UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA
DEPARTAMENTO DE MATEMÁTICA APLICADA

MARCIO ROBERTO SIMÕES JÚNIOR

Uma aplicação de ensemble ponderado em classificação de imagens de texturas

Monografia apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos para obtenção de créditos na disciplina Projeto Supervisionado, sob a orientação do(a) Prof. Dr. João Batista Florindo.

Campinas
2020

1. Introdução

Após o estudo de redes neurais convolucionais e de alguns descritores de texturas para classificação de imagens médicas realizado em [1], na disciplina de MS777 do segundo semestre de 2019, para este trabalho iremos entender melhor como modelos de predição de problemas multiclasses funcionam. Utilizando classificadores binários como base e por meio do cálculo de métricas de distância entre as classes, será visto como podemos melhorar esses resultados.

Assim, generalizaremos qualquer classificador binário por meio da técnica chamada de *ensemble learning*, em que o ponderamento de *scores* nos ajuda a construir classificadores mais precisos e robustos para a classificação de nossas imagens médicas. Por fim, utilizaremos uma rede neural simples com o objetivo de combinar distâncias e *scores* e obter resultados ainda melhores.

2. Classificando imagens

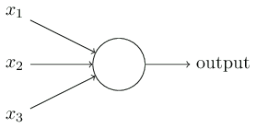
No aprendizado de máquinas, existem algumas técnicas utilizadas para a classificação de imagens. Duas delas são: redes neurais convolucionais e a classificação realizada a partir da extração de descritores de texturas.

2.1. Redes neurais convolucionais

As redes neurais artificiais tem como ideia imitar o funcionamento do nosso cérebro, onde dado um conjunto de neurônios, a unidade básica de um cérebro humano cuja a função é conduzir impulsos nervosos, uma combinação de impulsos é responsável pelo processamento de informações.

Para as redes neurais foi criado a ideia do neurônio matemático, onde dado uma entrada x_i, os pesos sinápticos w_i é capaz de controlar a intensidade de um impulso por meio de um limiar b que determinará a saída. O Perceptron é o modelo mais básico de uma rede neural, onde por exemplo, seja as entradas: x_1, x_2 e x_3, temos que:

Figura 1: Perceptron



$$y = \begin{cases} 1, & \text{se } \sum w_i x_i \geq -b \\ 0, & \text{se } \sum w_i x_i < -b \end{cases}$$

o resultado de y pode nos dizer, por exemplo, se a nossa entrada pertence ($y = 1$) ou não ($y = 0$) a uma determinada classe. Para obtermos os nossos parâmetros w_i e b, uma etapa de treinamento é necessária, onde a partir de uma função de erro e valores iniciais aleatórios podemos ir melhorando tais paramêtros.

Uma rede neural pode se tornar tão complexa quando um problema exige, através da adição de camadas ocultas e da utilização de diferentes arquiteturas. A arquitetura ao lado foi aplicada em [1] para classificação de digítos, para classificarmos imagens consideramos cada entrada sendo um *pixel* da mesma. Todo o código da aplicação pode ser encontrada aqui, para tal foi utilizado o Jupyter Notebook e a linguagem Python.

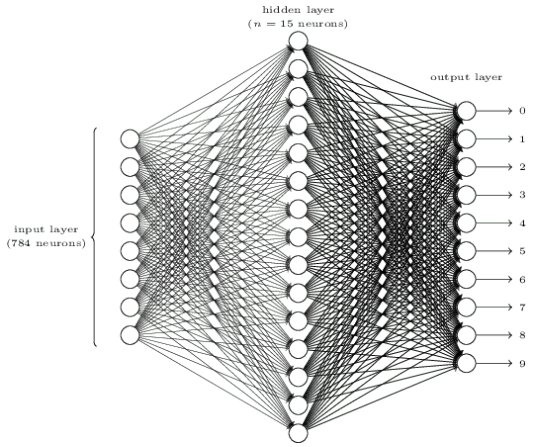


Figura 2: Exemplo de arquitetura de uma rede neural.

Uma arquitetura muito utilizada para classificação de imagens são as **redes neurais convolucionais**, essa arquitetura considera a estrutura espacial de uma imagem, além de trabalhar com diferentes canais que podem, por exemplo, representar cada espectro do RGB, e um exemplo de aplicação realizada para a classificação de digitos pode ser encontrada aqui.

