Utilização do algoritmo PSO para ajuste dos pesos em redes RBF

Daniel Vilas-Boas
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco
daanielvb@gmail.com

Leonardo Figueiroa
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco
leonardofigueiroa@live.com

Rodrigo Cunha
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco
r-cunha@outlook.com

Abstract

O uso de redes neurais de base radial ou RBFs vem sendo bastante utilizadas para classificação de padrões por apresentar diversas vantagens sobre outras redes (como a MLP), apresentando um treinamento mais eficiente e melhor grau de separabilidade. Este trabalho envolveu o desenvolvimento de uma RBF em conjunto com o algoritmo de otimização por enxame de partículas ou PSO. A rede neural foi desenvolvida de forma que seus pesos de saída fossem ajustados pelo PSO visando obter os melhores pesos na camada de saída e consequentemente menor taxa de erro e melhor classificação das características de entrada.

Palavras-chave — RBF; PSO; redes neurais; classificação de padrões

1. Introdução

As redes denominadas *funções de base radial*, convencionalmente conhecidas como *RBF (radial basis function)*, são usadas em variados tipos de problemas tais como aproximação de funções e classificação de padrões. Ela pertence a arquitetura **feedforward** de camas múltiplas, cujo treinamento é efetivado de forma supervisionada. Sua estrutura é composta tipicamente por apenas uma camada intermediária, na qual as funções de ativação são do tipo *gaussiana*. O fluxo de informações tem início na camada de entrada, passando então pela respectiva camada intermediária, e finalizando na camada neural de saída com neurônios com funções de ativação linear [1].

O *PSO (Particle Swarm Optimization)* é um algoritmo de otimização por enxame de partículas. Tem uma abordagem **estocástica**, baseada em população que simula o processo comportamental de interação entre os indivíduos de um grupo. Sua teoria é baseada em comportamento de atividades de grupos de animais como pássaros e peixes, que realizam tarefas de otimização na execução de atividades como a busca por alimentos. O *PSO* se inicia com um enxame de partículas com posições aleatórias. Cada partícula é dita ser uma possível solução para o problema investigado, sendo atribuído a cada indivíduo (partícula) um valor que está relacionado a adequação da partícula com a solução do problema (denominada **fitness**), e, também, uma variável velocidade que representa a direção do movimento da partícula. Com o passar do tempo, as partículas vão ajustando suas velocidades em relação a seu melhor **fitness**, encontrada pela própria partícula, e também pela melhor solução do grupo de partículas.
Elas continuam realizando este processo até que encontrem uma solução ótima. O valor **fitness** é definido pela natureza do problema de otimização e é computada por uma função objetivo que avalia um vetor solução.
Neste papel são discutidos a implementação da rede neural *RBF* em conjunto com o algoritmo de otimização *PSO*, os experimentos realizados após a implementação, e os resultados encontrados após os experimentos realizados. Todo o histórico de desenvolvimento do trabalho e código fonte atualizado se encontram disponíveis em `https://github.com/Daanielvb/RBF_PSO`

1.1. Procedimentos

Nesta seção serão discutidos os procedimentos realizados pela rede neural e pelo algoritmo de otimização. Os procedimentos são outros algoritmos conhecidos na literatura de **inteligência artificial** e **redes neurais artificiais** (como o *kMeans* e a *RBF*). Serão brevemente discutidos como funcionam e o seu papel na implementação do projeto.

1.2. kMeans

O primeiro procedimento realizado foi a **clusterização** das amostras por meio do algoritmo *kMeans*. A partir de um valor n que representa o número de centros na camada escondida da *RBF* o algoritmo realiza a separação das amostras, calculando n centros e suas respectivas variâncias.

1.3. Radial Basis Function — RBF

Diferentemente das redes neurais do tipo *MLP*, as redes do tipo *RBF* possuem uma única camada escondida, cujos neurônios são constituídos de **funções de base radial**. O aprendizado neste tipo de rede é equivalente a encontrar uma superfície no espaço que forneça o melhor ajuste para os dados de treinamento. A estrutura típica de uma rede de função de base radial possui uma cama de entrada que está associada diretamente às informações de entrada da rede, uma única camada escondida constituída por funções de ativação de base radial que realizam uma transformação não-linear do espaço de entrada, e uma camada de saída linear que fornece a resposta ao padrão aplicado nas entradas da rede, conforme a figura abaixo. [3] Nas redes neurais

