Redes Neurais — Backpropagation
Trabalho 2 — INF01017

Prof. Bruno Castro da Silva
bsilva@inf.ufrgs.br

Profa. Mariana R. Mendoza
mrmendoza@inf.ufrgs.br

1 Objetivo

O Trabalho 2 da disciplina consiste na implementação de uma rede neural treinada via *backpropagation*. O trabalho será desenvolvido em **grupos de até 3 alunos** e, conforme discutido em aula, inclui a implementação de todas as características principais de redes neurais discutidas ao longo do semestre. Em particular, os grupos deverão implementar:

- uma rede neural composta por um número ajustável de neurônios e camadas e treinada via *backpropagation*;
- funcionalidade que permita, via linha de comando, informar a sua implementação a estrutura de uma rede de teste (i.e., estrutura de camadas/neurônios, pesos iniciais, e fator de regularização), e um conjunto de treinamento, e que retorne o gradiente calculado para cada peso;
- funcionalidade que permita, via linha de comando, efetuar a *verificação numérica* do gradiente, a fim de checar a corretude da implementação de cada grupo;
- um mecanismo para normalização das features/dados de treinamentos;
- mecanismo para uso de regularização.

Ao implementar a sua rede, não esqueça de incluir uma entrada de *bias* em cada neurônio, de utilizar um método apropriado para normalização de atributos, e de tomar cuidado para que a atualização dos pesos dos neurônios de *bias* seja feita da maneira correta/apropriada caso regularização seja utilizada (mais detalhes nos slides de aula).

Cada grupo, ao aplicar sua rede neural nos problemas de classificação, deverá utilizar a metodologia de uso de validação cruzada (*cross-validation*), a fim de ajustar o número de camadas e de neurônios da rede, assim como o fator de regularização. Isso é importante para ajustar a complexidade do modelo e evitar o problema de *overfitting*, garantindo assim que a rede neural tenha boa capacidade de generalização em dados de teste.

Como de costume, todos os trabalhos submetidos serão examinados de forma manual pelos professores, e também comparados com implementações conhecidas de redes neurais e com as implementações dos demais alunos, a fim de detectar plágios. É proibido o uso, total ou parcial, de bibliotecas ou implementações de aprendizagem de máquina ou de redes neurais que resolvam partes deste projeto. Se houver quaisquer dúvidas sobre se algum recurso de software pode ou não ser utilizado, por favor consultem um dos professores antes de fazê-lo.

2 Definição

- Os grupos deverão enviar seus códigos fonte pelo Moodle do INF até a data de entrega do trabalho, de acordo com o cronograma. Além disso, também será necessário produzir um relatório (em formato **pdf**) descrevendo as características gerais da implementação de cada grupo (p.ex., descrição de como os pesos são armazenados, como é feita a propagação da entrada para produzir a saída, detalhes sobre possíveis otimizações feitas para tornar o algoritmo mais eficiente, método de normalização utilizado em cada conjunto de dados, etc);
- Fica a critério de cada grupo se o *backpropagation* será implementado em sua forma clássica (não vetorizada) ou se os cálculos envolvidos (i.e., de saídas preditas, função de custo, e gradientes) será feito através de técnicas de vetorização. Recomendamos que seja feita a implementação vetorizada, pois ela é *significativamente* mais rápida do que a forma clássica do algoritmo, e também resulta em uma implementação relativamente mais simples;
- Cada grupo deverá prover uma função que permita a verificação, por parte dos professores, da corretude do algoritmo de *backpropagation*. Em particular, você deve implementar um programa que possa ser chamado pela linha de comando da seguinte forma:

  ```
  $ ./backpropagation network.txt initial_weights.txt dataset.txt
  ```

  onde

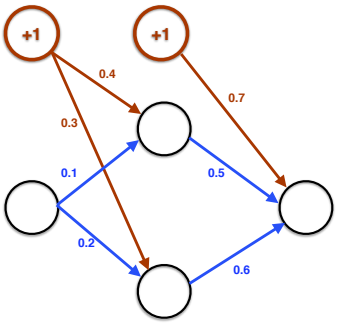
  1. *network.txt*: arquivo indicando a estrutura da rede. A primeira linha do arquivo deve armazenar o fator de regularização λ a ser utilizado. Cada linha subsequente no arquivo representa uma camada e o valor numérico armazenado naquela linha indica o número de neurônios na camada correspondente. Note que a primeira camada corresponde à camada de entrada da rede e a última camada corresponde à camada de saídas. O neurônio de bias *não* deve ser contabilizado na quantidade de neurônios de cada camada.

     Uma rede com 2 entradas, uma camada oculta com 3 neurônios, e 1 saída, por exemplo, e com fator de regularização $\lambda = 0.25$, seria descrita neste arquivo da seguinte forma:

     ```
     0.25
     2
     3
     1
     ```

  2. *initial_ weights.txt*: arquivo indicando os pesos iniciais a serem utilizados pela rede. Cada linha do arquivo armazena os pesos iniciais dos neurônios de uma dada camada—isto é, os pesos dos neurônios da camada i são armazenados na linha i do arquivo. Os pesos de cada neurônio devem ser separados por vírgulas, e os neurônios devem ser separados por ponto-e-vírgula. Note que o primeiro peso de cada neurônio corresponde ao peso de bias.

