UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INSTITUTO DE INFORMÁTICA

INF01017 Aprendizado de Máquina

Trabalho 2: Redes Neurais com backpropagation

Augusto Bennemann

Fabrício Martins Mazzola

Porto Alegre, 21 de Junho de 2018

Introdução

Neste trabalho, exploramos a técnica de aprendizado de máquina por meio de redes neurais. Para treinar o algoritmo e analisar seu desempenho utilizamos os quatro *datasets* propostos.

Inicialmente, fizemos um pré-processamento nos conjuntos de dados para padronizá-los de acordo com o formato de entrada que especificamos para nosso programa. Essas alterações foram, basicamente, incluir o nome das colunas na primeira linha e mover a coluna da classe para a última posição.

O algoritmo foi desenvolvido na linguagem Python 3. Junto a este relatório há a versão final do código (*t2.py*) e o verificador de *backpropagation*, que pode ser executado com a linha de comando *python3 backpropagation.py network.txt initial_weights.txt dataset.txt*

Estruturas de dados

Para implementar a rede neural, criamos três classes diferentes: Neuron, Layer e Network. Essas classes definem, respectivamente, um neurônio, uma camada da rede e a rede neural completa.

Um objeto Neuron armazena informações do delta, da função de ativação e os pesos que chegam nesse neurônio. Tanto os pesos como os deltas são armazenados em listas. A classe Layer é composta por uma lista de objetos Neuron. Já a classe Network é composta por uma lista de objetos Layer. Além disso, a classe Neuron armazena informações de fator de regularização (λ) e do passo de atualização (α). A classe Network implementa os métodos de backtracking, cálculo de erros da rede, função de ativação e atualização de pesos.

```
class Neuron:
  delta = 0
  activation = 0
  weights = []

  def __init__(self, initialWeights):
    self.weights = initialWeights
    self.delta = 0
    self.activation = 0

  def setDelta(self, delta):
    self.delta = delta

  def getDelta(self):
    return self.delta

  def __str__(self):
    return "  ".join("%.5f" % w for w in self.weights)
```

Definição da classe Neuron

```
class Layer:
  neurons = []

  def __init__(self, weights):
    self.neurons = []
    for w in weights:
      self.neurons.append(Neuron(w))

  def __str__(self):
    msg = ""
    for i, n in enumerate(self.neurons):
      msg += "\t%s\n" % n
    return msg
```

Definição da classe Layer

```
class Network:
  regularizationFactor = 0
  alpha = 0
  layers = []

  def __init__(self, regularizationFactor, layersWeights, inputSize, alpha=0.0001):
    print("Inicializando rede com a seguinte estrutura de neuronios por camadas: [%d %s]" % (inputSize, " ".join("%d" % len(a) for a in la

    self.regularizationFactor = regularizationFactor
    self.alpha = alpha
    self.layers = []
    for lw in layersWeights:
      self.layers.append(Layer(lw))

  def __str__(self):
    msg = "Parametro de regularizacao lambda=%.3f\n" % self.regularizationFactor
    for i, l in enumerate(self.layers):
      msg += ("\nTheta%d inicial (pesos de cada neuronio, incluindo bias, armazenados nas linhas):\n" % (i+1))
      msg += str(l)
    return msg
```

Definição da classe Network

Características Gerais da Implementação

A implementação desenvolvida foi feita utilizando diversas técnicas para aumentar o desempenho das redes neurais, como uso de vetorização, validação cruzada e a metodologia *mini-batch*. A técnica de vetorização permite que os cálculos, tanto dos gradientes como dos erros de cada neurônio, sejam feitos de maneira muito mais rápida quando comparada à tradicional, pois utiliza operações de matrizes que são mais eficientes.

A validação cruzada estratificada foi realizada com *k = 10 folds*, conforme sugerido. Essa técnica permite que a rede neural seja generalizável. Além disso, foi utilizada a metodologia de treinamento de *mini-batch*. O *mini-batch* desenvolvido permite a escolha do valor do *batch* (K) por parâmetro, ao chamar a função de abertura do dataset escolhido. Para o treinamento das redes neurais aqui apresentadas, utilizamos o tamanho do *batch* como K = 20.

