Redes Artificiais de Arquitetura Variante

LUCAS O. DAVID *
Ciência da Computação - Pós-Graduação
*E-mail: *ld492@drexel.edu*

***Abstract* – apesar do grande desenvolvimento de redes neurais artificiais aplicadas a problemas cada vez mais complexos, pouco foi feito no que diz respeito à otimização da arquitetura dos modelos utilizados. Este trabalho propoe que esta otimização ocorra pela busca sobre o espaço de arquiteturas possíveis, percorrendo-o de forma a alcançar um candidato à solução que seja tanto eficaz na generalização do conjunto de dados em mãos quando enxuto em relação ao seu número de parâmetros, caracterizando assim um problema de busca clássico. Resultados empíricos preliminares sugerem que tal busca resulta em refinamentos interessantes das arquiteturas de redes.**

***Palavras-chave* – Aprendizado de Máquina, Redes Neurais Artificiais.**

I. INTRODUÇÃO

Redes neurais artificiais têm recorrentemente apresentado efetivas soluções para problemas de aprendizado de máquina. Elas podem ser, entretanto, computacionalmente custosas. Com um alto número de parâmetros causado por uma quantidade significativa de camadas e unidades, tais redes exigem alto desempenho computacional, *hardware* específico (e.g. GPUs) e longos períodos de trainamento. Na última decada, também é notável uma aumento do emprego de redes convolucionais em problemas de tratamento e interpretação de sinais (e.g. audio, video, texto), característicamente profundas e compostas por um número massivo de parâmetros.

Um problema imediato ao decidir se utilizar de redes neurais artificiais é encontrar uma arquitetura adequada ao problema em questão. Devido a grande dificuldade deste problema, muitos autores recorrem à reutilização de arquiteturas já existentes, que tiveram seu bom comportamento demonstrado por experimentos empíricos em domínios de problema similares.

Contrariamente à escolha de arquiteturas tradicionais, é possível buscar uma arquitetura através da exploração do espaço de escolhas possíveis, preferindo por aquelas que adequadamente abordam o problema, enquanto minimizando sua complexidade. Como o número de arquiteturas é possívelmente infinito, percorrer o espaço eficientemente é de fundamental importância.

Este trabalho tem como principal objetivo a modelação do problema de otimização de arquitetura como um problema de busca, podendo ser portanto resolvido por simples agentes inteligentes baseados em utilidade e algoritmos de busca clássicos (e.g. *Hill-Climbing* e algoritmos genéticos).

O restante deste relatório está organizado da seguinte forma: na Seção II são apresentados conceitos importantes no entendimento do trabalho; o trabalho proposto, bem como o modelo utilizado para a tarefa são descritos na Seção III. Resultados de experimentos são discutidos na Seção IV. Finalmente, as considerações finais são feitas na Seção V.

II. CONCEITOS FUNDAMENTAIS

Nesta sessão, conceitos chave para o entendimento do trabalho proposto serão brevemente enunciados. Tais conceitos vão desde uma breve descrição das representações numéricas de redes neurais artificias ao conceito de busca

A. Redes Neurais Artificiais

Uma rede neural artificial é um modelo de aprendizado de máquina inspirado na estrutura de uma rede neural biológica [1]. Associada à uma arquitetura específica, uma rede neural se utiliza de seus parâmetros para transformar um sinal de entrada em outro. Por exemplo, um sinal de entrada pode ser transformado em um de menor dimensionalidade (redução dimensional); gerar probabilidades de pertencimento à múltiplicas classes (classificação); ou regredir um ou mais valores contínuos (regressão). Fig. 1 exemplifica abstratamente uma rede neural, composta por três camadas com 1, N e 3 unidades, respectivamente.

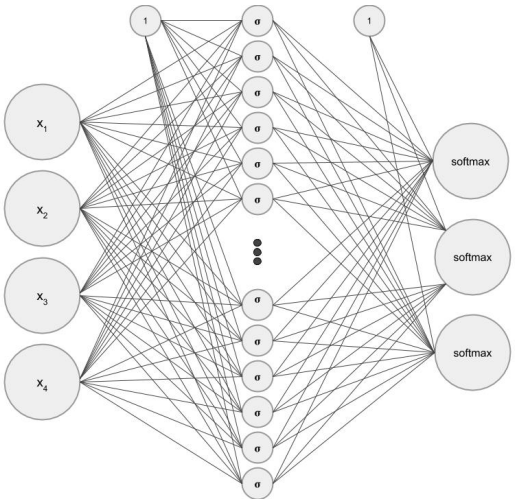

Figure 1. Ilustração de uma rede neural com três camadas, com função de ativação σ na segunda camada e $softmax$ como função de ativação na terceira.

