# Computação Natural
# Trabalho prático 3 - Redes Neurais Artificiais

**Pedro Paulo Valadares Brum**

Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

pedrobrum@dcc.ufmg.br

## 1. Introdução

Nesse trabalho são colocados em prática conceitos relacionados a redes neurais. Foi implementada uma rede neural capaz de resolver um problema de classificação supervisionada utilizando a biblioteca Keras com TensorFlow. A base de dados utilizada para estudar os parâmetros da rede foi uma amostra de dados do *Sloan Digital Sky Survey*. Ela possui 5000 instâncias cada uma com 17 *features* e 1 classe. Cada instâcia pode ser classificada como uma galáxia (GALAXY), uma estrela (STAR) ou um quasar (QSO). Como podemos ver na figura 1, a maior parte das instâncias são da classe GALAXY, correspondendo a 2501 instâncias. Uma quantidade significativa das instâncias são da classe STAR, cerca de 41.7%, e apenas 8.2% das instâncias correspondem à classe QSO.

## 2. Modelagem e implementação

### 2.1. Bade de dados

Foi necessário realizar transformações nos dados de entrada para utiliza-lós na rede. Não são necessários todas as *features* para conseguir classificar um instância. Os atributos $objid$, $specobjid$ e $fiberid$ são apenas identificadores e não são necessários para classificação. Além disso, os atributos $run$, $rerun$, $camcol$ e $field$ são valores que descrevem a câmera no momento da observação do corpo celeste[1]. Além disso, temos que os atributos $objid$ e $rerun$ possuem os mesmos valores para todas as instâncias. Assim, esses atributos não oferecem nenhuma informação relevante para a rede, uma vez que não é possível separar as instância com eles.

[1] http://www.sdss3.org/dr9/imaging/imaging_basics.php

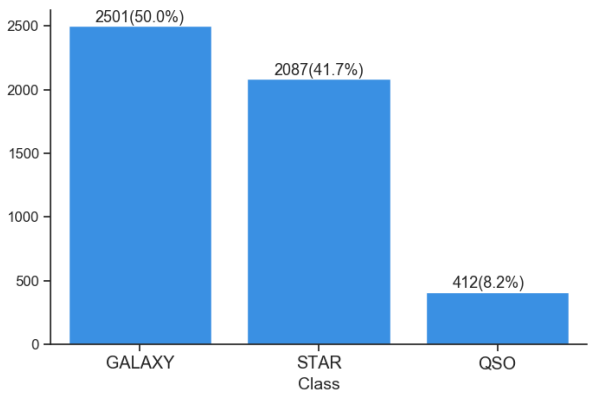


Figura 1: Distribuição das classes na amostra de dados.

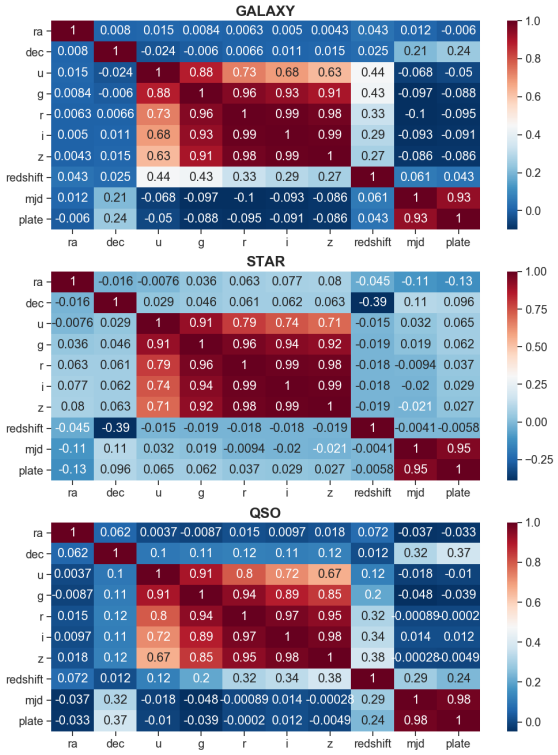


Figura 2: Correlações entre *features* para cada uma das classes.