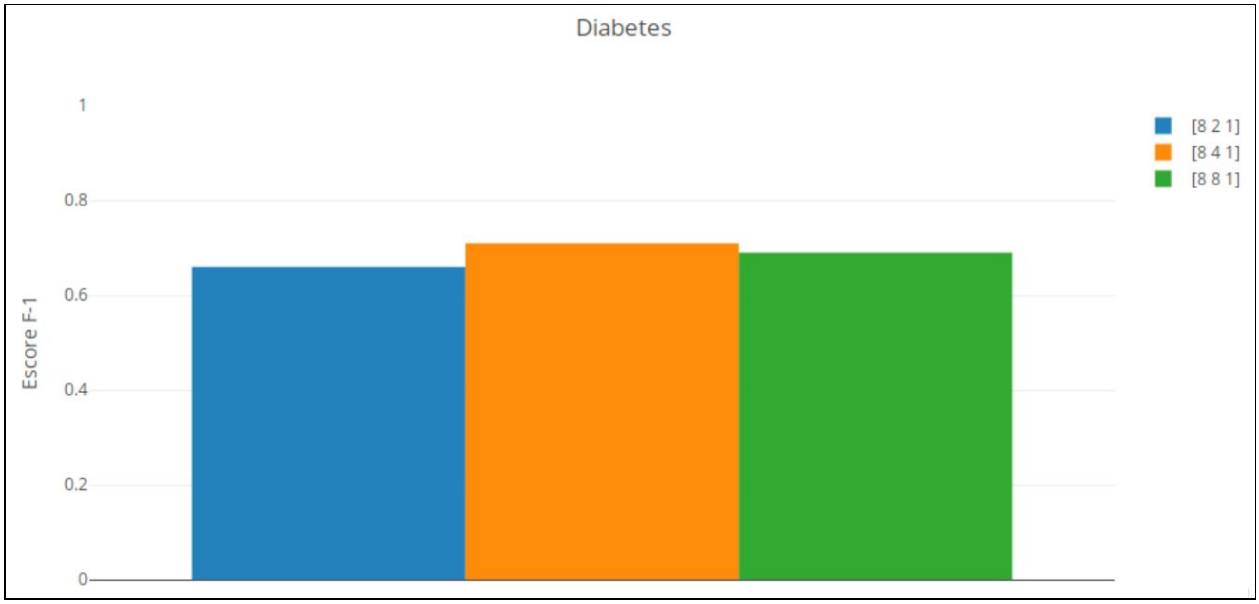
A técnica mini-batch reúne as vantagens presentes na metodologia estocástica e na técnica de batch. Essa otimização evita que os pesos da rede sejam alterados a cada instância de treinamento que passa pela rede (batch) ou que os pesos somente sejam atualizados uma única vez (estocástica), diminuindo consideravelmente o tempo de convergência.

Análise de desempenho

Para avaliar o desempenho da rede neural, testamos os quatro conjuntos de dados. Como métrica, foi utilizada a F1-measure para todos os datasets.

1. Diabetes

F1 \ Rede Neural	**[8 2 1]**	**[8 4 1]**	**[8 8 1]**
Média	0.66	0.71	0.69
Desvio	0.07	0.05	0.05

