Treinamento Machine Learning – LBP + KNN

Paulo Ricardo Lisboa de Almeida
Departamento de Informática
Universidade Federal do Paraná – UFPR
Curitiba, Brasil

I. Introdução

Este documento tem objetivo de ser um guia de estudos sobre conceitos básicos de Machine Learning clássico, incluindo conceitos como extração de características e criação de classificadores com aprendizagem supervisionada. Siga os passos em ordem, como descrito em cada seção.

As descrições e referências que você vai encontrar nesse texto servem como um norte para sua pesquisa, mas provavelmente não serão o suficiente. Procure por textos e tutoriais na internet para te ajudar a cumprir cada tarefa e, quando necessário, procure ajuda do professor.

II. Imagens e PKLot

Neste tutorial serão utilizadas imagens para criar um sistema de classificação. Mas tenha em mente que um sistema de classificação pode tomar qualquer tipo de sinal como entrada (som, vídeo, dados tabulados em um CSV, ...).

Para começar, faça o download da base PKLot, disponível em https://web.inf.ufpr.br/vri/databases/parking-lot-database. A descrição dessa base de dados pode ser encontrada em [1] e [2].

Você deve usar a versão mais recente da biblioteca OpenCV, usando C++ ou Python (escolha) para cumprir as seguintes tarefas:

1) Abrir e exibir uma imagem qualquer da PKLot na tela.
2) Usar os arquivos XML disponíveis para cada imagem da PKLot, para recortar e salvar cada imagem de vaga individual em diretórios.
   - As posições das vagas são retângulos rotacionados. Você vai precisar usar a mesma estratégia descrita em [1] para rotacionar e transformar em um retângulo não rotacionado e salvar em um jpg.
   - A PKLot já tem essas imagens salvas em um diretório. Verifique essas imagens e use de exemplo. Após o seu processamento, o resultado deve ser idêntico ao existente na PKLot.

Tenha em mente que toda imagem em um espaço de cores RGB é uma matriz de três dimensões. Não passe para as próximas seções antes de entender esses conceitos claramente. Um bom livro para entender esses conceitos é [3].

III. Características e Extratores de Características

Para classificar um determinado item, é necessário extrair suas características relevantes, que descrevem o objeto usando um vetor de tamanho fixo e não muito grande (geralmente de no máximo algumas milhares de posições).

Entenda o que é um vetor de características, e por que é importante o extrair de um sinal. Bons livros onde você pode encontrar essas informações incluem [4], [5], e [6]. Não se preocupe em se aprofundar em conceitos sobre os extratores, mas entenda o que é um vetor de características, e seus conceitos básicos.

Feito isso, extraia características LBP para uma imagem qualquer da PKLot usando o OpenCV. Exiba a imagem resultante na tela. A descrição do que é um LBP pode ser encontrada em [1]. Esse tutorial também pode te ajudar: https://pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv.

Extrair o LBP para cada pixel da imagem não gera um vetor de características. Mas se você computar um histograma dos dados do LBP gerados, você poderá usá-lo como vetor de características. Você pode encontrar uma boa descrição sobre o que é um histograma, e como usá-lo com o OpenCV em [7]. Exiba o histograma resultante na tela.

Feito isso, extraia o histograma LBP de cada umas das quase 700k imagens recortadas na seção II. Esse histograma será o vetor de características de cada imagem. Salve cada vetor em um único arquivo CSV (que vai conter cerca de 700k linhas), no formato:

dado0img0;dado1img0;dado2img0;...;dado255img0;classeimg0
dado0img1;dado1img1;dado2img1;...;dado255img1;classeimg1
...
dado0imgn;dado1imgn;dado2imgn;...;dado255imgn;classeimgn

Onde classe vai ser 1 se a imagem representava uma vaga ocupada, ou 0 se a imagem representava uma vaga vazia.

IV. Treinamento e Testes

Separe os dados entre treinamento e teste. Note que esse passo é crucial e é fácil fazer besteira aqui. Existem várias formas de se fazer isso. Pesquise na literatura citada anteriormente, e na internet.

A ideia é criar um protocolo de testes que reflita o mundo real. Você deve pegar uma proporção dos dados para treinar um modelo de *machine learning* (você fará isso nos próximos passos), e separar uma porção dos dados para testar esse modelo. Seu objetivo é concluir como o modelo treinado se comportará no mundo real.

Pense em como criar um protocolo de testes que evita resultados enviesados (essa é a tarefa básica de todo Cientista).

