Bruno Severino Mascarenhas

Relatório das Práticas Laboratoriais

Visão Computacional e Reconhecimento de Padrões

Salvador

2021

Sumário

1 Prática 3 – Classificação

1.1 Objetivo

Nesta etapa, serão introduzidas as duas fases da classificação: treino e inferência. O objetivo é observar qual o desempenho da inferência do classificador após o treino com determinados parâmetros. Assim, espera-se reflexões sobre quais condições de treino o classificador apresentou melhor resultado, avaliando os aspectos referidos nas práticas anteriores.

1.2 Descrição da prática

1.2.1 Roteiro I

1. Realize o treinamento no classificador SVM com 3 modelos diferentes de classificação3, sobre os vetores de features resultantes da prática anterior.
2. Com base nos 3 modelos de classificação (1 para cada kernel), realize um leave-one-out 2-fold cross-validation (dividir a data set todo em dois grupos e usar uma hora um para treino e o outro para teste e vice-versa; não esqueça de manter nos dois grupos uma quantidade adequada de glomérulos normais e lesionados). Isso deve ser feito para cada modelo.
3. Com base no resultado das predições geradas no passo 2, calcule a acurácia média de cada classificador (média entre os dois grupos de imagens).

1.2.2 Roteiro II

1. Realize o treinamento no classificador RNA-MLP4 sobre os vetores de features resultantes da prática anterior;
2. Realize o procedimento de predição para classificação após o treinamento do passo 1, e calcule a acurácia média de cada rede.

1.3 Analise crítica dos resultados

A classificação dos dados, que consiste em dada uma nova informação, determinar a qual classe do conjunto domínio esta nova informação pertence, é a última parte do pipeline do reconhecimento de padrões em imagens. Nesta prática foram utilizados dois métodos de classificação, o *Support Vector Machine* (SVM) e o *Multi-layer Perceptron* (MLP).
O conjunto de dados contempla 811 imagens no total, 300 imagens pertencentes a classe N e 511 imagens pertencentes a classe P. Todas as imagens foram redimensionadas para o tamanho de 150 pixels x 150 pixels e foram aplicadas as técnicas de pré-processamento e extração de features descritas pelas práticas anteriores. Foi utilizada a técnica de *Leave-One-Out 2-Fold Cross-Validation* com o auxílio da função *StratifiedKFold* da biblioteca scikit-learn, a qual divide o conjunto de features e anotações passadas em instâncias para treino e teste preservando o balanceamento das classes.

1.3.1 SVM

O primeiro classificador utilizado foi o SVM, que é um algoritmo de aprendizado supervisionado utilizado para a classificação e regressão de dados. No caso do SVM cada feature é analisada como um vetor multidimensional as quais devem ser separadas, de acordo as suas respectivas classes, por hiperplanos no espaço construídos pelo algoritmo. A complexidade do SVM pode variar de $O(n^2)$ à $O(n^3)$, como demonstrado por Bottou et. al, 2006.
Dado que as classes são separáveis, o algoritmo constrói um hiperplano que separa as classes e após a construção do plano de decisão, o algoritmo determina os vetores de suporte (features) buscando maximizar a distância entre o plano e o vetor, ampliando assim a diferenciação entre as classes. Na figura 1 é ilustrado o resultado do algoritmo para o caso linear.

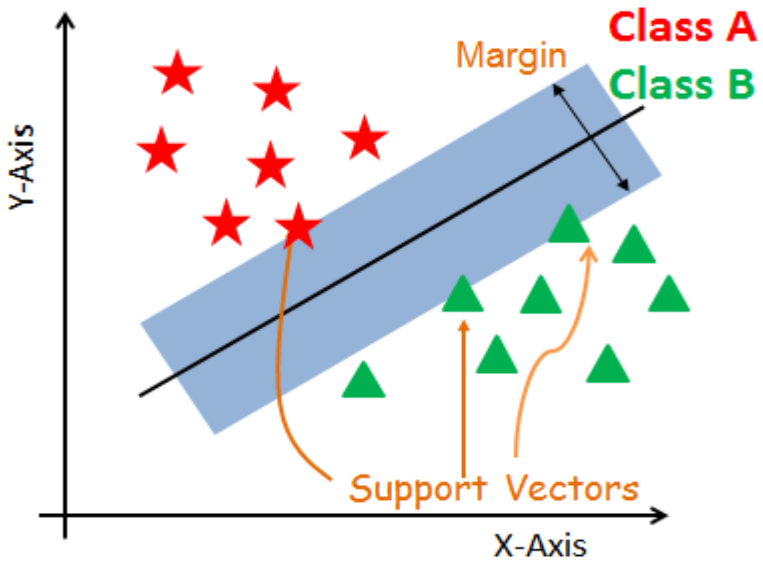

Figura 1 – Determinação do hiperplano e vetores de suporte para o classificador SVM. Imagem disponível em: https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python.