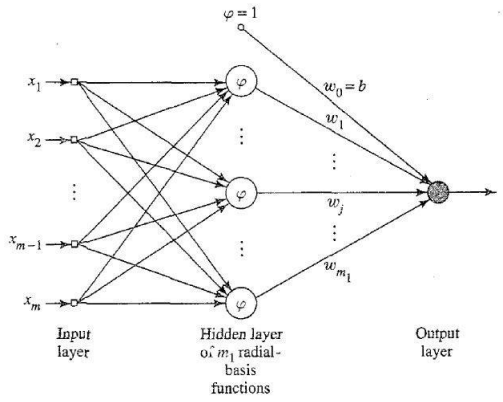

Fig. 1: *Estrutura teórica da RBF [3]*

do tipo *RBF* as informações das funções da camada escondida são determinadas pela distância, normalmente *euclidiana*, entre o vetor de entrada e o centro daquela unidade. Ou seja, as variáveis de entrada constituem os dados utilizados pelas funções de ativação da rede, onde os pesos que conectam as camadas de entrada e escondida podem ser considerados unitários. A camada de saída **não** contém funções de ativação, ela é considerada uma combinação linear que é dada pela soma ponderada dos valores gerados pelas funções de ativação das unidades escondidas, que são multiplicados pelos correspondentes pesos da camada de saída da rede.

1.4. Cálculo das saídas intermediárias — Y

A partir dos centros das amostras e suas respectivas variâncias é utilizada a **função de base radial** para determinar o valor de saída da camada intermediária. A fórmula é dada abaixo:

$$e^{-\frac{(u-c)^2}{2\sigma^2}} \tag{1}$$

Onde e é a função exponencial de $(u - c)^2$ que é a **distância euclidiana** entre centro (c) e as amostra (u), pela **variância** (σ) ao quadrado multiplicada por 2.

1.5. Particle Swarm Optimisation — PSO

A otimização através de enxame de partículas é uma técnica **estocástica** baseada em processos comportamentais de grupos de animais criada pelo *Dr. Eberhart* e *Dr. Kennedy* em 1995. O *PSO* possui várias similaridades com as técnicas evolucionárias como os **Algoritmos Genéticos**, pois utiliza o conceito de vida artificial para interpretar comportamentos biológicos, e, por isso, é bastante utilizada em *computer animation* e *computer aided design*. No entanto, apesar de possuir algumas similaridades como inicialização aleatória da população, a presença de um fator comparativo para avaliar a população (fitness) e atualização da população à medida que procuram pela solução ótima, diferenciam-se quanto a presença de operadores evolucionários como o **crossover** e a **mutação**. Isso resulta na evolução apenas da procura pela solução ótima e nao dos organismos em si. [4]

O conceito do algoritmo é gerar uma coleção de partículas dentro de um espaço de função cujas dimensões são variáveis com o número de neurônios da camada escondida da *RBF*. Cada partícula, então, segue o líder do enxame (através da melhor partícula global em cada iteração), atualizando sua velocidade e consequentemente sua posição através das equações (a) e (b) [4] que serão discutidas abaixo. Após e iterações do algoritmo, uma partícula será escolhida como a melhor do enxame dentre todas as épocas, e sua posição será usada no cálculo final da classificação na *RBF* como o conjunto de pesos $\{w_1, w_2, w_3, ..., w_e\}$ otimizado.
Nessa implementação **não** foram consideradas topologias de vizinhança para as partículas, ou seja, não há troca de informação entre as partículas vizinhas. [2]
Abaixo, segue o algoritmo em pseudo-código do procedimento.

Algoritmo 1: Pseudo-código do PSO

```
início
    para cada particula faça
        inicialize a particula;
    fim
    repita
        para cada particula faça
            calcule o valor do fitness;
            se fitness for melhor (≤) que o fitness em
              pBest então
                guarde o valor como o novo pBest;
            fim
        fim
        Escolha a particula com melhor valor de
          fitness de todas as particulas como a gBest;
        para cada particula faça
            calcule a velocidade de acordo com a
              equação (a);
            atualize a posição de acordo com a
              equação (b);
        fim
    até número de épocas ser atingido;
fim
```