     Considere a seguinte rede de exemplo:

     Os seus pesos seriam codificados no arquivo *initial_ weights.txt* da seguinte forma:

     ```
     0.4, 0.1; 0.3, 0.2
     0.7, 0.5, 0.6
     ```

3. *dataset.txt*: o arquivo com um conjunto de treinamento. Cada linha representa uma instância e cada coluna representa um atributo. Após listados todos os valores dos atributos, há um ponto-e-vírgula, e os demais valores na linha indicam as saídas da rede para aquela instância. Por exemplo, uma instância com dois atributos (0.5 e 0.7) e duas saídas (1.0 e 0.1) seria codificada pela linha

```
0.5, 0.7; 1.0, 0.0
```

O seu programa deverá gerar uma saída no mesmo formato utilizado pelo arquivo *initial_ weights.txt*, mas indicando, para cada peso, o seu gradiente considerando os pesos iniciais da rede, o fator de regularização sendo utilizado, e o conjunto de treinamento. Os gradientes devem ser impressos usando 5 casas decimais. Cada grupo deve indicar em seu relatório a linha de comando para execução desta função. A função será testada pelos professores a fim de checar a corretude da implementação do algoritmo; a corretude da implementação da rede (independente da sua performance quando aplicada aos conjuntos de dados) é um quesito central na avaliação do trabalho;

- Cada grupo deverá prover uma função que implemente a *verificação numérica* do gradiente calculado por sua rede. O seu verificador numérico deve poder ser chamado pela linha de comando da mesma forma que o verificador de *backpropagation*, conforme descrito acima. Cada grupo deve indicar em seu relatório a linha de comando para execução desta função. A função será testada pelos professores a fim de checar a corretude da implementação do algoritmo; a corretude da implementação da rede (independente da sua performance quando aplicada aos conjuntos de dados) é um quesito central na avaliação do trabalho;

- Após verificada a corretude do *backpropagation*, cada grupo deverá treinar a sua rede (e avaliar a sua performance) nos seguintes *datasets* básicos de classificação:

  1. Pima Indian Diabetes Data Set *(8 atributos, 768 exemplos, 2 classes)*
     https://github.com/EpistasisLab/penn-ml-benchmarks/tree/master/datasets/classification/pima
     Objetivo: predizer se um paciente tem diabetes a partir de um pequeno conjunto de dados clínicos.
  2. Wine Data Set *(13 atributos, 178 exemplos, 3 classes)*
     https://archive.ics.uci.edu/ml/datasets/wine
     Objetivo: predizer o tipo de um vinho baseado em sua composição química.
  3. Ionosphere Data Set *(34 atributos, 351 exemplos, 2 classes)*
     https://archive.ics.uci.edu/ml/datasets/Ionosphere
     Objetivo: predizer se a captura de sinais de um radar da ionosfera é adequada para análises posteriores ou não (*'good'* ou *'bad'*)
  4. (opcional; pontos extras) Breast Cancer Wisconsin *(32 atributos, 569 exemplos, 2 classes)*
     https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)
     Objetivo: predizer se um determinado exame médico indica ou não a presença de câncer.

- Para cada *dataset* citado acima, utilize o método de **validação cruzada estratificada** para treinar e testar o algoritmo de redes neurais. O objetivo é comparar a performance de redes com diferentes arquiteturas e diferentes valores do parâmetro de regularização λ.

- Sugere-se utilizar $k = 10$ folds na validação cruzada. Destes k folds, $k - 1$ folds serão usados para treinamento da rede, aplicando-se o algoritmo de *backpropagation*. O fold restante (de teste) será usado para avaliar o desempenho da rede. Este processo de treinamento e teste será repetido k vezes, de acordo com o funcionamento básico da estratégia de validação cruzada visto em aula.

- Como medida de performance utilizada neste trabalho, indicamos o uso da **F1-measure**, definida em função da precisão e do recall (atribuindo mesmo peso a ambas, isto é, $\beta = 1$).

- Indique em seu relatório, através de gráficos ou tabelas, o desempenho do algoritmo (para cada *dataset*) para diferentes arquiteturas de rede e diferentes valores de regularização λ. Tais análises possibilitarão a escolha da melhor arquitetura cada *dataset*.

- Após identificada a melhor configuração da rede neural para cada um dos *datasets*, utilize essa rede para gerar um gráfico de aprendizado no qual você apresenta (no eixo y) a performance da rede em um conjunto de teste (i.e., o valor da função de custo J neste conjunto de dados) em função do número de exemplos de treinamento apresentados (no eixo x). Ou seja: após apresentados 5 exemplos, qual a performance da rede no conjunto de teste? Após apresentados 10 exemplos, qual a performance da rede no conjunto de teste? e assim por diante. Em uma rede com parâmetros bem ajustados, este gráfico indicará que a performance da rede cresce com o número de exemplos dados como treinamento; ou, de forma equivalente, que o custo J decresce com o número de exemplos de treinamento.