Caso as classes não sejam linearmente separáveis, é possível utilizar o truque do Kernel, em que consiste na adição de uma dimensão ao problema, tentando assim distinguir/segregar melhor o conjunto de dados. Foram os utilizados os seguintes Kernels:

- Linear (linear) – Definido como um simples produto interno entre dois vetores.
- Polinomial (poly) – Polinômio de grau estabelecido manualmente, capaz de distinguir dados não linearmente separáveis.
- Função de Base Radial (rbf) – Função que para mapear números reais de acordo à distância de um determinado centro, no caso foi implementada a função de base radial gaussiana.

A) B) C)

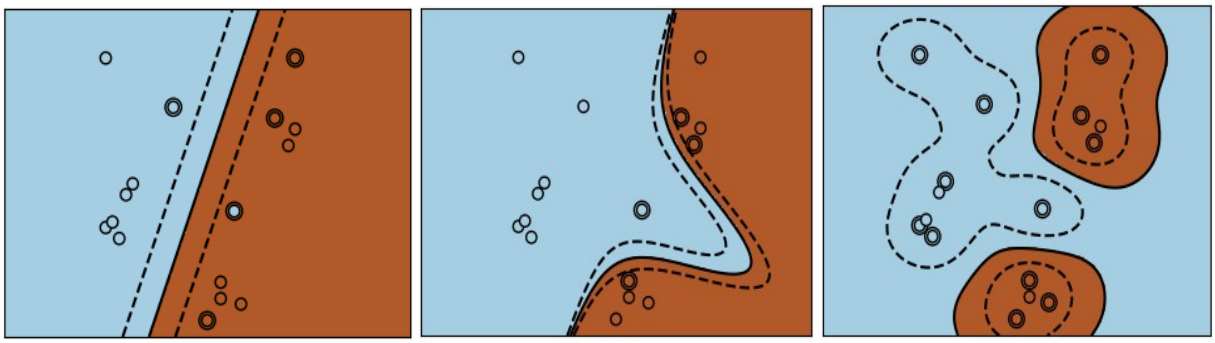

Figura 2 – Exemplo de um conjunto de dados sendo separado pelos três Kernels utilizados. A) representa o Linear, em B) Polinomial e C) Função de Base Radial. Imagem disponível em: https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html#sphx-glr-auto-examples-svm-plot-svm-kernels-py.

É possível alterar alguns hiperparâmetros do SVM para adequar o classificador ao conjunto de dados em questão. O hiperparâmetro C, também conhecido como parâmetro de regularização, é o termo que indica quanto de erro é aceitável pelo classificador. Ao diminuir o C, o SVM penalizará menos os vetores que não forem classificados corretamente ao longo do algoritmo, enquanto que aumentar o C consequentemente aumentará a penalização e o SVM classificará todos os dados com maior assertividade, porém valores altos de C tendem a causar uma condição de overfitting no conjunto de treino, tornado-se um modelo que não generaliza a sua classificação para outros conjuntos de dados. O Kernel por sua vez é outro hiperparâmetro o qual o SVM é extremamente sensível, pois afeta diretamente a

classificação a depender da distribuição do conjunto de dados (CESARSOUZA, 2010). Para o Kernel RBF, é possível alterar o valor do gamma da função gaussiana, controlando a distância no espaço em que um vetor é capaz de influenciar, para valores altos de gamma os pontos precisam estar próximos uns aos outros para serem considerados da mesma classe, enquanto para valores menores ampliam a influência do vetor no espaço. Valores altos de gamma tendem a causar overfitting no conjunto de treino, considerando apenas os grupos de vetores com muita semelhança, não generalizando o classificador. Para o Kernel polinomial é possível alterar o grau do polinômio que será utilizado no SVM.