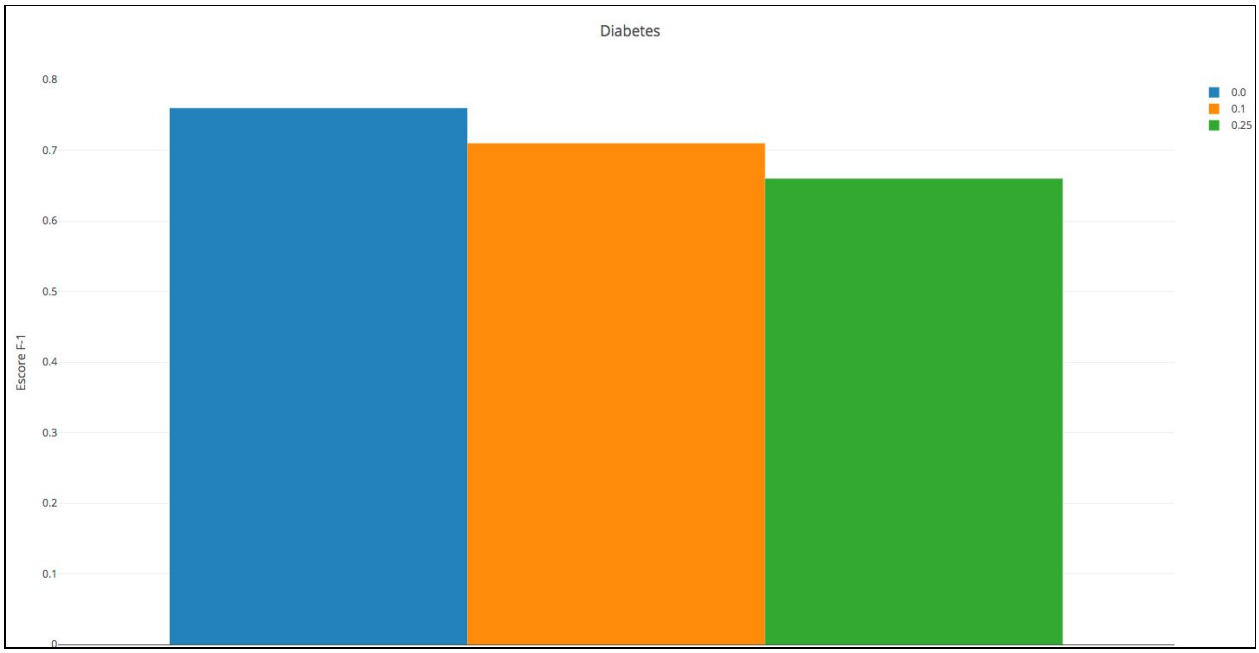


Variação da estrutura da rede

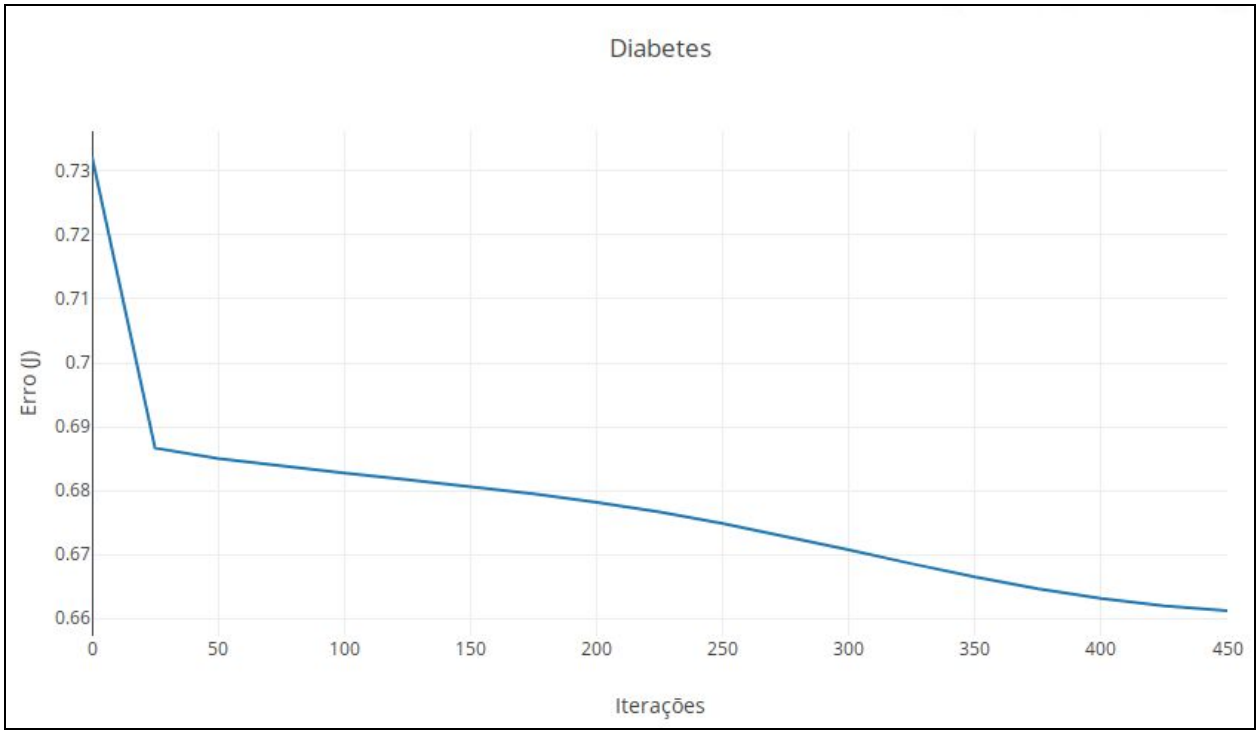
Esse conjunto de dados foi executado com os parâmetros $\alpha = 0.05$, $\lambda = 0.1$ e com 700 iterações. Os mesmos parâmetros foram utilizados nas três redes avaliadas.

Pode-se perceber que independente do número de neurônios presentes na camada intermediária, o desempenho do algoritmo variou pouco para esse conjunto de dados. Uma rede neural com duas camadas intermediárias foi testada e obteve resultado similar aos mostrados no gráfico acima, e por isso, seus resultados foram omitidos. Além da baixa variação de desempenho, pode-se observar que essa variação

ocorreu sem seguir uma progressão. Por esse motivo, não está evidente se a variância de rede neural teve alguma relevância estatística ou se as diferenças são apenas efeitos da estrutura da rede neural.