2.2. Extração de descritores de textura

Já na técnica de classificação de imagens por extração de descritores de textura, certos descritores quantitativos são extraidos diretamente do nosso conjunto de imagens e modelos já bem conhecidos de *machine learning* são aplicados sob esses descritores a fim de prever a classe das nossas imagens.

Alguns desses descritores são:

- **cor**: quantificada através do histograma de cores;
- **forma da imagem**: quantificada pelos Momentos de Hu [2];
- **textura**: quantificada através dos descritores de textura de Haralick [3].

Utizamos essa segunda técnica em [1] para realizar a classificação do nosso conjunto de imagens médicas.

2.3. As imagens médicas

As imagens médicas aqui analisadas representam cistos mandibulares. Cistos são formações anormais e lesões que contêm fluido ou semifluido e são revestidos pelo tecido epitelial. O diagnóstico de cistos mandibulares costuma ser baseado em características histopatológicas e os dois tipos importantes de cistos são os *radiculares* e o *queratocisto odontogênico* (OKC).

Os cistos radiculares são os mais comuns entre os cistos odontogênicos. Já os OKCs são os menos frequentes e dentro deles existem ainda dois subtipos: os cistos solitários (esporádicos) e os cistos múltiplos (sindrômicos). No nosso conjunto de imagens, existem 1215 imagens de OKCs esporádicos (classe K), 900 imagens de cistos radiculares (classe R) e 801 imagens de OKCs múltiplos (classe S), totalizando 2916 imagens em nosso conjunto.

Os descritores utilizados para a aplicação da segunda técnica foram: cor, forma do núcleo celular e textura. A aplicação completa desde a extração dos descritores até a aplicação dos vários modelos de *machine learning* se encontra aqui. O modelo que obteve melhor resultado foi o *Random Forest* (RF) que sob o conjunto de teste atingiu 98,76% de acerto.

3. O *Ensemble Learning*

Quando tomamos uma decisão em nossa vida, levamos em consideração várias ponderações, por exemplo: quando vamos realizar uma compra pela internet, consideramos o preço, a avaliação do produto e do vendedor, a opinião de amigos e de especialistas, etc. para só assim tomarmos um decisão.

No *ensemble learning* é usada uma ideia semelhante onde os resultados das classificações são combinados a fim de obter a classe mais condizente com a nossa entrada:

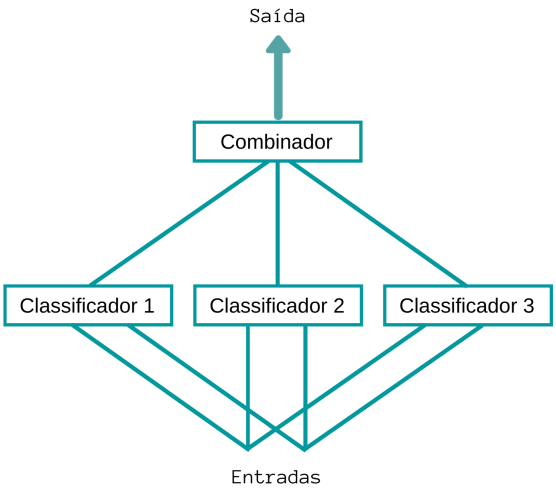


Figura 3: Exemplo de uma estrutura ensemble.

Essa combinação pode ser construída de diversas maneiras, algumas das utilizadas são: média, média ponderada e votação Máxima (3.1). Estas podem ser diretamente aplicadas sobre algum resultado numérico das

previsões dado para cada classe ou sobre algum *score* quando as classes representam objetos.

média: aqui, é simplesmente calculada a média das previsões dada pelos classificadores e o resultado dessa média será a nossa saída.

média ponderada: já aqui, cada classificador recebe um peso diferente para o cálculo da média, em que esse peso representa a relevância do classificador.

3.1. Votação Máxima

No método de Votação Máxima, as previsões dadas por cada classificador são utilizadas como **voto**, de modo que aquela com maior número de votos é a nossa saída.

Por exemplo, vamos supor que em um processo de classificação de imagens que podem pertecer às classes $\{C_1, C_2\}$, exitem 5 classificadores diferentes na nossa estrutura de *ensemble learning* e os resultados das previsões foram:

classificador 1	classificador 2	classificador 3	classificador 4	classificador 5
C_2	C_2	C_1	C_2	C_2

Assim nossa classe de saída será C_2, pois dos 5 classificadores 3 classificaram nossa entrada como C_2.