A Figura 2 apresenta um mapa de calor com as correlações entre as *features* para cada uma das classes. Podemos observar que existe uma forte correlação entre as *features* $u$, $g$, $r$, $i$ $z$ para as três classes. Além disso, olhando para as três classes, podemos notar que existe uma grande diferença nos valores de correlação para os outros atributos $ra$, $dec$, $redshift$, $mjd$ e $plate$. Para a classe GALAXY, por exemplo, temos que a correlação entre essas *features* é muito baixa. Os valores de correlação são diferentes para as três classes e, portanto, podemos separar as instâncias olhando para os valores dos atributos selecionados. Dessa forma, para cada instância temos 10 *features*. Após a seleção das *features* foi realizado o pré-processamento dos dados. Para cada classe foi atribuido um valor entre 0 e 2. Também é realizada uma normalização dos dados, considerando apenas as 10 *features* selecionadas, utilizando o método **StandardScaler** do pacote *scikit-learn*[2].

## 2.2. Rede neural

A rede neural escolhida para criar o modelo de classificação supervisionada foi uma rede *perceptron* de múltiplas camadas (*Multi-layer Perceptron* - MLP).

Os nós de entrada da rede representam os atributos selecionados, ou seja, o número de nós na primeira camada é exatamente igual ao número de atributos utilizados para treinar a rede. Os nós de saída representam as classes do problema. Como no conjunto de dados utilizado existem três classes, teremos exatamente três nós na última camada da rede. Optou-se por adicionar uma camada escondida com 50 neurônios. A função de

[2] `https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.RandomOverSampler.html?highlight=RandomSampler`

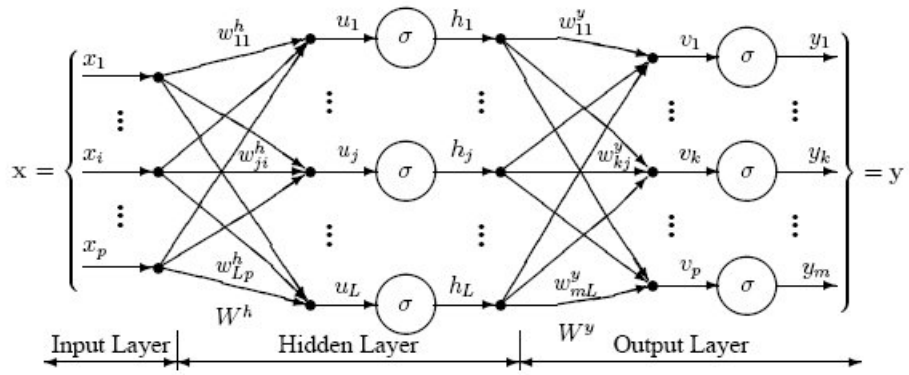


Figura 3: Arquitetura de uma rede MLP - *Multi-layer perceptron*.

ativação utilizada nos neurônios da camada escondida é a ReLu (*Rectified Linear Units*) e a função de ativação utilizada nos neurônios da camada de saída é a *softmax*.

A função de custo utilizada para calcular o erro do aprendizado é *category cros-sentropy* e o treinamento é realizado utilizando *mini-batches*.

## 3. Arquivos, compilação e execução

### 3.1. Arquivos

São partes do código-fonte do programa os seguintes arquivos:

- $main.py$ : arquivo principal de execução. Possui a implementação da rede.
- $utils.py$ : possui os métodos que salvam os resultados da classificação.

