção de 60% dos exemplos de cada classe para treinamento e 40% dos exemplos de cada classe para validação, essa divisão é feita aleatoriamente de forma que a cada execução a base de treinamento e de validação seja diferente. A base foi dividida 10 e depois 30 vezes de forma aleatória e os classificadores executados uma vez para cada combinação de teste e validação. Cada combinação de teste e validação gerou uma taxa de reconhecimento e uma matriz de confusão, dessa forma o resultado considerado é a media dos 10 e 30 experimentos para cada classificador.

Após alguns testes também surgiu a ideia de normalizar a base e averiguar se ocorreria mudança na taxa de reconhecimento, a forma de normalização utilizada foi a Min-Max, que consiste em encontrar o valor mínimo e máximo para cada característica, então após isso aplicar a seguinte formula:

$$zi = \frac{xi - min(x)}{max(x) - min(x)}$$

Onde xi é o valor não normalizado, min(x) é o menor valor para aquela característica na base e max(x) o maior valor para aquela característica na base. Dessa forma todos os valores da base estariam entre 0 (caso fosse o menor valor) e 1 (caso fosse o maior valor).

Os classificadores utilizados para os experimentos foram: Naive Bayes, Decision Tree, KNN, SVM (Support Vector Machines) utilizando a implementação do scikit-learn. A maioria dos classificadores utilizou a combinação padrão de parâmetros do scikit-learn para realizar os testes, além de algumas outras configurações, em busca de melhoria nos resultados. A exceção do classificador SVM, onde prelimi-

narmente foi utilizada a biblioteca libSVM para realizar a grid-search pela melhor combinação de parâmetros, Como é visto no gráfico da FIGURA 1 uma vez esses parâmetros encontrados eles foram utilizados na implementação do scikit-learn.

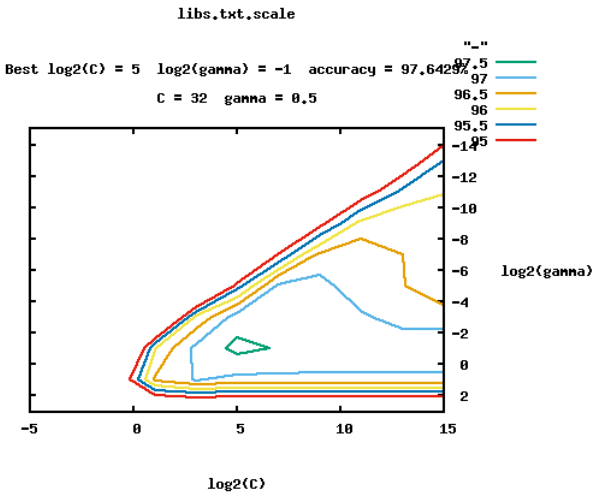

Figura 1. Gráfico - Parâmetros SVM LIBS

Realizados os primeiros testes sobre os classificadores executados individualmente foram testadas estratégias para a combinação em série e em paralelo de múltiplos classificadores, assim como também foram testadas estratégias de ensembles. Para os classificadores em paralelo foram executados um grupo de classificadores para a mesma base de treinamento e validação, o resultado desses classificadores foi então usado numa votação para decidir qual seria a classificação final de cada entrada. Para tal votação foi utilizada a função VotingClassifier do skikit Learn, e foram testados os parâmetros hard vote e soft vote que são respectivamente um voto majoritário e um voto que leva em consideração as probabilidades de cada predição. Tanto com o parametro soft como hard foram utilizados os mesmos classificadores, KNN, arvore de decissão e naive bayes com o mesmo peso para todos os classificadores com a configuração padrão de parametros do scikit-learn.

Para os classificadores em série um único classificador é executado por vez, se a probabilidade de acerto para uma determinada amostra está abaixo de um determinado limiar manualmente definido, essa amostra deverá ser classificada pelo próximo classificador.

Para as estratégias de ensembles foram utilizadas as implementações do skikit learn de bagging e random forests. Foram feitos baggings de Naive Bayes, Decision Tree, KNN e SVM, com parâmetros 50% para o máximo de características a ser utilizada para cada classificador do bagging e 50% de máximo de exemplos para treinar cada bagging. O número de classificadores em cada bagging é 10, valor padrão do scikit-learn.

3 Experimentos

A realização dos experimentos procedeu da seguinte forma, uma vez a base dividida todos os classificadores com diversas configurações foram treinados com a mesma partição da base para treino (60%) validados (40%) com os mesmos exemplos restantes, ou seja, cada configuração foi treinada e validada com os mesmos exemplos em cada repartição da base.

