Uma Implementação em Software do Classificador WISARD

Cíntia M. Soares*, Cassiano L. Fróes da Silva*, Massimo de Gregorio• e Felipe M. G. França*
* Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ, Rio de Janeiro, RJ, Brasil
• Istituto di Cibernetica – CNR, Nápolis, Itália
cintiams@domain.com.br, cassiano@cos.ufrj.br, massimo@sole.cib.na.cnr.it, felipe@cos.ufrj.br

Resumo

Uma das principais vantagens das Redes Neurais Sem-Peso (RNSP) está na pequena distancia semântica entre seus modelos e sistemas digitais convencionais. Este trabalho descreve uma implementação em sotware do neuroclassificador de imagens WISARD, capaz de reconhecimento e aprendizado de imagens em tempo real. O único hardware específico utilizado foi uma câmera de vídeo padrão para conexão com um PC desktop de configuração padrão no mercado. Explorações preliminares com dois aprimoramentos da WISARD também são descritos: (i) DRASIW - uma versão capaz de produzir um exemplo significativo de uma classe já aprendida via retroclassificação, e; (ii) uma versão multiresolucional da WISARD onde, visando um balanço entre capacidades de classificação e generalização, tuplas com números de entrada diferentes coexistem em uma mesma população.

1. Introdução

WISARD (**W**ilkie, **S**tonham and **A**leksander's **R**ecognition **D**evice) [1] é uma máquina para aprendizado e reconhecimento de imagens que utiliza princípios neurais, e foi o primeiro neurocomputador comercial a ser construído (o protótipo foi terminado em 1981, Brunel University, Londres e a máquina foi patenteada e produzida comercialmente em 1984). Seu mecanismo de aprendizado não se baseia na alteração dos valores de pesos explicitos, mas sim na alteração do conteúdo de memórias distribuídas. Para tal são utilizadas memórias RAM convencionais, que podem ser escritas e lidas, e operam de forma similar à neurônios de MaCulloc-Pitts. Pode ser fácilmente demonstrado que uma (1) camada desses neurônios operam de forma similar a uma camada de perceptrons.

Implementada a base de memórias convencionais, a máquina WISARD é capaz de efetuar uma (1) classificação de uma imagem em preto-e-branco a cada 1/25 segundos, velocidade consideravelmente alta, sobre imagens de até 512x512 pixels. Visando tirar proveito da agilidade do modelo WISARD sem os custos impostos em se ter um sistema de hardware especializado, este trabalho descreve a implementação de WISARD em software com o auxílio de um único hardware específico: uma câmera de vídeo padrão para conexão com um PC desktop. Duas investigações sobre o modelo WISARD são descritas: (i) DRASIW - uma versão de WISARD capaz de, quando requisitado, produzir um exemplo significativo de uma classe já aprendida via retroclassificação, e; (ii) uma versão multiresolucional da WISARD onde, visando um balanço entre capacidades de classificação e generalização, tuplas com números de entrada diferentes coexistem em uma mesma população.

O restante deste trabalho está organizado da seguinte forma: uma descrição em detalhes do modelo WISARD é feita na próxima seção; a Seção 3 apresenta nossa implementação em software do modelo, incluindo DRASIW e a introdução de diversidade nas tuplas. Nossa conclusões são traçadas na Seção 4.

2. Uma descrição do modelo WISARD

O modelo WISARD é composto basicamente por dois elementos: *tuplas* e *descriminadores*. Um descriminador é composto por um conjunto de tuplas e WISARD é um conjunto de descriminadores. A seguir apresentamos o funcionamento desses elementos.

Tuplas. Cada tupla é uma pequena memória, geralmente com endereços de 2, 4 ou 8 bits e um bit de dado por endereço. Isso significa que podemos endereçar 2^2, 2^4 ou 2^8 posições de um bit por tupla. Para cada treinamento realizado, a tupla recebe como entrada um subconjunto dos pixels da imagem (esse subconjunto é fixo para cada tupla).

A tupla lê esse conjunto de pixels e o mapeia para um endereço da sua memória. Por exemplo, se os pixels fossem '0011', o endereço 3 seria acessado. Como a tupla está sendo treinada, nessa posição da memória é escrito '1', representando que aquela disposição de pixels foi "vista". Portanto, o conteúdo de cada posição da memória da tupla representa o conhecimento obtido a partir de um conjunto de pixels das imagens presentes no treinamento. A Figura 1 ilustra o mapeamento de uma tupla.