Após as partículas serem inicializadas no espaço de função (intervalo de -1 a 1) é calculado o valor do **fitness** de cada partícula. O cálculo do valor é feito pela *RBF* como o percentual de erro, discutido na seção acima. Após o fitness ter sido calculado definimos o melhor global, que age como um fator de liderança para as demais partículas, e o melhor local da partícula (fator inércia). A partir deles, podemos calcular os vetores resultantes auxiliares e depois o vetor velocidade final que irá atualizar a posição da partícula no espaço. As equações de cálculo da velocidade e atualização da posição seguem:

$$\textit{(a)} \qquad \vec{v} = \vec{v} + c_1 \times rand() \times (\vec{pBest} - \vec{pos}) + c_2 \times rand() \times (\vec{gBest} - \vec{pos})$$

Onde $\vec{v}$ é o vetor velocidade, c_1 e c_2 são fatores de aprendizado ($c_1 = c_2 = 2$), $rand()$ é uma função que irá gerar um valor aleatório no intervalo 0 — 1, $\vec{pBest}$ e $\vec{gBest}$ são os vetores de posição do melhor local e global respectivamente, e $\vec{pos}$ é o vetor posição atual da partícula.

$$\textit{(b)} \qquad \vec{pos} = \vec{pos_{ant}} + \vec{v}$$

Onde $\vec{pos}$ é o vetor posição atual da partícula a ser atualizado, $\vec{pos_{ant}}$ é o vetor posição anterior a atualização, e $\vec{v}$ é o vetor velocidade calculado através da equação (a).

2. Experimentos

2.1. Variando o número de neurônios, épocas e partículas

Foram realizados experimentos modificando os parâmetros do número de neurônios, número de épocas e número de partículas no enxame. Os parâmetros são tratados como n, e, e t respectivamente. Os experimentos realizados tem como objetivo observar as mudanças quanto à **taxa de acerto** (c) e o **fitness** (f) para a base de dados de entrada em três iterações para cada parâmetro alterado. No apêndice encontram-se as tabelas com os resultados dos experimentos realizados.

2.2. Observações

Foi observado que, para ambas as bases de dados, quando a rede neural está configurada como $n = 5$, $e = 10$, e $t = 20$ foram obtidos melhores resultados em relação às saídas c e f. Há um declínio observado no fitness e na taxa de acerto quando aumentamos o número de neurônios, épocas e partículas em ambas as tabelas 1 e 2.

3. Conclusão

Redes neurais artificiais *ou RNAs*, possuem uma grande aplicabilidade em diversas áreas de classificação de padrões. Aliada com o algoritmo de otimização de enxame de partículas *PSO*, seu desempenho é melhorado em relação a taxa de acertos. Em nossos experimentos percebemos que ainda há alguns ajustes (*fine-tuning*) nos parâmetros tanto da *RBF* e *PSO*, para que possamos atingir uma solução ótima no teste com as bases de dados fornecidas.

Referências

[1] I. N. da Silva. *Redes Neurais Artificiais Para Engenharia e Ciências Aplicadas*. Artiliber, first edition, 2010.

[2] A. R. Gonçalves. Inteligência de enxames. *IEEE Congress on Evolutionary Computation*, 2013.

[3] S. Haykin. *Neural Networks - a comprehensive foundation*. Prentice Hall, second edition, 1999.

[4] X. Hu. Particle swarm optimization.

Apêndice

Tabelas dos experimentos

Aqui encontram-se as tabelas com os resultados dos experimentos feitos com as bases de dados *Iris.txt* e *Banknote.txt*.

	1			2			3		
	$n = 5$	$e = 10$	$t = 20$	$n = 10$	$e = 50$	$t = 100$	$n = 20$	$e = 30$	$t = 50$
tx. acerto									
c	70	78	76	66	70	54	56	64	60
fitness									
f	0.99833	0.99887	0.99909	0.99918	0.99975	0.99847	0.99969	0.99971	0.99981

Tabela 1: Resultados dos experimentos com a base de dados Iris.txt

	1			2			3		
	$n = 5$	$e = 10$	$t = 20$	$n = 10$	$e = 50$	$t = 100$	$n = 20$	$e = 30$	$t = 50$
tx. acerto									
c	78	82.0	72	60	56	66	72	70	76
fitness									
f	0.99881	0.99926	0.99909	0.99940	0.99955	0.99888	0.99946	0.99951	0.99971

Tabela 2: Resultados dos experimentos com a base de dados Banknote.txt