

Trabalho de RNA - GA

Giovani Facchini

8 de outubro de 2004

giovani@exatas.unisinos.br

1 Descrição do Problema

O desafio proposto era criar uma rede capaz de identificar as linhas da palma de uma mão. O problema era de classificação, pois dado alguns pontos a rede deveria responder se um determinado ponto pertenceria a uma linha da palma da mão, ou não.

Para isso foram tiradas fotos de algumas mãos com uma máquina fotográfica digital. Para gerar um arquivo que contivesse as linhas da palma da mão as fotos foram tratadas no Photoshop. Colocou-se uma layer transparente em cima da foto original e as linhas foram traçadas em preto por cima das linhas originais. O arquivo de saída foi somente a layer que continha as linhas pretas desenhadas com base nas linhas da mão.

Para gerar dados que fossem possíveis de inserir na rede, foi utilizado o software *ImageTester* utilizando um algoritmo chamado de *Walker*. Para gerar a entrada da rede, utilizamos o Walker como sendo uma janela de 5x5 pixels que percorria a foto original (palma da mão) e gerava um arquivo de saída que continha, em ASCII, o valor de cada canal RGB dos pixels da janela por linha.

O arquivo utilizado na saída (para fazer a supervisão) foi gerado, também, com o algoritmo Walker sobre a imagem gerada com o Photoshop (em preto e branco). Desta vez não se utilizou uma janela e sim, percorreu-se a imagem dentro de um quadro que representava o deslocamento da janela na foto original andando sempre sobre o ponto central da janela sobre a foto original. Isso foi feito para se conseguir um 'contexto' de imagem para passarmos para rede, então a entrada seria a janela 5x5 e a saída seria a pixel central com a decisão preto ou branco.

Os arquivos de entrada e saída gerados tinham em média 300MB e 1,2MB, respectivamente. Como utilizar todos esses dados de entrada tornaria o aprendizado muito lento, optou-se por selecionar um conjunto de 8000 exemplos para ser utilizado. Observando o comportamento da rede de um dos colegas pode-se observar que se todos os exemplos fossem utilizados, a rede teria um erro

baixo, mas ela iria responder sempre para a classe branco, pois essa tinha uma dominância muito grande.

Foram feitos scrips em Python para particionar e selecionar pedaços da base de dados. Para balancear, foram escolhidos a mesma quantidade de exemplo de cada uma das classes (cerca de 4000 para cada classe). Esse número foi escolhido pois era a quantidade máxima de exemplos da classe preta. Com isso escolhemos aleatoriamente 4000 exemplos da classe branca dentro da base de dados.

Separamos a base de dados em duas partes, uma para treino e outra para teste, com o objetivo de fazer validação cruzada. Para a base de treino foram utilizadas 6000 amostras e para a base de teste 2000 amostras, sendo que cada amostra estava balanceada em 50% preta e 50% branca.

A número de entradas foi de 75. Isso aconteceu devido a janela 5x5 (25 entradas) multiplicada pelo número de canais RGB (3). E o número de saídas foi de 1 (classificação entre preto e branco). A codificação da entrada era a representação dos 25 pixels no canal RGB, ex: 125 145 156 54 1 10 243 2 ... e a codificação dos dados de saída era representada apenas por um número informando a classe, ex: 1.

2 Descrição da Rede Neural

Foram utilizadas várias redes para treino a fim de obter uma que tivesse uma quantidade de erros menor. A arquitetura era composta de uma camada de entrada com 75 entradas, uma camada escondida e um neurônio de saída. A camada escondida foi onde tivemos a variação da rede. Foram testadas redes com 5, 10, 15, 20, 25 e 30 neurônios na camada escondida.

A rede foi conectada utilizando uma conexão de cada entrada conectada em todos os neurônios da camada escondida e estes conectados no neurônio de saída. A função de transferência utilizada foi a sigmóide e a função de saída do neurônio de saída era binarizada, ou seja, se tivéssemos um valor até 0,5 o valor de saída seria 0 e acima disso seria 1.