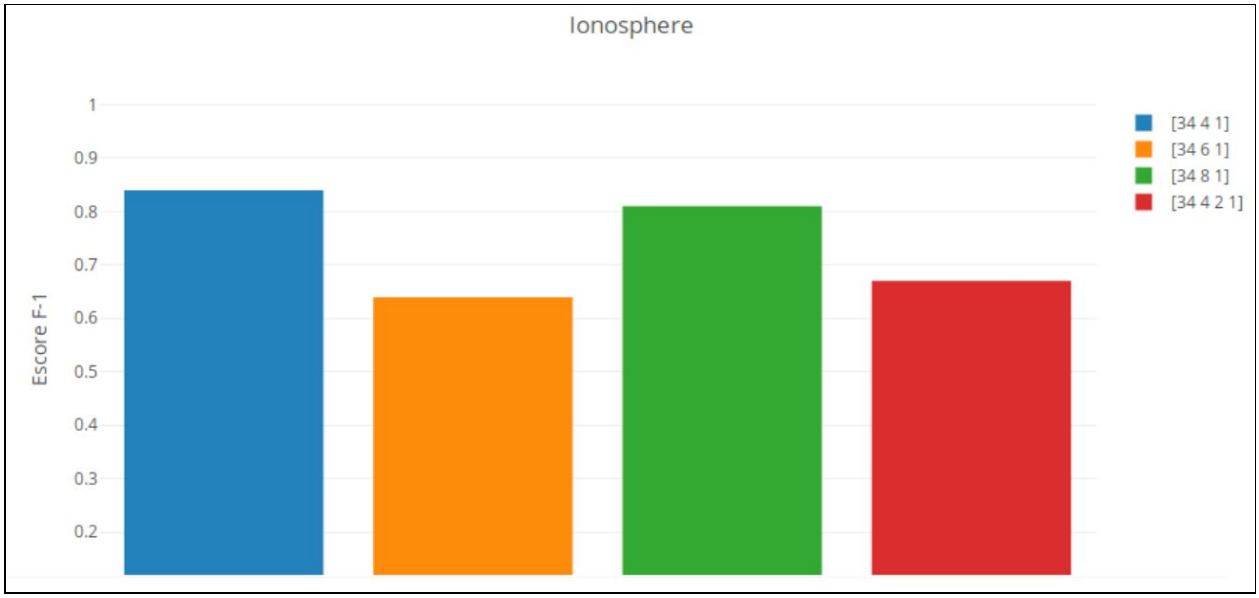
Variação do fator de regularização



Erro da rede (J)

2. Ionosphere

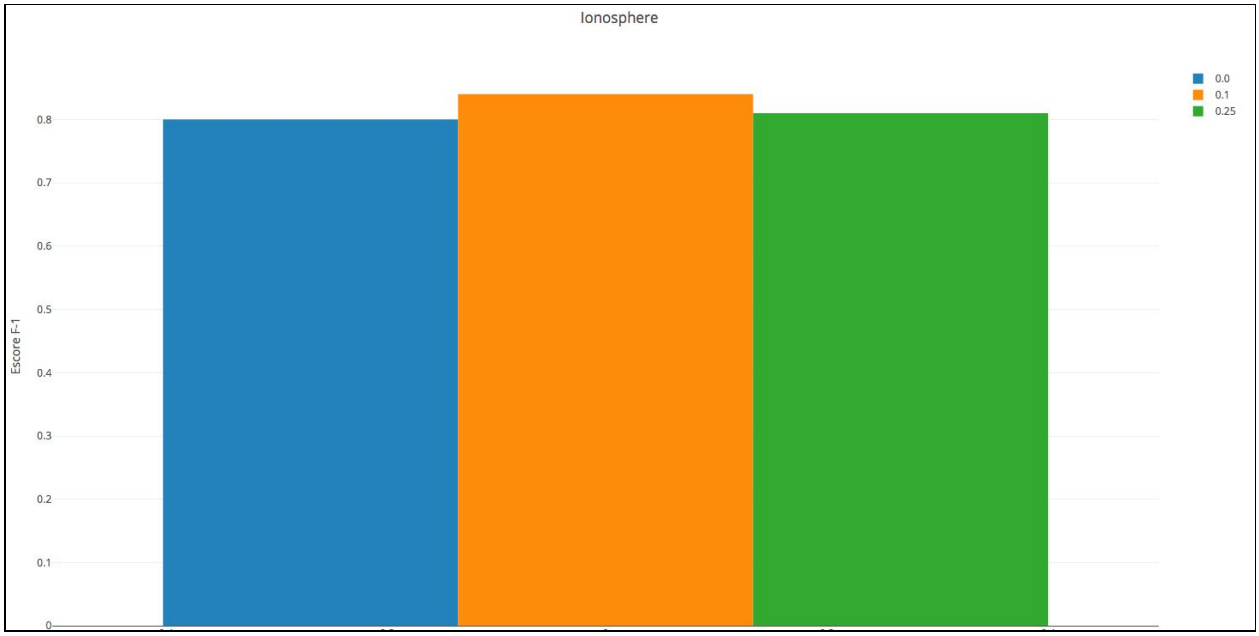
F1 \ Rede Neural	**[34 4 1]**	**[34 6 1]**	**[34 8 1]**	**[34 4 2 1]**
Média	0.84	0.64	0.81	0.67
Desvio	0.08	0.08	0.13	0.0