3.2. *Scores*

O *score* é um número real que possui um modo de atribuição para cada classificador e representa o quanto uma entrada está associada a uma determinada classe. Algumas vezes esse número representa a probabilidade de um objeto pertencer a uma classe dentro do conjunto total de classes.

Aqui nesse trabalho, dado um problema de classificação de n classes, chamaremos de vetor de *scores* ao vetor $R = (r_1, r_2, ..., r_n)$ tal que $r_i \in [0, 1]$ e representa o *score* de uma entrada em relação a uma classe C_i, ou seja, um valor de confiança do quanto um objeto (imagem no nosso caso) pertence a essa classe.

3.3. Classificação multiclasses

A área de aprendizado de máquinas apresenta uma série de estratégias que permitem lidar com problemas multiclasses utilizando classificadores binários (onde há apenas duas possibilidades para uma entrada dada uma classe: positiva ou negativa). Duas dessas estratégias que são as mais populares e usadas por muitos modelos são: *One-vs-All* (OvA) e *One-vs-One* (OvO).

3.3.1. *One-vs-All* (OvA)

No *One-vs-All*, um problema de classificação em multiclasses é dividido em vários problemas de classificação binária, de modo que dado um problema de n classes, são utilizados assim n classificadores binários.

Cada classificador, correspondendo a um classificador base, é responsável por distinguir uma classe C_i das demais classes, e assim, seja C_i uma classe correspondente a uma entrada, considera-se C_i como positiva e todas as demais como negativas. E uma abordagem comum utilizada para se combinar os resultados é, dado o vetor de *scores* R com $r_i \in [0, 1]$, a classe C_i de saída é tal que $i = \arg\max_{i=1,...,n} r_i$.

3.3.2. *One-vs-One* (OvO)

Já no *One-vs-One*, um problema multiclasses é dividido em $n(n-1)/2$ problemas de classificação binária, em que cada classificador, por meio de um método base, é responsável por distinguir uma classe C_i de uma outra classe C_j, e assim, é montada uma matriz de votação com os resultados de todas as classes positivas possíveis. Uma abordagem utilizada para combinar os resultados seria a utilização da Votação Máxima. Parte-se de uma matriz de *scores*

$$\mathrm{R} = \begin{pmatrix} - & r_{12} & ... & r_{1n} \\ r_{21} & - & ... & r_{2n} \\ \vdots & & & \vdots \\ r_{n1} & r_{n2} & ... & - \end{pmatrix},$$

em que $r_{ij} \in [0, 1]$ é o *score* atribuído a favor da classe C_i na distinção entre C_i e C_j, enquanto que r_{ji} é o *score* a favor da classe C_j e é tal que $r_{ji} = 1 - r_{ij}$, ou seja, a matriz R é simétrica. Logo, por Votação Máxima, temos que a classe C_i atribuida a uma entrada é tal que $i = \arg\max_{i=1,...,n} \sum_{1 \leq i \neq j \leq n} r_{ij}$.

4. *Ensemble* ponderado

Usando a técnica de *Ensemble Learning* ponderado, podemos aplicar um vetor de transformação $(\omega_1, \omega_2, ..., \omega_n)$ em nosso vetor de *scores* R a fim de melhorar os resultados, obtendo assim o vetor $R^\omega = (r_1\omega_1, r_2\omega_2, ..., r_n\omega_n)$. Logo, em OvA: $i = \arg\max_{i=1,...,n} \omega_i r_i$; e em OvO: $i = \arg\max_{i=1,...,n} \sum_{1 \leq i \neq j \leq n} \omega_{ij} r_{ij}$.

4.1. Extraindo descritores e classificando as imagens médicas

Realizaremos mais uma vez para esse trabalho o processo de extração de descritores do nosso conjunto de imagens médicas para estudarmos a melhor maneira de obtermos os pesos ponderados e aplicar essa ideia na classificação das nossas imagens.

Depois partiremos para a classificação, repetindo assim o processo de primeiramente extrair os descritores do conjunto de imagens e em seguida aplicar algum modelo de classificação.

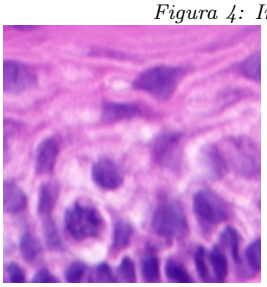
Aqui utilizaremos o classificador *Support Vector Machine* (SVM), o qual aqui usa a ideia do OvA. O conjunto total de dados será dividido meio a meio para treinamento e teste. Para a implementação, utiliza-se o Jupyter Notebook e a linguagem Python, junto com a biblioteca Scikit-learn, que já possui vários modelos de *machine learning* implementados e a biblioteca OpenCV-Python, utilizada para a extração dos descritores da imagem.