As seguintes configurações de classificadores foram utilizadas: KNN com parametro K = 3, 5 (padrão), 7, Decision tree com parâmetro que determina quantas características serão utilizadas para a classificação = todas (padrão), raiz quadrada do total, log base 2 do total de características, Naive Bayes com configuração padrão,SVM apenas foi utilizada a seguinte configuração, kernel RBF, c = 32 e gamma = 0.5.

Quanto a utilização de Baggings, sempre a configuração padrão foi utilizada para formar os elementos do bagging, com exceção do SVM que utiliza os parâmetros descritos anteriormente.

Quanto as estratégias de combinação em série os limiares de rejeição sempre foram 0.9 para o primeiro classificador, 0.6 para o segundo e sem limite, ou 0, para o último classificador.

4 Resultados

A seguir os resultados dos experimentos onde o tamanho do histograma do lbp que foi gerado, resultaram em 26 caracterısticas. As tabela contendo a precisão média obtida por cada classificador, o P-value com 5 casas decimais de precisão, entre o melhor classificador e os demais. E uma comparação dos resultados da base normalizada e não normaliza para a divisão aleatória da base de dados extraídos.

4.1 Tabelas com os resultados

4.1.1 Não normalizada: Base dividida por 10 vezes

Tabela 1
Base não normalizada

Random Forests (Default)	0.871964
Bagging (KNN)	0.858214
Bagging (Decision Tree)	0.843214
KNN (K = 7)	0.855179
KNN (default)	0.865089
KNN (K = 3)	0.873036
Voting (Hard)	0.842946
Voting (Soft)	0.837946
Serialize (SVM, Decision Tree, KNN)	0.006875
Decision Tree	0.792857
SVM (C = 32, RBF, Gamma = 0.5)	0.628304
Decision Tree (Sqrt)	0.755714
Decision Tree Log	0.751875
Serialize (SVM, NayveBayes, KNN)	0.003125
Bagging (SVM)	0.423929
Nayve Bayes	0.619464
Bagging (Nayve Bayes)	0.625089

Na Tabela 1 Podemos observar que o melhor desempenho para a base não normalizada, foi para o Random Forests (Default) 0.87% de acertos seguido pelo KNN (K = 3) 0.87%, KNN (default) 0.86%, KNN (K = 7) 85%, Bagging (KNN) 0.85%. O Resultado que chama atenção é para o SVM, mesmo com Baggins não obteve boa precisão. A estratégia de introduzir um SVM no início de um Serializado, não teve efeito pois um SVM é de mais custo do que um KNN.

4.1.2 Não normalizada: Base dividida por 30 vezes

Tabela 2
Base não normalizada

Random Forests (Default)	0.870804
Bagging (KNN)	0.853304
Bagging (Decision Tree)	0.845060
KNN (K = 7)	0.852679
KNN (default)	0.863006
KNN (K = 3)	0.872321
Voting (Hard)	0.842024
Voting (Soft)	0.835417
Serialize (SVM, Decision Tree, KNN)	0.006875
Decision Tree	0.790357
SVM (C = 32, RBF, Gamma = 0.5)	0.628661
Decision Tree (Sqrt)	0.762054
Decision Tree Log	0.754226
Serialize (SVM, NayveBayes, KNN)	0.006875
Bagging (SVM)	0.389196
Nayve Bayes	0.613452
Bagging (Nayve Bayes)	0.610565

Na Tabela 2 Mesmo redividindo a base de dados por 30 vezes, os resultados obtidos permaneceram parecidos com o anterior. Confirmando e retificando os valores. Desta vez o KNN (K = 3) conseguiu ficar a frente do Random Forests (Default).

4.1.3 Normalizada: Base dividida por 10 vezes

Tabela 3
Base Normalizada

Random Forests (Default)	0.870089
Bagging (KNN)	0.872679
Bagging (Decision Tree)	0.842679
KNN (K = 7)	0.891071
KNN (default)	0.896786
KNN (K = 3)	0.906161
Voting (Hard)	0.844375
Voting (Soft)	0.842321
Serialize (SVM, Decision Tree, KNN)	0.004643
Decision Tree	0.795268
SVM (C = 32, RBF, Gamma = 0.5)	0.968571
Decision Tree (Sqrt)	0.763571
Decision Tree Log	0.756786
Serialize (SVM, NayveBayes, KNN)	0.004643
Bagging (SVM)	0.927054
Nayve Bayes	0.619643
Bagging (Nayve Bayes)	0.619732

Na Tabela 3 Nesta etapa dos testes de classificação, foi normalizada a base de dados de características, e resultou uma ótima performace do SVM (C = 32, RBF, Gamma = 0.5) com 96% de acertos. Em seu combinadi Bagging (SVM), também obteve boa acurácia. Já o KNN que já vinha com um bom acerto, aumentou sua precisão aqui também como todos os demais classificadores. O que não melhorou, foi os serelizados que iniciam com o SVM, confirmando que para os seriarelizados, não é uma boa estratégia iniciar com um classificador de maior custo.