Durante a classificação, o mesmo mapeamento é realizado, e a saída da tupla é o conteúdo do endereço acessado. A tupla, portanto, responde se aquela configuração de pixels esteve ou não presente no treinamento.

o passar do tempo e treinamentos posteriores não afetam a memória atual.

O tamanho a ser escolhido para as tuplas depende do tipo de classificação que se quer obter. No caso de tuplas menores (2 bits, por exemplo) a classificação é

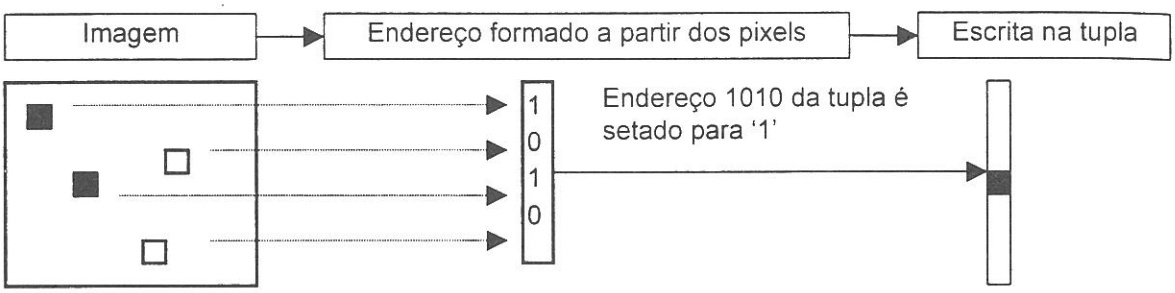


Figura 1. Esquema do processo de treinamento de uma tupla.

Discriminadores. Como cada tupla enxerga apenas um pequeno pedaço da imagem são necessárias várias tuplas para cobrir uma imagem por completo. Esse conjunto de tuplas é chamado de discriminador. Um discriminador é responsável por sumarizar as respostas das suas tuplas, sendo sua resposta o somatório das respostas das tuplas.

WISARD é um modelo de multi-discriminadores, pois utiliza um discriminador para cada classe presente no treinamento. A resposta do WISARD é baseada na classe cujo discriminador teve a maior saída. Isto porque o discriminador com a maior saída é aquele cujas tuplas já "viram" mais subconjuntos de imagens parecidas com a que está sendo classificada. Uma parte importante do modelo é a interpretação do resultado, pois a resposta do WISARD não é simplesmente 'sim' ou 'não'. A resposta informa que a imagem se parece mais com uma determinada classe do que com outras; ou talvez que a imagem pareça muito com duas ou mais classes conhecidas, e dificilmente seja de uma outra classe em particular.

Essa resposta pode chegar a 100% (por exemplo, quando são classificadas imagens que estavam presentes no treinamento). Uma vez que a informação está gravada na memória, o WISARD nunca 'esquece' com

mais genérica pois o número de tuplas a treinar sem saturação é menor. Logo, a capacidade de classificação vai aumentando à media que o tamanho da tupla aumenta.

Algumas observações importante sobre como distribuir os pixels pelas tuplas:

1. Tipicamente, as tuplas enxergam pixels espalhados pela imagem. É realizado inicialmente um mapeamento, distribuindo pixels aleatoriamente pelas tuplas.
2. Uma vez feito esse mapeamento, a tupla sempre observará aquele conjunto de pixels que foi atribuído a ela.
3. Geralmente não existe sobreposição, ou seja, cada pixel é mapeado em apenas uma tupla, mas isso não é um requisito fundamental.
4. Não é necessário que todos os discriminadores trabalhem com o mesmo mapeamento. Mapeamentos diferentes podem até melhorar o resultado, já que um tipo de mapeamento pode se adequar melhor a uma determinada classe.
5. O mapeamento não precisa ser aleatório. Existem estudos sobre mapeamentos diferentes, seguindo um determinado padrão, de acordo com as imagens que serão analisadas. Essa abordagem requer, obviamente, um estudo anterior sobre as classes que serão treinadas.