Uma rede neural também pode ser definida algebricamente, onde sua saída é um y em função do sinal de entrada e de seus parâmetros θ.

1) Redes Densas: também conhecidas como *Multilayer Perceptron* (MLPs), são redes caracterizadas pela total conexão de todos os neurônios da camada i para os da camada $i-1$ e os da camada $i+1$. A propagação do sinal (i.e. a saída da rede $\boldsymbol{y}$) pode ser representada na seguinte forma algébrica:

Seja $\boldsymbol{X} \in \mathbb{R}^{n \times f}$ um conjunto de n amostras descritas por f atributos, $\theta = (\boldsymbol{W}, \boldsymbol{b})$ os parâmetros da rede e σ uma função denominada "ativação". Então,

$$\boldsymbol{y}(\boldsymbol{X}) = \boldsymbol{\sigma}(\boldsymbol{W} \cdot \boldsymbol{X} + \boldsymbol{b})$$

2) Redes Convolucionais: baseadas especificamente no cortex visual, essas redes propagam o sinal através da operação de convolução entre o sinal de entrada e *kernels* (descritores basicos de padrões), sendo ideais para identificar padrões locais [2]. Ademais, a sucessiva concatenação de convoluções, eventualmente intercaladas por camadas de *pooling* (ou subamostragem), é capaz de construír efetivos reconhecedores para padrões complexos, como objetos e faces.

Uma camada convolucional de uma rede convolucional pode ser descrita pela seguinte equação:

$$(\mathbf{I} * \mathbf{K})(i,j) = \sum_{k=0}^{h} \sum_{l=0}^{w} \mathbf{I}(i-k, j-l)\mathbf{K}(k,l)$$
$$y_{i,j} = \mathbf{I} * \mathbf{K}(i,j)$$

Ou seja, a própria convolução discreta; limitada inferiormente e superiormente; e definida sobre tensores de rank 2 (matrizes).

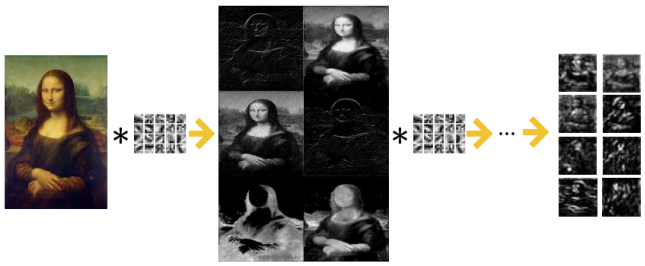

Figure 2. Ilustração de um sinal de entrada (uma pintura) sendo propagado por uma rede convolucional – a arquitetura VGG19 [3], treinada sobre o conjunto de dados ImageNet [4].

Finalmente, é importante destacar a utilizando de redes dense concatenadas à convolucionais, criando efetivos modelos de classificação/regressão sobre amostras representadas sinais de alta dimensionalidade (e.g. imagens, sons, texto) [2], [3].

B. Intelligência Artificial Clássica

1) Mundo: a modelação de um problema computacionalmente descritível. Por exemplo, o caso de estudo clássico da **limpeza de um ambiente** sujos pelo agente de limpeza automatizado [5].

2) Estado: um objeto representativo de condições possíveis para o mundo. Por exemplo, o vetor $(1,1,1,1,0)$ pode representar uma condição do problema de **limpeza de ambiente** onde os quatro primeiros elementos representam se um ambiente está ou não sujo, enquanto o último sinaliza a posição corrente do agente.

3) Agente: uma entidade inserida no mundo, que o percebe com **sensores** e atua sobre ele através de **atuadores** [5]. Dentre os inúmeros tipos de agentes, destaca-se o agente **baseado em utilidade**, que associa a cada estado um valor real e, caso seja um agente racional, buscará estados que maximizem esse valor.

4) Espaço de busca: um grafo direcionado que indica como os estados do mundo se relacionam.

5) Busca: A atividade de navegar pelo espaço de possíveis estados de um mundo a fim de se encontrar um estado objetivo ou o caminho para ele. Fig. 3 exemplifica a **busca em largura** sobre um espaço qualquer.

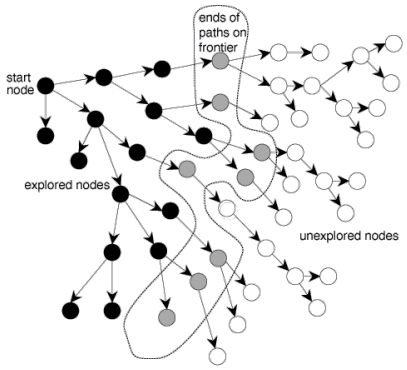

Figure 3. Ilustração da execução de uma **busca em largura**, em um espaço de busca para um problema qualquer.