4.1.4 Normalizada: Base dividida por 30 vezes

Tabela 4
Base Normalizada

Random Forests (Default)	0.868065
Bagging (KNN)	0.875417
Bagging (Decision Tree)	0.848244
KNN (K = 7)	0.888661
KNN (default)	0.895833
KNN (K = 3)	0.904911
Voting (Hard)	0.842917
Voting (Soft)	0.834405
Serialize (SVM, Decision Tree, KNN)	0.004583
Decision Tree	0.792232
SVM (C = 32, RBF, Gamma = 0.5)	0.967143
Decision Tree (Sqrt)	0.759762
Decision Tree Log	0.753571
Serialize (SVM, NayveBayes, KNN)	0.002857
Bagging (SVM)	0.927798
Nayve Bayes	0.616339
Bagging (Nayve Bayes)	0.613036

Na Tabela 4 Os resultados aqui, só confirma a iteração normalizada por 10 vezes. A quantidade de vezes em que a base é particionada, não interfere nos resultados.

4.2 Conclusão

Pode-se observar nas tabelas exibidas que, em bases normalizadas as melhores taxas de acerto são maiores do que as melhores taxas de acerto em bases não normalizadas, dessa forma pode-se concluir que normalizar a base oferece melhores resultados.

Caso os exemplos sejam normalizados o classificador SVM obtém resultados um pouco melhores que os outros classificadores.Dessa forma concluí-se que para o problema em questão o classificador SVM com parâmetros de kernel = RBF, C = 32, gamma = 0.5 é a melhor solução desde que os exemplos sejam normalizados, superando inclusive estratégias de combinação e de ensambles.

Entretanto caso os exemplos não possam ser normalizados pode-se verificar que o classificador KNN obtém os melhores resultados seguido do classificador Decision Tree, entretanto ambos são superados pela estratégia de ensambles Random Forests.

APÊNDICE A
MATRIZES DE CONFUSÃO DOS MELHORES CLASSIFICADORES

Tabela 5
Random Forests (Default) - 10 vezes - Não normalizada

103	0	4	0	3	0	0	0	0	0
0	100	0	8	0	0	1	1	0	0
10	0	89	0	3	0	4	0	4	0
0	8	0	95	0	1	5	1	0	0
9	0	2	0	85	1	0	3	1	6
0	0	0	3	1	102	0	3	0	0
0	2	6	7	0	0	91	0	3	0
0	0	0	1	0	4	0	104	0	0
0	0	7	0	0	0	4	0	98	0
0	0	0	0	6	0	0	0	0	103

Tabela 6
SVM (C = 32, RBF, Gamma = 0.5) - 10 vezes - Normalizada

110	0	1	0	0	0	0	0	0	0
0	108	0	3	0	0	0	0	0	0
1	0	108	0	0	0	0	0	1	0
0	4	0	105	0	0	2	0	0	0
1	0	0	0	107	0	0	0	0	0
0	0	0	0	0	110	0	1	0	0
0	1	0	3	0	0	105	0	0	0
0	0	0	0	0	0	0	110	0	0
0	0	2	0	0	0	0	0	108	0
0	0	0	0	0	0	0	0	0	110

Tabela 7
KNN (K = 3) - 10 - Não normalizada

103	0	5	0	2	0	0	0	0	0
0	100	0	8	0	0	1	1	0	0
11	0	91	0	1	0	2	0	4	0
0	10	0	93	0	0	4	3	0	0
18	0	5	0	74	0	0	6	0	4
0	0	1	2	0	102	0	4	0	0
0	2	4	5	0	0	95	0	4	0
0	0	0	1	0	1	0	108	0	0
0	0	8	0	0	0	2	0	100	0
1	0	0	0	2	0	0	0	0	106

Tabela 8
SVM (C = 32, RBF, Gamma = 0.5) 30 vezes - Normalizada

110	0	1	0	0	0	0	0	0	0
0	108	0	3	0	0	0	0	0	0
1	0	108	0	0	0	0	0	1	0
0	4	0	104	0	0	2	0	0	0
1	0	1	0	106	0	0	0	0	0
0	0	0	0	0	109	0	1	0	0
0	2	0	3	0	0	105	0	0	0
0	0	0	0	0	0	0	110	0	0
0	0	2	0	0	0	0	0	108	0
0	0	0	0	0	0	0	0	0	110