Descreva seu protocolo de testes em um documento de texto e, só depois de criá-lo, compare com o protocolo proposto em [1], e também com a discussão de [2]. Você conseguiu separar os dados entre treinamento e teste sem injetar nenhum viés? Se tinha viés, qual era?

V. Normalização

Normalize os **dados de treinamento** usando a técnica min-max. Pesquise na literatura e internet como ela funciona, e o motivo da normalização geralmente ser benéfica. Salve os parâmetros de normalização em um arquivo ou algo semelhante, para que você possa normalizar instância de teste posteriormente.

VI. Treinamento Supervisionado

No treinamento supervisionado, são dadas amostras de treinamento rotuladas para treinar um classificador (lembra dos dados que você extraiu o LBP, e depois separou em treinamento e teste?). Os dados de treinamento serão os vetores de característica, acompanhados de suas classes (vaga ocupada ou vazia). Um classificador deve ser capaz de aprender com esses dados.

Crie um classificador (programe do zero, não use nada pronto) do tipo K-NN, que significa *k-nearest neighbors*, ou o algoritmo dos k vizinhos mais próximos. Pesquise na literatura. Em resumo, esse é um dos algoritmos mais simples de se entender e implementar.

Para exemplificar, considere um 1-NN (você pode estender essa ideia para k posteriormente). A ideia é, dada uma amostra de teste x (a qual desejamos classificar entre ocupada ou vazia), calcular a distância Euclidiana (podem ser outras distâncias, mas vamos nos contentar com a Euclidiana) entre x e cada uma das amostras de treinamento, e escolher a amostra de treinamento z com a menor distância entre ela e x. A classe de x será a mesma da instância de treinamento z, pois essa é a instância mais próxima de x.

Faça um K-NN considerando $k = 3$. Geralmente vamos escolher um valor de K que é primo e diferente do número de classes no problema. Podemos considerar que $K = 3$ é um bom valor para o nosso problema em questão. Você consegue entender o motivo?

VII. Acurácia e Matriz de Confusão

Lembre-se que todas as instâncias do dataset PKLot possuem um rótulo dado por um humano. Como sabemos que esses rótulos estão corretos, chamamos isso de *Ground Truth*.

Execute seu K-NN para cada uma das instâncias de treinamento do dataset que você separou no passo da Seção IV. A partir das respostas do seu classificador, e sabendo qual a classe correta para cada instância, verifique quantas instâncias você acertou e errou. A partir disso, calcule a acurácia do seu classificador. Você conseguiu chegar em resultados similares a [1]?

Pesquise sobre o que é uma matriz de confusão, e crie a matriz de confusão para as suas amostras de testes. A partir da matriz de confusão, você consegue chegar a alguma conclusão que não conseguiria olhando apenas para a acurácia?

VIII. Conclusão

Se você conseguiu concluir todos os passos desse documento corretamente, parabéns, você conseguiu criar um sistema de machine learning do início ao fim (dê um tapinha nas suas próprias costas).

A partir daqui, você pode começar a criar sistemas com classificadores e extratores de características mais sofisticados, ou então adentrar em conceitos como o uso de Redes Neurais Convolucionais, ou aprendizado não supervisionado.

Referências

[1] P. R. De Almeida, L. S. Oliveira, A. S. Britto Jr, E. J. Silva Jr, and A. L. Koerich, "Pklot–a robust dataset for parking lot classification," *Expert Systems with Applications*, vol. 42, no. 11, pp. 4937–4949, 2015.

[2] P. R. L. de Almeida, J. H. Alves, R. S. Parpinelli, and J. P. Barddal, "A systematic review on computer vision-based parking lot management applied on public datasets," *Expert Systems with Applications*, vol. 198, p. 116731, 2022.

[3] R. Gonzalez and R. Woods, *Digital Image Processing, Global Edition*. Pearson Education, 2018. [Online]. Available: https://books.google.com.br/books?id=P8AoEAAAQBAJ

[4] C. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. Springer, 2006. [Online]. Available: https://books.google.com.br/books?id=qWPwnQEACAAJ

[5] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley, 2012. [Online]. Available: https://books.google.com.br/books?id=Br33IRC3PkQC

[6] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Elsevier Science, 2010. [Online]. Available: https://books.google.com.br/books?id=ntJQNQAACAAJ

[7] A. Kaehler and G. Bradski, *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, 2016. [Online]. Available: https://books.google.com.br/books?id=LPm3DQAAQBAJ