Dentre as diversas buscas existentes, destacamos duas:

a) Hill-Climbing (exemplo de busca): um tipo de busca local onde estamos preocupados em percorrer o espaço de forma a maximizar a utilidade do estado atual para o agente. A Fig. 4 exemplifica essa busca em um espaço contínuo, de uma única variável.

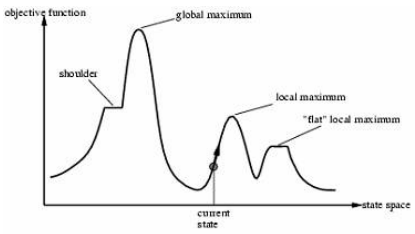

Figure 4. *Hill-Climbing* sobre um espaço contínuo.

Naturalmente, *Hill-Climbing* não garante o obtenção de máximo global. Entretanto, uma forma simples de fortificar as respostas seria reiniciar a busca multiplas vezes a partir de um ponto aleatório (uma estratégia conhecida como *random-restart*).

b) Algoritmos Genéticos (exemplo de busca): um modelo de busca inspirado na teoria de seleção natural de Darwin [6]. Aqui, uma população de estados é iterativamente submetida a operadores de criação (e.g. *cross-over*, mutação) e seleção de tal forma a reforçar positivamente a sobrevivência dos estados com alto valor de utilidade. A tabela I exibe o operador de *cross-over* com um único ponto de corte (gene 2) sobre dois indivíduos A e B, resultando em dois novos indivíduos (C e D).

indivíduo A	0	0	1	1	0	1
indivíduo B	1	1	0	1	0	0
indivíduo C	0	0	0	1	0	0
indivíduo D	1	1	1	1	0	1

Table I
A OPERAÇÃO DE *cross-over* ENTRE DOIS INDIVÍDUOS (A E B).

III. Trabalho Proposto

Nesta seção, será apresentado uma abordagem para o problema de seleção de arquiteturas.

A. Definição do problema

Seja $(\boldsymbol{X}, \boldsymbol{y})$ um conjunto de dados pré-determinado, o mundo é definido como o problema de encontrar a arquitetura de rede neural capaz de generalizar adequadamente o conjunto $(\boldsymbol{X}, \boldsymbol{y})$, ao passo que possui o menor número de parâmetros possíveis (e.g. camadas e unidades nas camadas).

B. Arquiteturas representadas por estados do mundo

Define-se uma representação que descreva possíveis arquiteturas para as redes. Tal representação se constitui por duas coleções de elementos, onde cada coleção contém camadas descritas por pares (# unidades/kernels, função de ativação), como exemplificado pela listagem 1.

```
S = {
  'dense': ((4096, 'relu'), (10, 'softmax'))
  'conv': ((32, 'relu'), (128, 'relu'))
}
```

Listing 1. Representação de uma arquitetura de rede neural artificial como um estado no espaço de busca.

C. Transições entre arquiteturas

O agente alcança novas arquiteturas ao aplicar os operadores descritos abaixo à uma ou mais arquiteturas já existentes. Da lista, os três primeiros são necessárias à busca utilizando algoritmos genéticos, enquanto as demais são essenciais à buscas locais, como o *Hill-climbing*.

1) **Random:** produz um arquitetura aleatória.
2) **Cross-over:** combina duas arquiteturas, resultando em uma terceira.
3) **Mutate:** produz uma mudança de intensidade q com probabilidade p em cada camada existente.
4) **Reduce conv:** reduz o número de camadas convolucionais.
5) **Reduce dense:** reduz o número de camadas densas.
6) **Reduce kernels:** reduz para $\frac{2}{3}$ o número de *kernels* na última camada convolucional.
7) **Reduce units:** reduz para $\frac{2}{3}$ o número de unidades na última camada densa.
8) **Increase conv:** aumenta o número de camadas convolucionais.
9) **Increase dense:** aumenta o número de camadas densas.
10) **Increase kernels:** aumenta para $\frac{3}{2}$ o número de *kernels* na última camada convolucional.
11) **Increase units:** aumenta para $\frac{3}{2}$ o número de unidades na última camada densa.