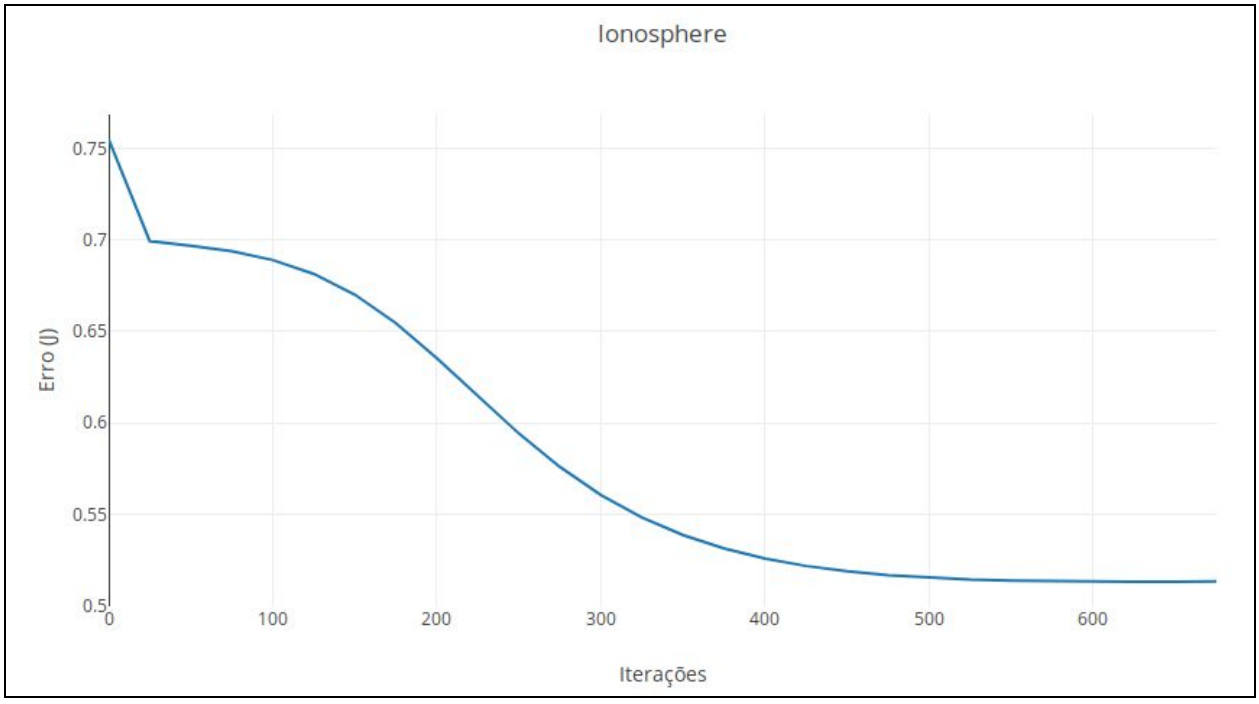
Variação da estrutura da rede

Esse conjunto de dados foi executado com os parâmetros $\alpha = 0.05$, $\lambda = 0.1$ e com 700 iterações. Os mesmos parâmetros foram utilizados nas três redes avaliadas.

O dataset Ionosphere apresentou a maior variação de desempenho entre as diferentes redes avaliadas. Pode-se perceber que o desempenho variou de acordo com o número de neurônios presentes na camada intermediária. A rede que possui 4 neurônios na sua única camada intermediária apresentou o melhor desempenho em relação às demais. Já a versão com 6 neurônios na sua única camada intermediária apresentou o pior desempenho. Além disso, é possível observar que a adição de uma nova camada intermediárias não traz grandes ganhos de performance, em relação às redes com camada intermediária única.



