Predizendo o modelo da câmera de uma fotografia a partir de técnicas de aprendizado de máquina

Gustavo Ciotto Pinton, RA117136*

Abstract

São propostos neste documento dois métodos de classificação baseados em aprendizado de máquina e capazes de determinar, a partir de uma foto, o modelo da câmera que a gerou. No total, 10 modelos de câmeras distintas, mas de mesmos fabricantes, são utilizados neste problema. O primeiro, baseado na técnica de regressão logística, consiste em separar as 10 classes em grupos binários de maneira a testar sempre um dos modelos contra os demais, enquanto que o segundo, baseado em redes neurais, busca diretamente a classe de determinada imagem, sem quaisquer divisão binária dos grupos. Além disso, neste artigo, estão descritos os atributos recuperados das imagens e os parâmetros configurados em cada um dos métodos, bem como a sua influência no resultado final. Por fim, são apresentados os resultados de cada uma das técnicas e o score obtido para o problema na plataforma Kaggle.

1. Introdução

A técnica de regressão logistíca, ao contrário do método de regressão linear cujo objetivo é adequar os dados a uma equação linear, busca classificar os dados de entrada em dois grupos discretos, comumente denominados 0 e 1. Para tal, define-se a matriz $\boldsymbol{x}$ de dimensões $N+1 \times M$ contendo M entradas descritas por $N+1$ atributos (contando evidentemente um elemento de *bias*), o vetor $\boldsymbol{\theta}$ representando o peso que cada atributo terá no modelo e a uma função $g\left(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}\right)$ cujo conjunto imagem é o intervalo $[0, 1]$. Esta última é responsável por predizer a classe $\hat{\boldsymbol{y}}^{(i)}$ de cada entrada $\boldsymbol{x}^{(i)}$, tendo em vista que valores $\hat{\boldsymbol{y}}^{(i)}$ mais próximos de 0 indicarão que determinada entrada tem mais chance de pertencer à classe 0 e vice-versa. A função g, a princípio, pode assumir qualquer equação desde que sua imagem fique entre 0 e 1. Para este documento, tal função assumirá a forma sigmoidal, conforme equação 1.

$$\hat{\boldsymbol{y}}^{(i)} = g\left(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}\right) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}}} \tag{1}$$

Neste caso, se $\theta^T \boldsymbol{x}^{(i)}$ é grande, então $e^{-\theta^T \boldsymbol{x}^{(i)}}$ tende a zero e a classe atribuida à entrada i será 1. Em oposição, se $\theta^T \boldsymbol{x}^{(i)}$ é muito negativo, a função tenderá a 0, assim como a classe escolhida.

Assim como quase toda técnica de aprendizado de máquina, a determinação dos valores do vetor $\boldsymbol{\theta}$ envolve a minimização de uma função de custo $J(\boldsymbol{\theta}, \boldsymbol{x})$, descrita pela equação 2, a partir de um conjunto de entradas $\boldsymbol{x}^{(i)}$ de saídas $\boldsymbol{y}^{(i)}$ conhecidas.

$$J(\boldsymbol{\theta}, \boldsymbol{x}) = -\frac{1}{M} \sum_{i=1}^{M} \Big[\boldsymbol{y}^{(i)} \log\left(\hat{\boldsymbol{y}}^{(i)}\right) + \dots + \left(1 - \boldsymbol{y}^{(i)}\right) \log\left(1 - \hat{\boldsymbol{y}}^{(i)}\right) \Big] \tag{2}$$

Como visto em aula, a função $J(\boldsymbol{\theta}, \boldsymbol{x})$ é convexa e, portanto, não possui mínimos locais além do global e sua minimização pode ser realizada também através do método de *gradient descent*, explorada na última atividade. Os dados utilizados para a minimização $\boldsymbol{x}^{(i)}$ e $\boldsymbol{y}^{(i)}$ pertencem a um conjunto denominado **conjunto de treinamento**, enquanto que a validação dos parâmetros obtidos ocorre em um conjunto que recebe este mesmo nome. Um regra geral é separar 80% dos dados disponíveis para treinamento e o restante, 20%, para testes e validação.