### 3.2. Compilação

A compilação do programa pode ser realizada da seguinte forma:

```
$ python main.py -g <dataset> -b <batch size>
-e <epochs> -l <learning rate> -o <arquivo de saida>
```

### 3.3. Execução

A passagem de parâmetros é opcional, e eles assumirão seus respectivos valores padrões se não forem informados:

1. Tamanho do *batch* (inteiro). Padrão: 50
2. Número de épocas (inteiro). Padrão: 50
3. Taxa de aprendizado (float). Padrão: 0.1
4. Número de neurônio na camada escondida (inteiro). Padrão: 50
5. Número de neurônios na última camada (inteiro). Padrão: 3
6. Realizar oversampling nos dados (inteiro). Padrão: 0
7. Adicionar mais uma camada escondida (inteiro). Padrão: 0

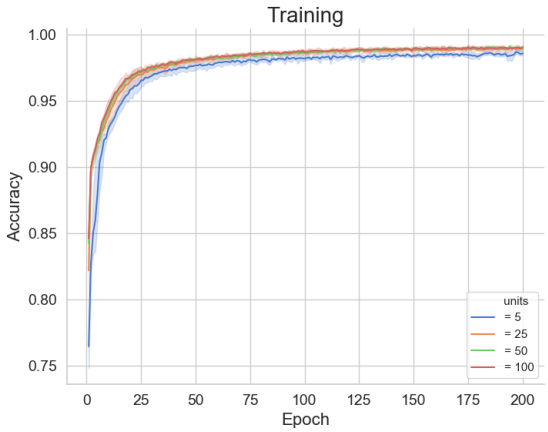


Figura 4: Valores de acurácia para diferentes números de neurônios na camada escondida.

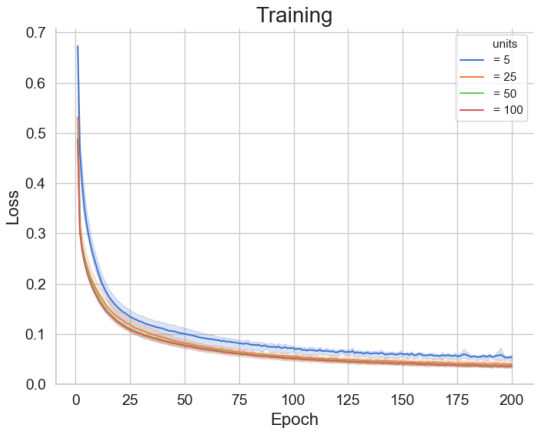


Figura 5: Valores de erro para diferentes números de neurônios na camada escondida.

## 4. Experimentos

### 4.1. Bases de dados

Como já dito anteriormente, para realizar a avaliação experimental do modelo neural artificial foi utilizada uma amostra de dados do *Sloan Digital Sky*, com 5000 instâncias que possuem 17 *features* e 1 classe cada.

### 4.2. Método

Foi realizada uma análise de sensibilidade de alguns dos principais parâmetros da rede neural: número de neurônios na camada escondida $n_h$, número de camadas escondidas $c_h$, taxa de aprendizado $lr$ e tamanho do *mini-batch* $b$. Além disso, também foi analisada como a técnica de *oversampling*, usada para balancear os dados, influencia o resultado gerado pela rede. Ao mudar um dos parâmetros, todos os outros foram mantidos constantes e para garantir que os resultados de generalização não foram obtidos ao acaso foi utilizado um procedimento de validação cruzada de 3 partições. Abaixo, temos uma lista não-exaustiva de parâmetros a avaliados:

- Número de neurônios na camada escondida
- Número de camadas escondidas
- Taxa de aprendizado
- Tamanho do *mini-batch*

### 4.3. Análise dos parâmetros

Abaixo estão descritos os experimentos realizados na base de dados bem como discussões sobre os resultados.