1.3.2 MLP

O classificador Multi-Layer Perceptron (MLP) é uma rede neural artificial (RNA) de perceptrons totalmente conectados de, pelo menos, três camadas, sendo composto minimamente pela camada de entrada, uma camada oculta e a camada de saída. É amplamente utilizado pela sua alta capacidade de resolver problemas estocasticamente (solução por aproximação).

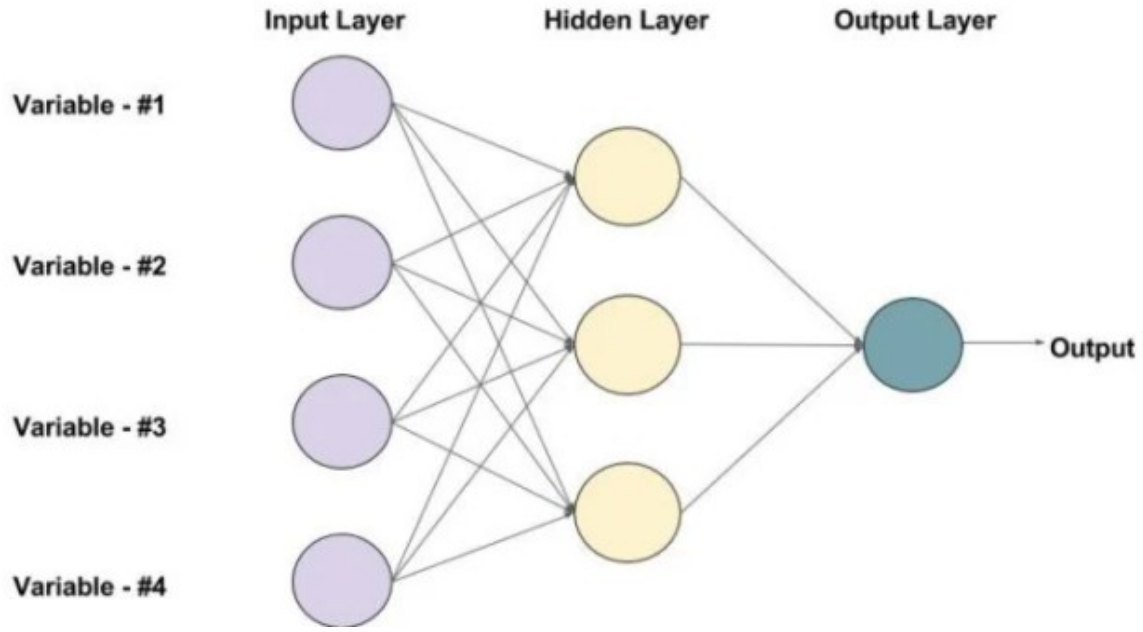

Figura 3 – Exemplo de uma rede MLP simples. Imagem extraída de https://learnopencv.com/understanding-feedforward-neural-networks/.

Cada perceptron é composto pelas entradas com os seus respectivos pesos e bias, além de uma função de ativação que é responsável para propagar para a próxima camada a sua saída. O algoritmo de aprendizado funciona de forma que os pesos de cada entrada em todas as camadas e os seus respectivos bias são ajustados a cada iteração que no caso é chamada de época, pelo algoritmo de backpropagation. O algoritmo de backpropagation busca reduzir o erro através do gradiente do erro,

calculando suas derivadas parciais e atualizando de acordo a uma taxa de aprendizado estabelecida. Foi adotada a regularização L2, a qual penaliza pesos muito altos durante a fase de treinamento (Gupta, 2021), evitando assim o overfitting.
Os dois algoritmos de backpropagation utilizados foram o ADAM e o SGD. O SGD ou Stochatisc Gradient Descent, é o algoritmo padrão o qual busca minimizar o erro através do gradiente do erro, encontrando o mínimo global. O algoritmo ADAM (Kingma, 2014) ou Adaptative Moment Estimation utiliza de duas estratégias adicionais para buscar o mínimo global do erro, a primeira estratégia é conhecida como Momentum que consiste em adicionar uma variável ao SGD com o intuito de acelerar a busca pelo mínimo e que decai ao longo do treinamento, a segunda estratégia é conhecida como RMSProp a qual é semelhante ao Momentum, porém ajuda a ajustar a taxa de aprendizado para cada parâmetro diferente, podendo aumentar ou diminuir ao longo do aprendizado.
É notável que apesar da aparente simplicidade do algoritmo do MLP, sua otimização para adequar-se ao problema é muito mais complexa em relação ao SVM devido a grande quantidade de hiperparâmetros a serem ajustados.