Derivando-se parcialmente a equação 2 em relação a $\boldsymbol{\theta}_j$, obtém-se o coeficiente linear e portanto a direção que devemos tomar para nos aproximar do mínimo global. Repetindo o raciocínio para todos os parâmetros $\boldsymbol{\theta}_j$, obtém-se à operação 3 que deve ser aplicada a cada um deles a cada iteração. Observa-se que, apesar de que o cálculo de 1 ser totalmente diferente da aproximação realizada pelo modelo linear, isto é, $\hat{\boldsymbol{y}}_{lr}^{(i)} = \boldsymbol{\theta}^T \boldsymbol{x}^{(i)}$, obtemos exatamente a mesma equação para cada iteração, conforme equação 3. Assim como explorado na atividade passada, α é chamado de *learning rate* e controla a velocidade de convergência ao mínimo global.

$$\boldsymbol{\theta}_j = \boldsymbol{\theta}_j - \alpha \frac{\partial J}{\partial \boldsymbol{\theta}_j} = \boldsymbol{\theta}_j - \frac{\alpha}{M} \sum_{i=1}^{M} \left(\hat{\boldsymbol{y}}^{(i)} - \boldsymbol{y}^{(i)}\right) \boldsymbol{x}_j^{(i)} \tag{3}$$

É importante observar que, para problemas com mais de duas classes, é necessário adotar estratégias para o uso

*Is with the Institute of Computing, University of Campinas (Unicamp). **Contact**: gustavociotto@gmail.com

da regressão logística, tendo em vista que ela é capaz de modelar apenas problemas *binários*. Sendo assim, exploramos duas estratégias durante as aulas. A primeira, chamada de *one vs. all* consiste em obter n modelos com n igual o número de classes e, para cada um, calcular $\hat{\boldsymbol{y}}_j^{(i)}$. A classe j escolhida será aquela com maior $\hat{\boldsymbol{y}}_j^{(i)}$. Na segunda estratégia, denominada *many vs. many*, seleciona-se aleatoriamente grupos de classes e calcula-se um preditor para cada um deles. A determinação da classe ocorre da mesma maneira, isto é, a partir da verificação da maior probalidade. Para este problema, entretanto, escolhe-se a primeira estratégia.

Apesar da relativa facilidade que a regressão logística oferece, problemas que possuem muitos atributos de entrada, tais como o apresentado nesse documento, tendem a não ser bem modelados por essa técnica, devido ao fato de que $\boldsymbol{\theta}^T\boldsymbol{x}$ permanece sendo uma relação linear. Pode-se argumentar que é possível gerarmos mais atributos de ordens superiores, porém isso nos conduziria a outros problemas, tais como aumento de complexidade e *overfitting*. É neste cenário que a segunda técnica, baseada em redes neurais, torna-se importante, à medida que é bem-sucedida em capturar as não-linearidades do modelo. As redes neurais são compostas, dependendo de sua arquitetura, de diversas unidades denominadas neurônios, que podem ser enxergadas individualmente como regressores logísticos, organizadas em camadas interligadas entre si. A cada neurônio da camada i, são ligadas $n_{i-1}+1$ entradas provindas da camada $i-1$, multiplicadas pelo vetor de pesos $\boldsymbol{\theta}_{(i-1)}$, gerando um saída a_i, denominada ativação, conforme equação 4. Uma rede neural possui, no mínimo, duas camadas, sendo elas as de entrada e a de saída. As camadas intermediárias, por sua vez, recebem o nome de *hidden layers*.

$$a_i = g\left(\boldsymbol{\theta}_{(i-1)}, \boldsymbol{x}_{(i-1)}\right) = \frac{1}{1+e^{-\boldsymbol{\theta}_{(i-1)}^T\boldsymbol{x}_{(i-1)}}} \tag{4}$$

A determinação dos parâmetros $\boldsymbol{\theta}_{(i-1)}$ envolve, assim como no caso da regressão logística, a minimização de um função de custo J. A técnica de *gradient descent* poderia ser evidentemente utilizada, porém ela exigiria muito poder computacional dado a maior complexidade do modelo e a quantidade de parâmetros a ser determinados. Desse modo, uma outra técnica, chamada de *backpropagation* se mostra interessante, à medida que é capaz de estimar o valor da derivada parcial da função J em relação a cada $\boldsymbol{\theta}_{(i-1)}$ a partir da acumulação dos erros obtidos indo-se da camada $l+1$ à l.

Para o nosso problema, propõe-se o uso de apenas uma camada intermediária e 10 neurônios da camada de saída, cada um gerando a probabilidade de determinada imagem pertencer a uma das 10 classes.