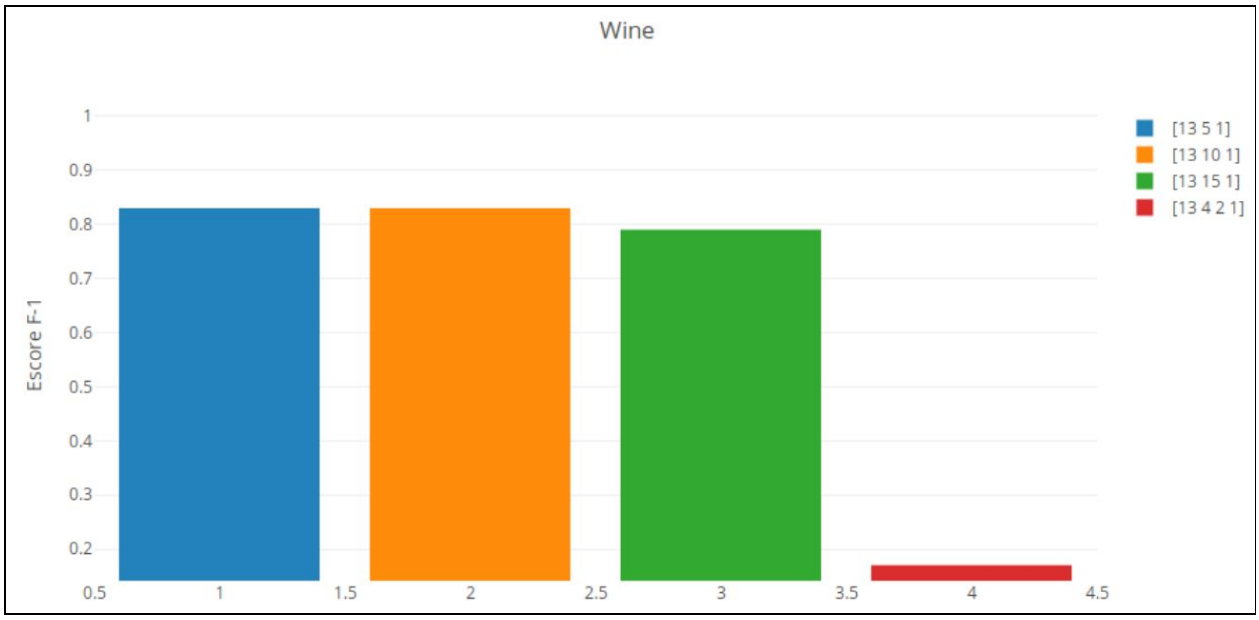
Variação do fator de regularização



Erro da rede (J)

3. Wine

F1 / Rede Neural	[13 5 1]	[13 10 1]	[13 15 1]	[13 4 2 1]
Média	0.83	0.83	0.79	0.17
Desvio	0.14	0.08	0.10	0.00