D. Utilidade

A fim de direcionar a busca à resultados satisfatórios, é necessário a definição de uma função de utilidade que beneficie "boas" arquiteturas e prejudique arquiteturas "ruins". Esta função de utilidade u é definida da seguinte forma:

$$
\begin{aligned}
u(S) &= -h(S) \\
h(S) &= \lambda_1 \sum_{i=1}^{|\text{layers}|} \frac{\text{units}_i - \text{min-units}}{\text{max-units} - \text{min-units}} \\
&\quad + \lambda_2 \frac{\text{layers} - \text{min-layers}}{\text{max-layers} - \text{min-layers}} \\
&\quad + \lambda_3 \text{train-loss} \\
&\quad + \lambda_4 \text{validation-loss}
\end{aligned}
$$

Onde $\lambda_1..\lambda_4$ são constantes definidas pelo usuário, ponderando a importância de cada métrica para a função de custo h.

A utilidade de uma arquitetura pode ser interpretada da seguinte forma: uma arquitetura (estado) terá mais alta utilidade quando apresentar mais baixo número de unidades e *kernels*, mais baixo número de camadas e mais baixas perdas de treino e validação.

IV. Resultados e Discussão

A busca por arquiteturas foi experimentada em dois conjuntos diferentes. Nesta seção, serão descritos os resultados de tais experimentos.

A. Conjunto de dados *Digits*

Contendo imagens em escala cinza de 10 diferentes dígitos, **Digits** tem como tarefa associada a correta classificação de uma amostra em 10 classes distintas.

Utilizando uma população de 50 arquiteturas, com probabilidade de mutação $p = .25$ e fator de mutação $q = .5$ e mantendo o processo evolutivo por 5 gerações, obteve-se uma arquitetura (listagem 2) com um maior número de camadas densas, porém com considerável redução em perda de treino (de 11.78 para 7.4) e validação (de 3.84 para 2.2).

```
# Initial architecture candidate:
architecture {
  'conv': [(47, 'relu')],
  'dense': [(10, 'softmax')]
```

```
]}:
|-train-loss: 11.782062
|-validation loss: 3.841851

...
# Final architecture candidate:
architecture {
  'conv': [(47, 'relu')],
  'dense': [(40, 'relu'), (10, 'softmax')]
]}:
|-train-loss: 7.403219
|-validation loss: 2.209876
```

Listing 2. Resultado da busca por algoritmo genético sobre o conjunto **Digits**.

B. *Conjunto de dados **Cifar-10***

Cifar-10 contém imagens em formato RGB percentendes à 10 distintas classes relacionadas à elementos do mundo real (e.g. aviões, carros, cavalos, navios).

Aqui, *Hill-Climbing* foi utilizado para partir de uma arquitetura com maior número de camadas convolucionais e alcançar uma arquitetura de notável menor complexidade (3), enquanto mantendo ambas perdas de treinamento e teste similares às originais.

```
# Initial architecture candidate:
architecture {
  'conv': [(47, 'relu'), (47, 'relu'),
           (54, 'relu'), (104, 'relu'),
           (200, 'relu')],
  'dense': [(75, 'relu'), (10, 'softmax')]
}:
|-train-loss: 0.429305
|-validation loss: 2.174616

...
# Final architecture candidate:
architecture {
  'conv': [(39, 'relu'), (127, 'relu'),
           (136, 'relu')],
  'dense': [(79, 'relu'), (10, 'softmax')]
]}:
|-train-loss: 0.422453
|-validation loss: 2.145460
```

Listing 3. Resultado da busca por *Hill-Climbing* sobre o conjunto de dados **Cifar-10**.

V. Considerações finais

Este trabalho apresentou uma possível modelagem do problema de seleção de arquiteturas como um problema de inteligência artificial clássico, resolvível por agentes baseados em utilidade e algoritmos de buscas dos mais variados. O modelo foi testado em dois conjuntos de dados distintos, exibindo interessantes resultados em ambos.

Como trabalhos futuros, destaca-se a necessidade de acelerar o processo de busca em si através do melhoramento do desempenho da avaliação de utilidade de uma determinada arquitetura. Este processo requer, atualmente, o treinamento completo dos modelos (iniciados com pesos aleatórios) descritos por cada arquitetura. Uma opção a se considerar é experimentar com a transferência parcial de pesos, possívelmente acelerando a convergência do treino e, portanto, das avaliações.

References

[1] B. YEGNANARAYANA, *ARTIFICIAL NEURAL NETWORKS*. PHI Learning, 2009. 1

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105. 2

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 2

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. 2

[5] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*. Prentice hall Upper Saddle River, 2003, vol. 2. 2

[6] C. R. Darwin, *The Origin of Species*. P.F. Collier & Son, 1909-14, vol. XI. 3