1.3.3 Resultados

A princípio, diversas combinações foram testadas manualmente para averiguar o comportamento em relação a acurácia média e tempo de execução das features nos classificadores SVM e MLP.

Para o algoritmo do MLP foram testados os seguintes hiperparâmetros:

- Número e tamanho das camadas ocultas: 64 com uma e duas camadas, 128 com uma e duas camadas;
- Taxa de aprendizado: 0,001, 0,0005 e 0.0001;
- Parâmetro alpha da regularização L2: 0,0001, 0,001e 0.01;
- Funções de ativação: identidade, tanh e ReLU;
- Algoritmos de backpropagation: ADAM e SGD;

Para o algoritmo SVM foram testados os seguintes hiperparâmetros:

- Parâmetro de regularização C: 0,1, 1, 10, 100
- Parâmetro gamma: 1, 0,001, ‘scale’ (1 / (num_features $\times$ variância))
- Kernel: linear, RBF, polinomial

Foi utilizada a função GridSearchCV do Sklearn para achar os melhores parâmetros, ela compila e ajusta o modelo com todas as combinações possíveis dos parâmetros passados, retornando a configuração que obteve a maior acurácia média. Além do GridSearchCV foram definidos os seguintes conjuntos de dados para uma busca exaustiva: apenas LBP, apenas GLCM, apenas o vetor de imagens (IMG), LBP + GLCM, LBP + IMG, GLCM + IMG, LBP + GLCM + IMG. As features do operador de Sobel foram retiradas após constatar que não foi obtido o aumento da acurácia em nenhum teste realizado com ela inclusa.

A configuração LBP + GLCM apresentou a maior acurácia em todos os testes realizados com o MLP e SVM, os resultados são evidenciados na Tabela 1.

Tabela 1 – Melhores resultados por configuração de dataset.

Modelo x Acurácia Média	GLCM	LBP	IMG	GLCM + LBP	GLCM + IMG	LBP + IMG	LBP + GLCM + IMG
SVM	0.86	0.75	0.84	0.92	0.86	0.83	0.85
MLP	0.84	0.74	0.84	0.86	0.76	0.80	0.79

Os melhores parâmetros encontrados para o classificador SVM foram: C igual a um, gamma igual a *scale* e com o Kernel RBF. Os melhores parâmetros encontrados para o classificador MLP foram: 128 perceptrons com uma camada, taxa de aprendizado de 0,0005, função de ativação ReLU e algoritmo de ADAM para o backpropagation.
O algoritmo SVM obteve a melhor classificação sem a utilização de data augmentation, apenas atraves da otimização dos hiperparâmetros, evidenciando a sua característica ótima de algoritmo de classificação. As features GLCM e LBP representavam apenas 0.7% das features quando incluso o vetor inteiro de imagens ((150+21)/(22500+150+21)), porém as features em conjunto obtiveram a melhor classificação em ambos os classificadores testados, evidenciado a importância da utilização de bons descritores de features.

Os testes foram realizados numa máquina com processador I7 10700, 16GB de ram 2933MH e SSD 2000MB/ leitura e gravação, tiveram uma duração de aproximadamente 36 horas para o classificador MLP e 7 horas para o classificador SVM. O significante aumento para o classificador MLP foi devido a sua maior quantidade de combinações possíveis de hiperparâmetros (432 combinações, contra 96 do SVM).

Todos os códigos desenvolvidos estão disponíveis no repositório público: https://github.com/Bruno-Mascarenhas/topicos-visao-computacional-1.

2 Referências

Bottou, L. ´., & Lin, C.-J. (2006). *Support Vector Machine Solvers*. https://leon.bottou.org/publications/pdf/lin-2006.pdf

Gupta, V. (2021, April 20). *Understanding Feedforward Neural Networks | LearnOpenCV*. LearnOpenCV – OpenCV, PyTorch, Keras, Tensorflow Examples

and Tutorials. https://learnopencv.com/understanding-feedforward-neural-networks/

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.