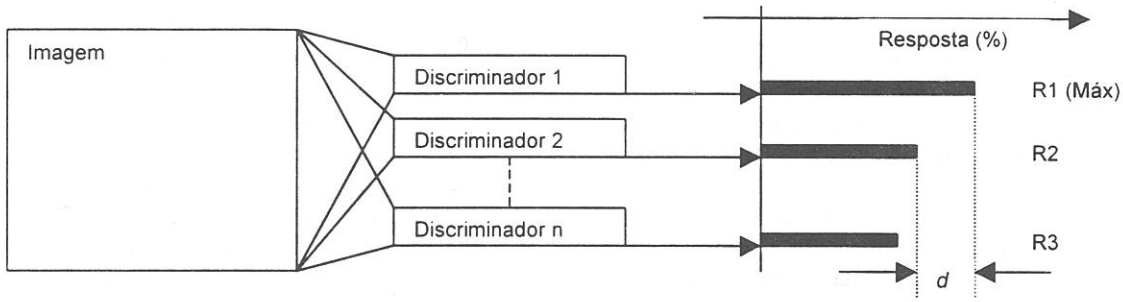


Figura 2. WISARD: resposta dos discriminadores, onde *d* é a *confiança* da resposta.

3. WISARD em software

A versão do WISARD apresentada aqui foi implementada, em linguagem Pascal (Borland Delphi), para uma estação tipo PC com a utilização de uma câmera digital (Connectix) [2]. A primeira versão testada foi o WISARD tradicional, com opção para tuplas de 2, 4 e 8 pixels, e imagens em até 512x512 pixels (o tamanho da imagem só é limitado pelas características da máquina). De uma primeira alteração no modelo obteve-se o DRASIW, que permite que, após o treinamento, seja possível pedir ao sistema que produza um exemplo significativo de uma determinada classe já treinada. Finalmente foi feita uma implementação do conjunto heterogêneo de tuplas.

3.1. DRASIW

No WISARD, que é formado por um conjunto de tuplas por cada classe, cada tupla recebe como entrada alguns bits, que compõem o endereço a ser marcado (no treinamento) ou lido (na classificação). Sabemos que esses bits são pixels da imagem sendo avaliada, e qual a sua localização nesta imagem. Logo, um procedimento que varra as tuplas e selecione os endereços que estão marcados conseguirá facilmente mapear quais foram os pixels da imagem original que causaram aquela configuração das tuplas.

Para fazer este processo reverso, basta varrer as posições da memória de cada tupla, avaliando os endereços marcados, e decodificando esses endereços para posições na imagem. Esse procedimento de varredura é bastante simples, pois todos os passos realizados para acessar um endereço da tupla podem ser invertidos. Por exemplo, se uma tupla de 4 bits possui o endereço 1001 marcado, o primeiro e o último pixels mapeados para ela serão pintados de preto, e o segundo e o quarto serão marcados de branco. Este processo de varredura "ao contrário" da memória foi batizado de DRASIW [3].

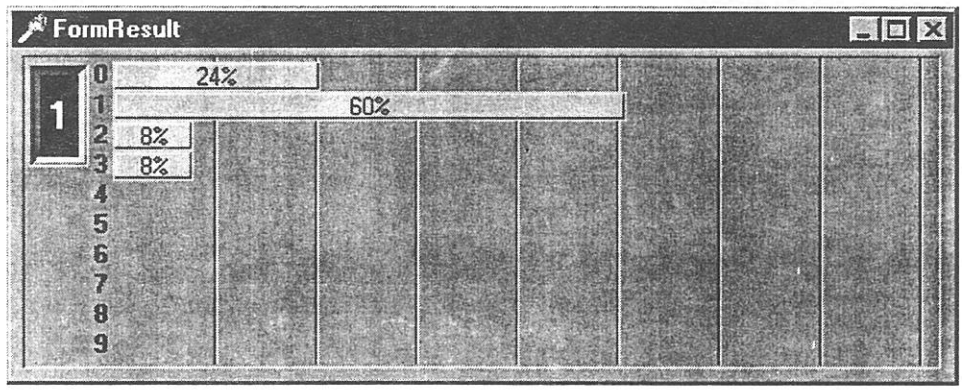


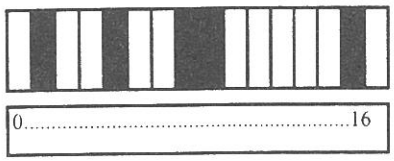
Figura 3. Tela do sistema mostrando o resultado de uma classificação. A resposta foi a classe '1'.