A taxa de acerto do nosso modelo foi de 87,65% e o código completo se encontra aqui.

4.2. Experimentando uma imagem teste qualquer

Selecionamos uma imagem teste qualquer do nosso conjunto teste e verificamos que ela pertence a classe S, em seguida, verificamos qual o *score* dela em relação a cada classe e é possível observar que o *score* mais alto é aquele relacionado à classe K:

Figura 4: Imagem teste - classe S



scores r_i		
K	R	S
2.2221	-0.2350	1.0943

Em seguida, selecionamos uma imagem de cada classe do nosso conjunto de treinamento e, utilizando a ideia destacada em [4], calculamos a Distância Euclidiana (4.3) entre os vetores descritores das imagens de cada classe selecionada e o vetor descritor da imagem teste, obtendo assim os resultados a seguir:

Figura 5: Imagens de treinamento e suas distâncias em relação à imagem teste.

distâncias d_i		
K	R	S
0.8177	1.4962	0.3231

Observe que se aplicamos a transformação $r_i \longrightarrow r_i/d_i$ em seu valor absoluto, obtemos como maior valor a classe S, como desejávamos. Logo, é possível aplicar uma ponderação combinando distâncias.

novos scores	K	R	S
$r_i \longrightarrow r_i/d_i$	2.7174	0.1571	3.3861

4.3. Distância Euclidiana

A distância é um resultado numérico que nos diz o quão próximo duas imagens estão (sendo uma delas pertencente a uma classe conhecida), ou seja, tem um significado geométrico intrínseco e é razoavel que seja usada como um parâmetro para ponderar nosso *ensemble*. Aqui iremos usar a distância Euclidiana, em que dados dois vetores $X = (x_1, x_2, ..., x_n)$ e $Y = (y_1, y_2, ..., y_n)$ temos que a distância $d(X, Y)$ é tal que:

$$d(X, Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2}.$$

Para as imagens, os vetores utilizados serão os próprios vetores descritores.

Outras maneiras de se calcularem distâncias são utilizadas na classificação de imagens, como a *Distância Espacial Ponderada*, que divide uma imagem em blocos e calcula a soma ponderada das distâncias entre os blocos correspondentes de cada imagem e na qual geralmente é dada maior importância ao centro da imagem; Outra maneira é por meio da *Matriz de Coocorrência* de Haralick.

4.4. Calculando os pesos

Logo, para o nosso modelo OvA, uma forma sugerida em [4] de calcular os nossos pesos ω_i seria tal que,

$$\omega_i = \left(\frac{1}{n} \sum_{j=1,...n} d_j \right) / d_i, i \neq j,$$

ou seja, é calculada a relação entre a média das demais distâncias e a distância da imagem de treinamento correspondente à imagem teste.

Isso significa que quanto mais perto está a imagem teste da sua imagem com a classe correspondente do conjunto de treinamento, mais o seu *score* será aumentado.

4.5. Aplicando o *ensemble* e generalizando para qualquer classificador

Para aplicar a técnica descrita acima em nosso conjunto, resselecionamos nossas imagens do conjunto de treinamento usadas para calcular a distância em relação à imagem teste. Para tal, calculamos a distância de cada imagem de treinamento em relação às outras imagens de treinamento pertencentes à mesma classe, e a partir da soma dessas distâncias, selecionamos a imagem de cada classe que menor se distancia de cada uma das demais imagens de sua classe.

Aplicando então o *ensemble learning* ponderado em nosso conjunto de teste usando o cálculo de pesos apresentado acima (lembrando que o código completo se encontra nesse link), obtemos uma nova taxa de acerto de 88,44%.

É possível ainda aplicar essa ideia em qualquer classificador base, e não necessariamente só em classificadores binários como apresentado, uma vez que aplicamos a transformação nos *scores* de saída de cada classe.

No código, foi aplicado também a mesma ideia em cima do classificador RF (*Random Forest*), para o qual obtemos o melhor resultado sob o conjunto teste dentre os modelos aplicados em [1] e a taxa de acerto foi de 98,76% para 98,80%.