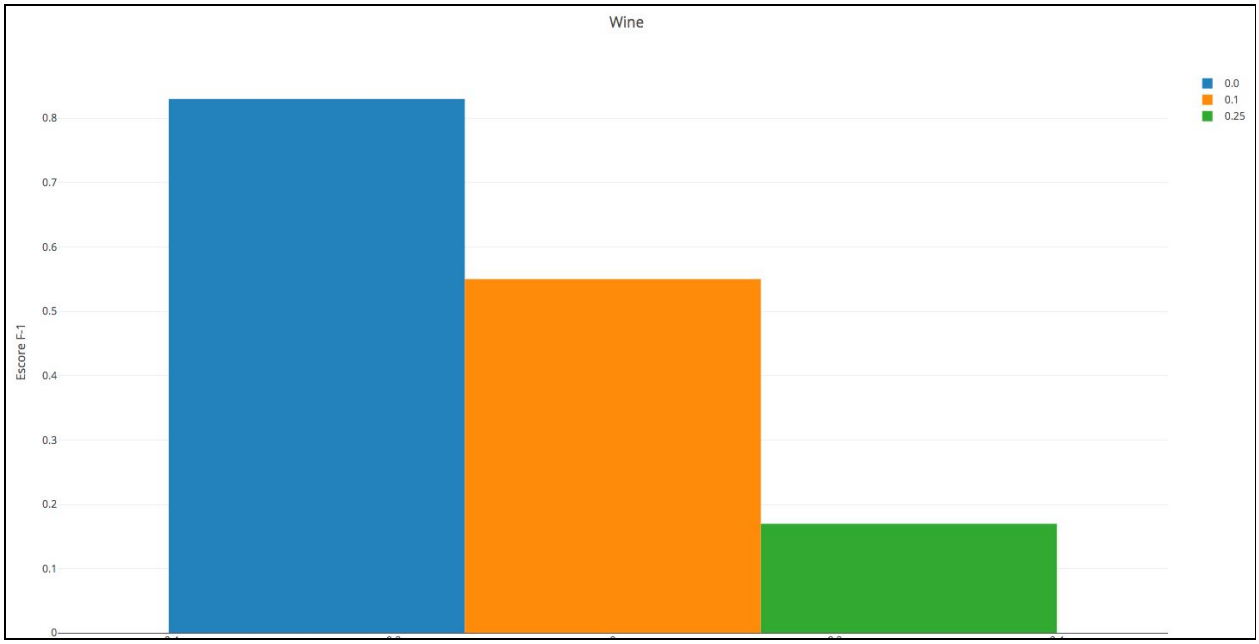


Variação da estrutura da rede

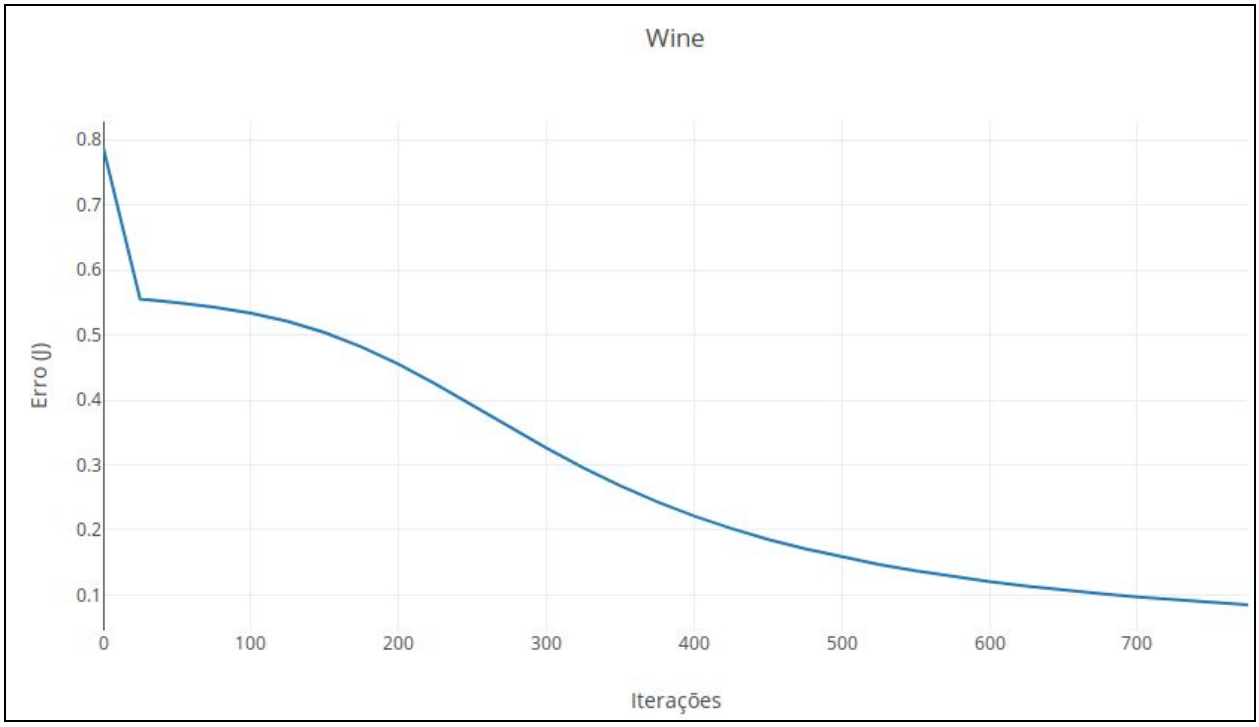
Esse conjunto de dados foi executado com os parâmetros α = 0.05, λ = 0 e com 800 iterações. Os mesmos parâmetros foram utilizados nas três redes avaliadas.

O dataset Wine apresentou uma variação praticamente negligível de desempenho entre as diferentes redes com uma única camada intermediária avaliadas. Pode-se perceber que o desempenho é inversamente proporcional ao número de neurônios presentes na camada intermediária. A rede que possui duas camadas intermediárias apresentou o pior desempenho em relação às demais, com um escore F-1 até quatro vezes menor quando comparado com redes que possuem uma única camada oculta.