Tendo visto duas ferramentas de classificação, podemos aplicá-las a dados reais. Neste relatório, usamos dados de uma competição publicada na plataforma *Kaggle*, em que os competitores foram desafiados a determinar o modelo da câmera que gerou determinada fotografia. Em outras palavras, busca-se dividir o conjunto de entrada em 10 classes distintas, cada uma representando um modelo de câmera distinto. No total, 275 fotografias de cada modelo foram disponibilizados para treinamento dos classificadores. Nas próximas seções, serão discutidos o processo de aquisição de atributos das imagens, o processo de treinamento e os resultados obtidos.

2. Atividades

As próximas subseções visam explicar as escolhas dos diversos parâmetros adotados pelo autor.

2.1. Cálculo dos atributos

Durante as aulas, discutimos que cada modelo de câmera utiliza seu próprio método de interpolação para cada *pixel* constituindo uma imagem. Em outras palavras, esta informação pode ser vista como a assinatura do equipamento na fotografia. Vimos também que a maneira mais correta de obtermos tal *assinatura* é através do ruído de cada uma das bandas *RGB*, isto é, aplicamos um filtro f à imagem I e subtraímos o resultado da imagem original, conforme equação 5.

$$N = I - f(I) \tag{5}$$

Alguns trabalhos [1][2] na área apontam que o uso de *discrete wavelet transform* produz melhores resultados na identificação, isto é, os atributos gerados são capazes de traduzir com mais pertinência e significância a *assinatura* em cada fotografia. Baseando-se nestes dois trabalhos citados, foram calculados para cada imagem os seguintes atributos:

- Conforme [1], **média**, **variância**, **skewness** e **kurtosis** das direções horizontal, vertical e diagonal de cada uma das quatro escalas da *discrete wavelet transform* aplicadas em cada uma das bandas *RGB*, resultando, no total, em 144 atributos.

- Ainda de acordo com [1], estatísticas da matriz de *co-ocurrence* também foram calculadas. Para as direções horizontal, vertical e diagonal de cada uma das quatro escalas da *discrete wavelet transform* em cada uma das bandas *RGB* e quatro ângulos diferentes, calculou-se a **energia**, **entropia**, **contraste**, **homogeneidade** e **correlação**, resultado, assim, em mais 720 atributos.

- O processo de obtenção de atributos de [2] foi reproduzido, isto é, foram calculados 9 momentos centrais para as três direções de cada uma das quatro escalas da *discrete wavelet transform* aplicada a N. N foi calculada da mesma maneira que representada em [2]. Nesta etapa, gera-se 324 atributos adicionais.

No total, gera-se, portanto, 1188 atributos para cada imagem.

2.2. Ajuste o parâmetro α da regressão logística

Para a técnica de *gradient descent*, a estratégia de modificação do parâmetro α (*learning rate*) é muito importante para impedir eventuais balanços em torno do mínimo global. Como utilizamos uma solução já pronta para a regressão logística, usamos uma das opções disponibilizadas pela biblioteca para tal estratégia. Utilizou-se, assim, aquela denominada *AdaGrad* [3], ou *adaptive gradient algorithm*, em que diferentes taxas de aprendizado são utilizadas de acordo com os atributos. Alguns atributos podem se mostrar úteis para um problema de otimização, mas podem não aparecer na maioria das entradas de treinamento. Se, quando aparecem, eles são tratados da mesma forma em termos de taxa de aprendizado em relação a um recurso que apareceu centenas de vezes, estamos considerando que a influência de tais atributos não significa nada na otimização geral. A fim de combater este efeito, a estratégia *AdaGrad* faz com que atributos que são mais esparsos nos dados tenham uma taxa de aprendizado maior, o que se traduz, desse modo, em uma atualização maior para esse determinado atributo.

2.3. Regularização

A fim de evitar efeitos indesejados gerados devido ao *overfitting*, consideramos para ambas as técnicas o uso de um parâmetro de regularização. Assim como destacado no primeiro exercício, esse parâmetro penaliza os pesos $\boldsymbol{\theta}_j$ impedindo que o modelo gerado apresente pequena variação com as entradas utilizadas no treinamento. Foi determinado experimentalmente o valor de $\lambda = 10$ para ambas as técnicas.