Se apenas uma imagem foi apresentada ao WISARD, a resposta do algoritmo será a imagem exata, pois cada tupla terá um único endereço marcado, que mapeará exatamente uma única seqüência de pixels na imagem. No entanto, conforme mais imagens são apresentadas, a resposta fica menos nítida. À medida que novos padrões são apresentados, serão encontrados pixels em posições diferentes, causando a marcação de mais endereços das tuplas. Possivelmente várias tuplas terão mais de um endereço marcado, dificultando o processo de varredura.

Um outro problema que ocorre é que um único pixel aparece em vários endereços de uma tupla (1000, 1001, 1010, etc.); mas se algum dos seus endereços

estiver marcado, ele será pintado de preto. Uma solução aqui é fazer uma análise de todos estes endereços onde o pixel aparece, e verificar se a maioria está ou não marcada, utilizando esse resultado para pintar ou não o pixel.

O processo funciona perfeitamente com o WISARD padrão, e pode-se ver claramente as imagens de todas as classes que foram apresentadas ao algoritmo. Porém, algumas classes possuem padrões bastante heterogêneos, podendo dentro de uma mesma classe ter-se imagens bastante diferentes. Essas classes serão classificadas pelo WISARD corretamente (desde que não haja uma saturação de muitas tuplas), mas produzirão apenas um borrão na imagem quando for realizado o procedimento de varredura da memória das tuplas, após o treinamento de um número significante de padrões.



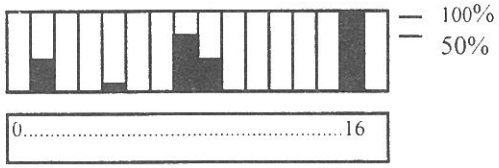


Figura 4. (a) WISARD - informação se o endereço está ou não marcado (0 ou 1); (b) DRASIW- informação sobre a freqüência do endereço (representada pela altura da barra)

Certamente alguns pixels aparecerão mais vezes do que outros nos diversos padrões da classe, mas o WISARD não reconhece isso: a informação é guardada em apenas um bit para cada endereço enviado da imagem. Ou seja, um endereço de uma determinada tupla que esteve presente em todos os padrões apresentados tem o mesmo peso que um outro endereço que apareceu em apenas um padrão, e a informação da freqüência com que o endereço ocorre é perdida. Essa informação pode ser útil para o processo de visualização do treinamento do WISARD, pois certamente os endereços com maior freqüência são mais importantes.

Uma pequena alteração no algoritmo permite que se obtenha maiores detalhes sobre os pixels e endereços apresentados à tupla: guardar um contador por endereço, e não apenas um bit. Esse contador deve ser inicialmente zerado para todos os endereços e incrementado a cada vez que o endereço for acessado no treinamento. Assim obtemos a informação da freqüência dos endereços das tuplas de cada classe.

Para realizar o procedimento de "lembrança" das classes, utiliza-se uma coloração diferente para os pixels, de acordo com a freqüência que eles aparecem. Assim, a imagem da lembrança terá uma cor mais escura para os pixels que aparecem mais vezes, já que as freqüências dos seus endereços será maior. Aqui damos preferência aos endereços com maior freqüência. Por exemplo, o endereço 1000 pode ter freqüência baixa, e o endereço 1111 ter freqüência alta; eles compartilham os mesmos pixels (pois são dois endereços da mesma tupla), e eles serão pintados de acordo com a cor para a freqüência alta, pois aparecem em pelo menos um endereço com esta freqüência. Outras abordagens podem ser utilizadas, porém esta se mostrou bastante eficiente. Por exemplo, pode-se utilizar uma média da freqüência de cada pixel.

Figura 5. A primeira figura é a memória de um discriminador 'simples', com tuplas de um bit por endereço; a segunda figura, de um discriminador com tuplas modificadas para guardar a freqüência dos endereços, e a terceira imagem é o resultado de um filtro sobre a segunda, mostrando apenas os pixels com maior freqüência.

Na implementação apresentada aqui, o contador utilizado possui 1 byte; Pode-se, portanto, armazenar informação de até 256 incidências no mesmo endereço, diminuindo o índice de saturação das tuplas. Nesta implementação, a classificação pode ser realizada da mesma forma que no WISARD padrão, ou seja, considerando todo endereço diferente de zero como sendo um endereço marcado, e com saída 1 para a classificação. Uma outra abordagem para a classificação utiliza a vantagem que se tem em se armazenar a freqüência dos endereços: o resultado da tupla, na classificação, é obtido a partir do contador. O valor do contador do endereço que estiver sendo consultado deve ser normalizado para gerar a saída da tupla. Esta normalização é necessária, pois um diferente número de padrões

pode ser apresentado a cada classe, causando um desequilíbrio nos valores armazenados nos contadores. Certamente, uma classe que foi treinada com 200 padrões terá valores maiores nos contadores do que uma classe que foi treinada com 100 padrões.