O algoritmo utilizado para aprendizado foi o Resilient Propagation. Os parâmetros passados para o simulador foram uma variação do aprendizado inicial e da perda dos pesos. Para aprendizado inicial foram usado os valores 0.1, 0.2 e 0.4 e para perda dos pesos 2.0, 3.0 e 4.0. Foram feitos experimentos variando todos esses parâmetros em cima de cada uma das redes.

Para simulação foi utilizado o JavaNNS para a criação da rede e definição das funções de transferência e saída. Para rodar os experimentos de aprendizado e teste foi utilizado o batchman. Muitos experimentos foram executados. Os scripts, saídas e resultados serão enviados em um arquivo ao professor por e-mail ou pessoalmente (pois trata-se de um arquivo muito grande).

3 Metodologia Adotada para Treino

Para treinamento foram utilizadas todas as redes descritas acima e variando-se os parâmetros de entrada como descrito. Para cada experimento (uma configuração de rede com um conjunto de parâmetros de entrada) foram realizados 10 treinamentos com sementes diferentes. Cada script de treinamento gerado foi guardado.

Para aprendizado, foi utilizado um loop que ficava treinando a rede e para cada 5 treinos ele fazia um teste com a base de teste e verificava o erro. Foram executadas mais de 1000 épocas de treinamento para cada rede. A rede que continha o menor erro era salva no decorrer da simulação. Percebeu-se que a melhor configuração de rede acontecia sempre em menos de 500 épocas.

Depois de testadas todas as possibilidades descritas foi escolhida a rede que tinha o menor erro.

4 Resultados Finais

A rede final obtida foi um resultado de 540 simulações. Os arquivos em anexo mostram os resultados de todas as redes salva. Estas estão separadas em diretórios que indicam o número de neurônios na camada escondida. O segundo nível de diretórios separa por parâmetros de entrada $p1p2$ ($p1$ = aprendizado inicial, $p2$ = decaimento dos pesos). Dentro de cada diretório estará salvo as redes e resultados que contêm no nome o erro atual da rede, quantidade de ciclos treinados e semente utilizada na inicialização. Também estão anexados os scripts utilizados e os arquivos gerados pelos scripts para utilizar com o batchman.

5 Comentários Finais

Os resultados da rede pareceram extremamente bons, pois o número de acertos foi bem alto e o de erros pequeno. Os pontos negativos são o tempo que leva para se treinar a rede e atingir resultados bons (pois precisamos testar muitas variações).

Agora vou citar os pontos que poderiam melhorar, somente quero mostrar a minha visão de tudo que ainda poderia ser feito, não que isso seja que eu ache o meu trabalho ruim, pois eu sempre acho muitos pontos que podem ser mudados.

Os pontos fracos do trabalho são, principalmente, a utilização de apenas um algoritmo de aprendizado (sem testar toda a gama existente). Além de testar vários algoritmos, poderíamos ter uma variação maior dos tipos de rede utilizadas (colocando mais camadas escondidas e variando o número de neurônios de cada camada). Os parâmetros de cada método de aprendizado poderiam ser testados ostensivamente.

Outro fator de melhora seria conseguir escolher dados de entrada que conseguissem 'confundir' a rede, para podermos testar a seu nível de aprendizado. Isso poderia ser feito, conseguindo pontos que se parecem com linhas, mas não são, podendo testar a eficácia.

Para termos melhores resultados visuais do que a rede aprendeu, poderíamos passar para ela como entrada uma foto de uma mão e ver gerar a saída. Com isso poderíamos utilizar um algoritmo inverso ao Walker para remontar os traços das linhas da palma da mão e visualizar se estes 'batem' com a mão original.

Depois de variar todos esses parâmetros e utilizar como entrada vários tipos de mãos, poderíamos testar mãos ainda não vistas para ver o que aconteceria com a resposta da rede. Isso geraria resultados muito interessantes em termos de pesquisa. Como o trabalho proposto e tempo para desenvolvê-lo não permitem todo esse desenvolvimento (que acarretaria muito tempo para análise de dados, criação de scripts e muitas semanas de processamento), contenta-se com resultados preliminares que mostram que a rede conseguiu se sair bem para uma base de testes e treinamento escolhida aleatoriamente.