Parte I - Redes Neurais

Um dos problemas de maior importância para o desenvolvimento histórico de *machine* e *deep learning* foi o reconhecimento de dígitos manuscritos. Neste contexto, a base de imagens MNIST[1] teve papel fundamental. A forma mais usual de reconhecer dígitos nessa base usando os algoritmos que aprendemos neste curso é linearizando cada imagem original de tamanho 20×20 para um vetor de comprimento 400. O arquivo *imageMNIST.csv* contém em cada linha uma imagem da base MNIST transformada em vetor. Já o arquivo *labelMNIST.csv* contém o dígito correspondente àquela linha.

- Implemente uma rede neural regularizada com uma camada escondida como a vista em aula para resolver este problema. Considere usar, por exemplo, 25 "neurônios" na camada escondida. Mostre a porcentagem de imagens classificadas corretamente. Mostre também as imagens que a rede classificou incorretamente, juntamente com o dígito verdadeiro e o dígito que o classificador atribuiu. NOTA: Seu código deve funcionar também para um número qualquer de exemplos de treinamento, de classes, de camadas e de "neurônios".

- Implemente o algoritmo de checagem do gradiente para o seu *backpropagation*. Devido ao custo computacional, não é conveniente que este algoritmo seja executado sobre a rede toda, mas sim sobre uma rede bem menor. Para isso, gere uma rede pequena com exemplos de treinamento fictícios (valores aleatórios). Sugere-se uma rede com 3 unidades na entrada, 5 na camada escondida e 3 na saída, isso para 3 exemplos de treinamento. Considere $\epsilon = 10^{-4}$.

- Implemente a mesma rede regularizada para classificação da MNIST, mas agora usando gradiente conjugado (*scipy.optimize.minimize* no Python; *fmincg* no Matlab/Octave). Variar o número de iterações (ex. 400) e o λ.

- A i-ésima linha de $\Theta^{(1)}$ é um vetor de 401 elementos. Descartando-se o *bias*, temos um vetor de 400 elementos, o qual pode ser rearranjado para uma imagem 20×20. Esta imagem é uma representação visual da forma como cada unidade escondida atua na rede. Mostre esta imagem para cada unidade escondida. Normalize (*scaling*) os valores de cada imagem entre 0 e 1 para facilitar a visualização.

[1] http://yann.lecun.com/exdb/mnist/

Parte II - Seleção de Modelo

- Divida os exemplos (amostras da MNIST no caso) nos 3 conjuntos de treino, validação e teste. Sugere-se 60% para treino, 20% para validação e 20% para teste. Calcule o erro de treino, validação e teste para a rede neural regularizada da Parte I.

- Implemente um algoritmo para gerar a curva de aprendizado (erro de treino e validação em função do tamanho do treino) da rede neural regularizada.

- Automatize o processo de escolha do valor de λ. Use o conceito de conjunto de validação para isso. Sugere-se testar
  $\lambda = 0, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10$. Faça um gráfico do erro de validação em função do λ. Calcule o erro de teste para este λ ótimo encontrado.