Um exemplo: Foram apresentados 5 padrões de uma determinada classe para o DRASIW. Uma determinada tupla teve alguns endereços que apareceram em todos os padrões, tendo, portanto, o contador igual a 5. Um outro endereço apareceu 2 vezes (contador = 2). Os endereços que não apareceram continuam se comportando da mesma forma para a classificação, resultando num valor de saída 0. Os endereços com contador 5 gerarão como saída 1, pois estão presentes em 100% dos padrões usados no treinamento. Já os endereços com contador 2 terão como saída 0.4 já que apareceram em 2/5 = 40% dos padrões.

3.2. Tuplas de tamanhos diversos

Como foi visto anteriormente, o fato da classificação ser mais ou menos específica está ligado ao tamanho das tuplas. No caso de um tamanho menor teremos uma maior generalização. Pode-se pensar então em fazer uma implementação com tuplas grandes apenas. Mas nesse caso estaríamos gastando muita memória e para certas aplicações é mais interessante ter uma classificação mais genérica. Trabalhando com vários tamanhos de tuplas conseguiríamos esse cenário.

O problema agora é definir que tuplas teriam que tamanho. Se classificarmos as tuplas de acordo com sua distância ao centro da imagem podemos dizer que as tuplas dos pixels centrais necessitam de uma classificação mais específica pois definem mais a imagem que as tuplas do contorno. Essas últimas podem variar mais, por efeitos de sombra, distorções etc, de forma que devem utilizar tamanhos maiores.

A implementação realizada utiliza tuplas de 2, 4 e 8 bits concorrentemente. O tamanho das tuplas é definido através de uma função fuzzy de acordo com a distância do pixel ao centro. Como se trata de uma função fuzzy, ou seja, não diz se uma tupla é de um tamanho específico mas o quanto ela pode ser daquele tamanho, usamos um parâmetro randômico para definir a que classe ela pertence.

Supondo que para um certo pixel tenhamos os valores de 0.5 para 8 bits, 0.3 para 4 bits e 0.2 para 2 bits, a estratégia consiste em gerar um número aleatório para cada tamanho e multiplicá-lo pelo valor correpondente na função. O maior valor obtido será o tamanho escolhido. Se o número gerado para 8 bits for 1 para 4 bits 0.5 e para 2 bits, 0.4 nos resta escolher o maior entre os valores 0.05, 0.15 e 0.08. Portanto o tamanho escolhido será o de 4 bits.

4 Conclusão

Implementou-se o classificador WISARD com duas extensões interessantes: a possibilidade de visualização do treinamento (e portanto de controlar sua qualidade) através da estratégia DRASIW e a introdução de uma retina multiresolucional artificial, que sugere a possibilidade de se desfrutar de um melhor balanço entre as capacidades de classificação e generalização de WISARD. A implementação permite a classificação com sucesso de imagens relativamente grandes, da ordem de 500x500 pixels, ou maior, dependendo apenas de limitações de hardware. O algoritmo de treinamento, por ser bastante simples, é fácil de ser implementado

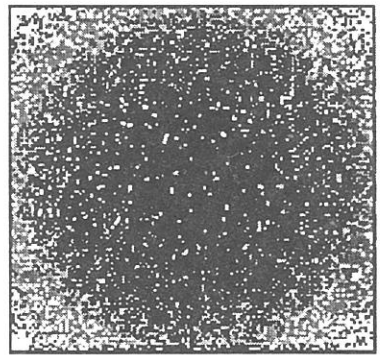


Figura 6. Mapeamento de uma imagem:As cores mais escuras representam as tuplas maiores.

5 Referências

[1] I. Aleksander and H Morton. *An Introduction to Neural Computing*. Chapman and Hall. 1990.

[2] C. Soares e C Fróes. "Uma Implementação em Software Tempo Real do Classificador WISARD". Projeto de Fim de Curso em andamento. DCC/IM - UFRJ. 1998.

[3] M. De Gregorio. "Image-Driven Reasoning". *Anais do Workshop em Inteligência Computacional: Projetos ICOM e IPAC. Protem III-C. CNPq. ES-450*. Setembro 1997.