2.4. Arquitetura da rede neural

A rede neural utilizada foi constituída apenas de três camadas: a de entrada com 1188 atributos, a de saída com 10 neurônios, um para cada classe, e apenas uma intermediária com o número de neurônios igual aquele de atributos, isto é, 1188. Esta escolha foi adotada a partir de testes com os conjuntos de treinamento e do fato de que, nesta atividade, não foi visado o uso de *deep neural networks*, isto é, redes com muitas camadas intermediárias. Para cada neurônio, a função *sigmóide* foi utilizada como função de ativação.

3. Soluções propostas

Essa seção é dedicada à discussão das soluções propostas. Foram realizados testes com regressores logistícos e redes neurais. Em todos os casos, usou-se 80% do conjunto de imagens para treinamento dos modelos e 20% para validação.

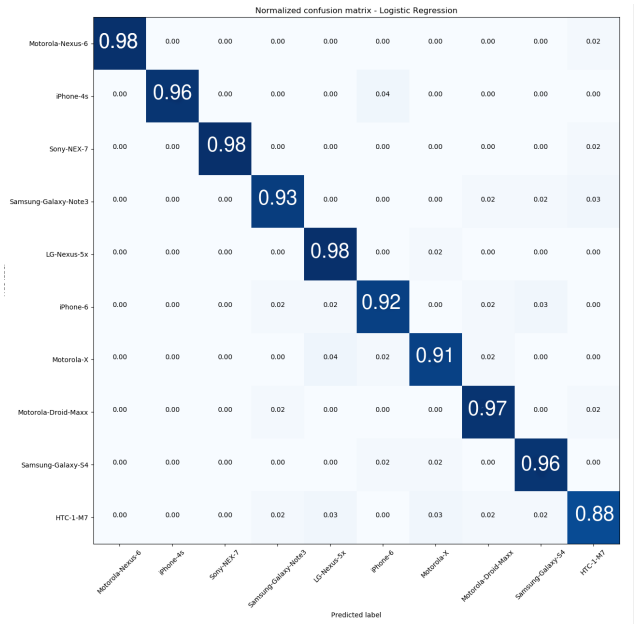

Figura 1. Matriz de confusão para a regressão logística. Quanto mais escura uma célula, mais próximo de 1.00 seu conteúdo é.

3.1. Conjunto de treinamento original

Nesta subseção, serão descritos os resultados obtidos para o conjunto de treinamento de imagens original, isto é, sem quaisquer adições a tal conjunto.

3.1.1 Regressão logística

Conforme comentado na seção 1, utilizamos a estratégia de *one vs. all* para a classificação das classes. A figura 1 representa a matriz de confusão resultante do conjunto de validação. Em média, os regressores foram bem sucedidos em predizer a classe de 94.7% das imagens do conjunto de validação. O modelo que obteve o maior número de erros foi o *HTC 1 M7*. Utilizando o modelo com o conjunto de testes, que é constituído por imagens que foram recortadas e transformadas, tal média caiu para apenas 20.22%, indicando que o modelo não é capaz de predizer corretamente o modelo de imagens que sofreram algum tipo de alteração.

3.1.2 Rede neural

A figura , por sua vez, reproduz a matriz de confusão para a rede neural. Neste caso, obtém-se um resultado ligeiramente inferior àquele obtido para a regressão logistíca: em média, apenas 81.4% foram classificadas corretamente. Para quatro modelos, o índice de acerto foi inferior a 80%, sendo que o pior foi o *Samsung Galaxy S4*, em que um quinto das imagens de validação foram classificadas corretamente. Para a submissão na plataforma *Kaggle*, obteve-se um resultado também inferior, isto é, 17.8% de acerto.

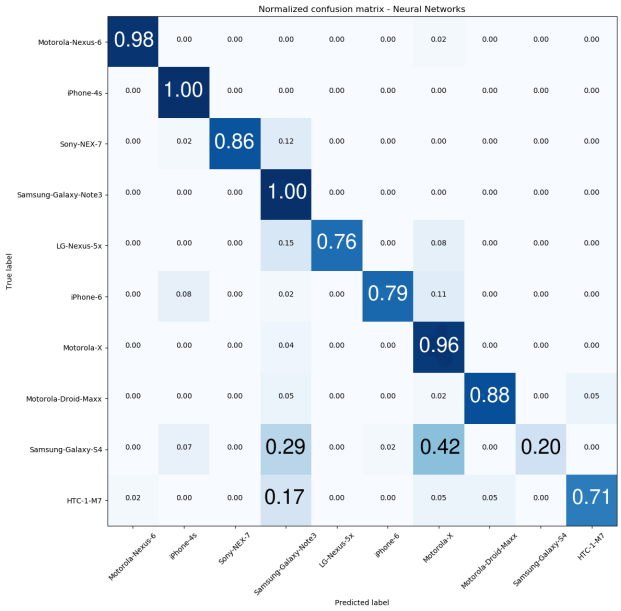

Figura 2. Matriz de confusão para a rede neural. Quanto mais escura uma célula, mais próximo de 1.00 seu conteúdo é.