- Embora não seja essencial, nós recomendamos que o treinamento seja feito utilizando a metodologia de *mini-batch* (para um tamanho de *mini-batch* a ser escolhido pelo grupo), a qual acelera significativamente a convergência do algoritmo.

- Algumas dicas:

  1. Inicialize os pesos da rede com valores aleatórios pequenos entre -1 e +1, ou então com números amostrados de uma distribuição normal (Gaussiana) de média 0 e variância 1;
  2. Inicie seus testes com um número pequeno de camadas (inicialmente com apenas uma camada oculta; aumente o número de camadas apenas caso necessário);
  3. Para uma rede com um determinado número de camadas, inicie seus testes utilizando taxas de aprendizado (parâmetro α) altas. Taxas muito pequenas podem causar convergência prematura para uma solução que é um mínimo local ruim. Caso a performance da rede durante o treinamento oscile demasiadamente para tal valor de α, pode estar ocorrendo o fenômeno de *overshooting*; nesse caso, diminua o valor de α;
  4. Após encontrar um valor de α que faz a performance da rede crescer de forma consistente e estabilizar, verifique se tal performance é suficiente boa. Caso uma rede com poucas camadas tenha convergido para uma solução de baixa performance (*underfitting*), tente aumentar o número de camadas e repita a análise descrita anteriormente relativa à avaliação de diferentes valores de taxa de aprendizado;

- Resumo dos itens/requisitos descritos acima, e que serão explicitamente avaliados:

  1. **(obrigatório)** Verificação numérica do gradiente;
  2. **(obrigatório)** Interface de linha de comando para teste do *backpropagation*;
  3. **(obrigatório)** *Backpropagation* implementado de forma completa e correta;
  4. **(obrigatório)** Suporte a regularização;
  5. **(obrigatório)** k-folds+F1 score;
  6. **(obrigatório)** Testes (para cada dataset) com diferentes arquiteturas de rede;
  7. **(obrigatório)** Testes (para cada dataset) com diferentes meta-parâmetros: taxa de aprendizado, taxa de regularização, etc;
  8. **(obrigatório)** Curvas de aprendizado mostrando a função de custo J para cada dataset;
  9. **(obrigatório)** Análise dos datasets 1-3;
  10. **(obrigatório)** Pontualidade na entrega do trabalho. Atrasos na entrega do trabalho serão penalizados proporcionalmente ao tempo de atraso, sendo descontado 1 (um) ponto por dia de atraso (o trabalho como um todo vale 10 pontos);
  11. **(obrigatório)** Apresentação oral dos resultados: qualidade da apresentação e domínio da implementação e resultados, bem como capacidade de arguição acerca dos mesmos;
  12. **(obrigatório)** Qualidade do relatório final;
  13. *(opcional) Análise do dataset 4;*
  14. *(opcional) Suporte a treinamento mini-batch;*
  15. *(opcional) Suporte a vetorização;*
  16. *(opcional) Suporte ao método do momento.*

3 Política de Plágio

Grupos poderão **apenas** discutir questões de *alto nível* relativas a resolução do problema em questão. Poderão discutir, por exemplo, quais técnicas de normalização estão sendo utilizadas, e dificuldades gerais encontradas ao se analisar cada *dataset*. **Não** é permitido que os grupos utilizem quaisquer códigos fonte provido por outros grupos, ou encontrados na internet. Os alunos **poderão** fazer consultas na internet ou em livros **apenas** para estudar o modo de funcionamento das técnicas de redes neurais, e para analisar o **pseudo-código** que as implementa. **Não** é permitida a análise ou cópia de implementações concretas (em quaisquer linguagens de programação) da técnica escolhida. O objetivo deste trabalho é justamente implementar a técnica do zero e descobrir as dificuldades envolvidas na sua utilização para resolução de um problema de aprendizado. Toda e qualquer fonte consultada pelo grupo (tanto para estudar os métodos a serem utilizados, quanto para verificar a estruturação da técnica em termos de *pseudo-código*) **precisa obrigatoriamente** ser citada no relatório final. Os professores utilizam rotineiramente um sistema anti-plágio que compara o código-fonte desenvolvido pelos grupos com soluções enviadas em edições passadas da disciplina, e também com implementações conhecidas e disponíveis online.

Qualquer nível de plágio (ou seja, utilização de implementações que não tenham sido 100% desenvolvidas pelo grupo) poderá resultar em **nota zero** no trabalho. Caso a cópia tenha sido feita de outro grupo da disciplina, *todos* os alunos envolvidos (não apenas os que copiaram) serão penalizados. Esta política de avaliação **não** é aberta a debate posterior. Se você tiver quaisquer dúvidas sobre se uma determinada prática pode ou não, ser considerada plágio, ***não*** **assuma** nada: pergunte ao professor. Os grupos deverão desenvolver o trabalho **sozinhos**.