Acreditamos que essa queda de desempenho se dá pelas características do dataset. Dado que o conjunto de dados possui poucas entradas, a adição de uma nova camada oculta, com mais neurônios adiciona uma quantidade não otimizada de *features* na rede, o que acaba prejudicando o desempenho.



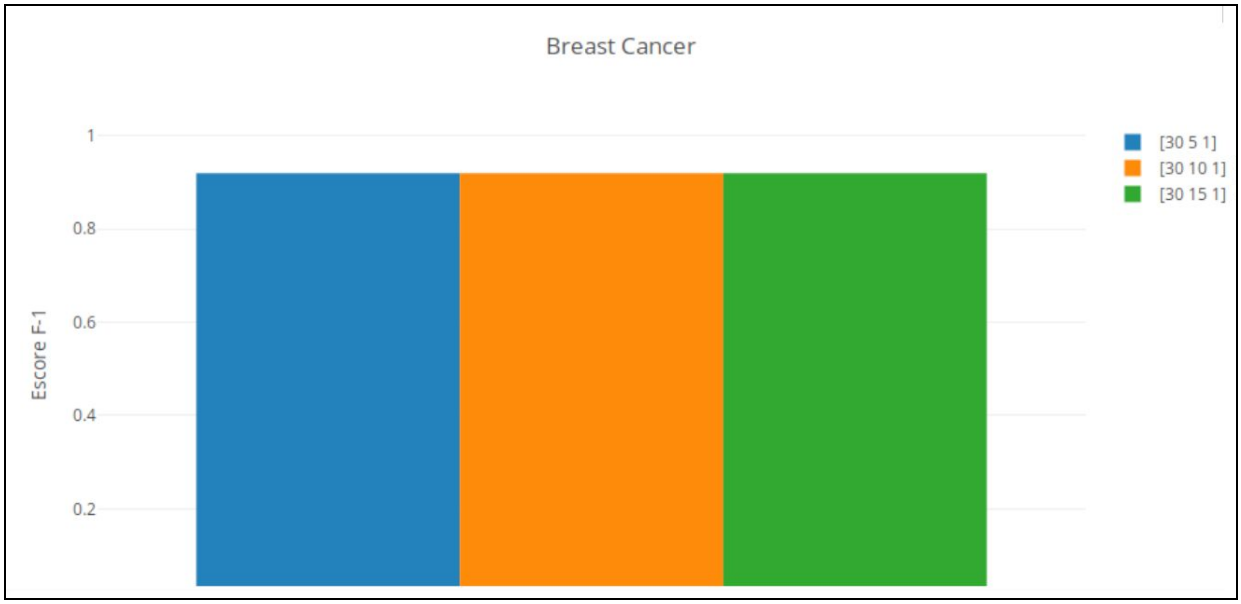
Variação do fator de regularização



Erro da rede (J)

4. Breast Cancer (Extra)

F1 \ Rede Neural	[30 5 1]	[30 10 1]	[30 15 1]
Média	0.92	0.92	0.92
Desvio	0.04	0.03	0.05



Variação da estrutura da rede

Esse conjunto de dados foi executado com os parâmetros $\alpha = 0.05$, $\lambda = 0.1$ e com 700 iterações. Os mesmos parâmetros foram utilizados nas três redes avaliadas.

O dataset Breast Cancer não apresentou uma variação de desempenho entre as diferentes redes neurais avaliadas. Acreditamos que isso aconteceu pois, independente do número de neurônios na camada oculta, a rede irá possuir um bom desempenho dadas as características de fácil aprendizado do dataset.

Conclusão

Neste trabalho, foi realizada a implementação, o treinamento e a avaliação de redes neurais para quatro conjuntos de dados distintos. Para tornar a rede mais eficiente e correta, foram utilizadas técnicas de vetorização, validação cruzada e a metodologia *mini-batch*.

Pôde-se perceber uma grande influência da estrutura da rede neural nos resultados, de forma mais acentuada para alguns datasets. Isso mostra que a escolha da rede adequada deve passar por um processo de experimentação extensivo.

Durante a avaliação das redes neurais, para os diferentes conjuntos de dados, foi esperado que o desempenho seria superior à técnica de florestas aleatórias desenvolvida no primeiro trabalho. No geral, essa hipótese mostrou-se verdadeira.

Ademais, esperava-se que a variação de parâmetros influenciasse bastante na rede final obtida. No entanto, para alguns casos a diferença de desempenho foi muito pouco perceptível.

Realizar esse trabalho foi bastante interessante e também importante, pois contribuiu para compreendermos mais a fundo vários conteúdos vistos em sala de aula.