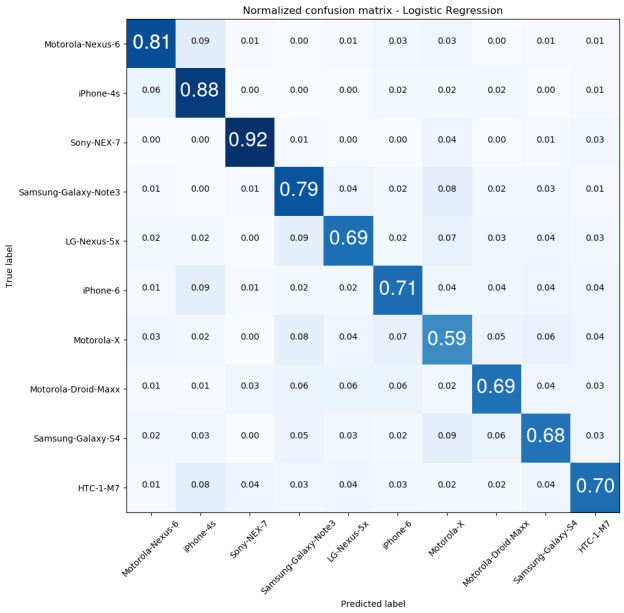

Figura 3. Matriz de confusão para a regressão logística para conjunto de treinamento aumentado. Quanto mais escura uma célula, mais próximo de 1.00 seu conteúdo é.

3.2. Conjunto de treinamento aumentado

Tendo em vista os resultados pobres obtidos nas submissões na plataforma *Kaggle*, decidiu-se aumentar o conjunto de imagens de treinamento para incluir imagens de 512x512 recortadas a partir do centro das originais e corregidas em *gamma* com valores de 0.8 e 1.2. Espera-se, portanto, que o índice de acerto para as imagens do conjunto de teste aumente.

3.2.1 Regressão logística

A figura 3 representa a matriz de confusão para a técnica de regressão logística com o conjunto de treinamento aumentado. Neste caso, a média de acerto para as imagens do conjunto de validação cai para 74.7%, porém a taxa de acerto para o conjunto de testes quase dobra, passando de 20.22% para 37.4%. Isto indica que, apesar dos resultados inferiores no conjunto de validação, o modelo está um pouco mais abrangente.

3.2.2 Rede neural

A rede neural proposta submetida aos novos dados de treinamento produziu uma taxa de acerto, em média, de 52.3% para o conjunto de validação. Assim como na regressão logística, a acurária obtida na validação foi inferior àquela do caso que utilizou o conjunto de treinamento original, porém a taxa de acerto para o conjunto de teste melhorou, indo de 17.8% para 25.8%. Assim como no caso anterior, o modelo tornou-se ligeiramente mais abrangente.

4. Conclusões

Tanto a técnica de regressão logística quanto a de redes neurais obtiveram bons resultados, isto é, acima de 80% de acerto em média, na classificação das imagens do conjunto de treinamento, porém baixos índices para as imagens modificadas do conjunto de teste. Quando aumentou-se o conjunto de treinamento para considerar algumas das características das imagens presentes no de teste, melhoramos a taxa de acerto em praticamente duas vezes, indicando, portanto, que se quiséssemos aumentá-la ainda mais, teríamos que considerar todas as transformações realizadas no conjunto de teste para o treinamento dos modelos propostos.

Referências

[1] B. Wang, Y. Guo, X. Kong, and F. Meng. Source camera identification forensics based on wavelet features. In *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 702–705, Sept 2009. 2

[2] J. R. Corripio, D. M. A. González, A. L. S. Orozco, L. J. G. Villalba, J. Hernandez-Castro, and S. J. Gibson. Source smartphone identification using sensor pattern noise and wavelet transform. In *5th International Conference on Imaging for Crime Detection and Prevention (ICDP 2013)*, pages 1–6, Dec 2013. 2

[3] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011. 3