5. Usando uma rede neural como *ensemble*

E se achássemos a melhor forma de combinar os *scores* e as distâncias por meio de uma rede neural? A ideia aqui é construir uma rede neural como na figura ao lado, que dado os *scores* e as distâncias aplique uma transformação em nossos *scores* que melhore os resultados.

Para tal, utilizamos uma rede neural simples de *perceptrons* com 6 neurônios de entrada (os 3 *scores* e as 3 distâncias), apenas uma camada oculta com 50 neurônios, função de ativação ReLu e 3 neurônios de saída que representam cada uma das nossas classes. Usamos também o classificador de

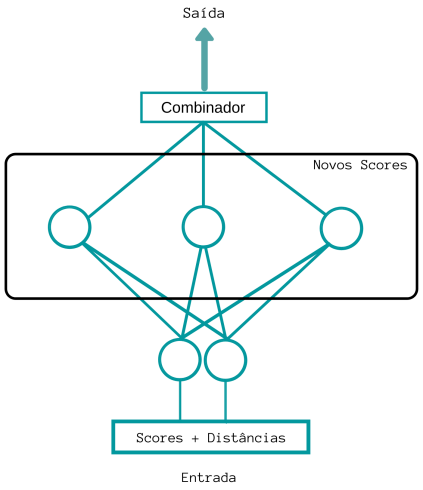


Figura 6: Exemplo de uma estrutura ensemble com rede neural.

rede neural *Multi-layer Perceptron* (MLP) existente na própria biblioteca Scikit-learn usada anteriormente.

Para a combinação SVM+Rede neural obtemos uma taxa de acerto de 88,20% e para a combinação RF+Rede Neural obtemos 99,45%, de modo que para a segunda combinação obtemos uma taxa maior do que as anteriores.

6. Resultados

Montamos em seguida uma tabela a fim de compararmos os resultados obtidos em cada combinação de classificador com cada uma das técnicas de *ensemble learning* apresentadas. Acrescentamos ainda algumas redes neurais (RNN) com mais uma camada oculta (50 neurônios).

	sem *ensemble*	**pesos calculados**	**RNN (3)**	**RNN (4)**
SVM	87,65%	88,44%	88,20%	89,50%
RF	98,76%	98,80%	99,45%	99,58%

Logo, é possível observar-se que com o *ensemble learning* conseguimos melhorar os resultados e a melhor combinação para a classificação das nossas imagens médicas foi RF+Rede Neural. E que, usando redes neurais com uma arquitetura um pouco mais robusta, obtemos ainda melhores resultados.

7. Conclusão

No primeiro estudo realizado em MS777 [1] foram usadas duas técnicas de aprendizado de máquina para a classificação de imagens: redes neurais convolucionais e a classificação realizada a partir da extração de alguns descritores de textura. Obtivemos ótimos resultados com a aplicação da segunda técnica na classificação das nossas imagens médicas.

Nesse trabalho, aprofundamos o entendimento de problemas de multiclasses e a partir da combinação de métricas de distâncias entre classes com os resultados obtidos anteriomente, conseguimos estudar formas de melhorar a técnica anterior. Assim, aplicamos juntamente a técnica do *ensemble learning*, em que utilizando-se métricas de distância foi possível obter pesos ponderados a fim de melhorar os *scores* atribuídos a cada classe antes de ser realizada de fato a determinação de uma saída.

Por fim, utilizamos nosso conhecimento em redes neurais para realizar o poderamento combinando distâncias e *scores* atráves de uma rede neural simples de *perceptrons*. Por meio da combinação de classificador e rede neural obtivemos os melhores resultados. Como utilizamos redes neurais simples com entradas e saídas do tamanho das nossas classes, essa etapa não exige um custo computacional adicional muito alto e com isso mostra-se ser um ótimo caminho para construírmos classificadores mais precisos e robustos.

Referências

[1] Redes neurais convolucionais e máquinas de vetores de suporte para classificação de imagens médicas.
URL `https://www.ime.unicamp.br/~mac/db/2019-2S-183360.pdf`

[2] Hu moments.
URL `https://en.wikipedia.org/wiki/Image_moment`

[3] Haralick texture.
URL `http://haralick.org/journals/TexturalFeatures.pdf`

[4] E. B. H. B. "J.I. Forcén", "M. Pagola", Combination of features through weighted ensembles for image classification, Applied Soft Computing Journal 84. `doi:https://doi.org/10.1016/j.asoc.2019.105698`.