#### 4.3.1. Neurônios na camada escondida

Para gerar os resultados para diferentes números de neurônios na camada escondida $h$, foram mantidos o tamanho do mini-batch $b$ = 50, o número de épocas $e$ = 200, a taxa

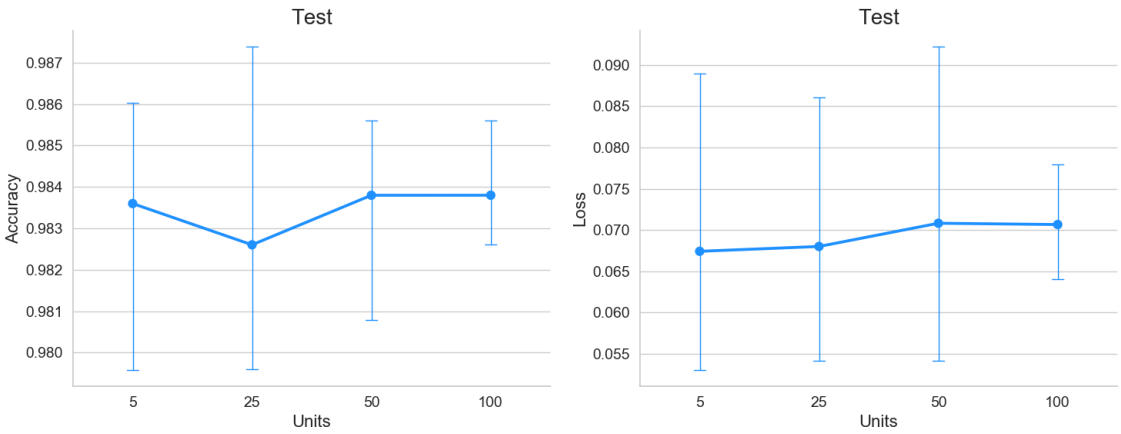


Figura 6: Valores de acurácia para diferentesFigura 7: Valores de erro para diferentes números de neurônios na camada escondida.números de neurônios na camada escondida.

de aprendizado $lr$ = 0.1 e o número de camadas escondidas $k$ = 1. Além disso, não foi realizado *oversampling* dos dados. Foram gerados resultados utilizando de 5, 25, 50 e 100 neurônios na camada escondida.

Na Figura 4 podemos observar como a acurácia varia ao longo das épocas considerando diferentes números de neurônios na camada escondida. Pode-se notar que quando utilizamos 5 neurônios a acurácia em cada época é menor em relação a quando utilizamos 25, 50 e 100 neurônios. Além disso, a acurácia não apresenta um aumento significativo com a variação do número de neurônios na camada escondida e para cada número de neurônios na camada escondida, a variação da acurácia ao longo das épocas é muito similar. O mesmo ocorre para a taxa de erro, como podemos observar na Figura 5. Temos que quando utilizamos 5 neurônios na camada escondida a taxa de erro é maior e que a erro não varia significativamente com o aumento do número de neurônios na camada escondida.

Na Figura 6 podemos observar os valores finais de acurácia obtidos na fase de teste. Temos que o maior valor de acurácia foi obtido utlizando 50 neurônios na camada escondida e que o menor foi obtido utilizando 25 neurônios. Na Figura 7 podemos observar os valores finais de erro obtidos na fase de teste. Temos que o menor valor de erro foi obtido utlizando 5 neurônios na camada escondida e que o maior foi obtido utilizando 25 neurônios.

### 4.3.2. Números de camadas escondidas

Para gerar os resultados utilizando diferentes números de camadas escondidas $k$, foram mantidos o tamanho do mini-batch $b$ = 50, o número de épocas $e$ = 200, a taxa de aprendizado $lr$ = 0.1 e o número de neurônios na primeira camada escondida $h$ = 50. Além disso, não foi realizado *oversampling* dos dados. Foram gerados resultados utilizando de 1 e 2 camadas escondidas. Para a rede com uma camada escondida, o número de neurônios na camada escondida é 50. Para a rede com duas camadas escondidas, o número de neurônios na primeira camada escondida é 50 e na segunda é 30.

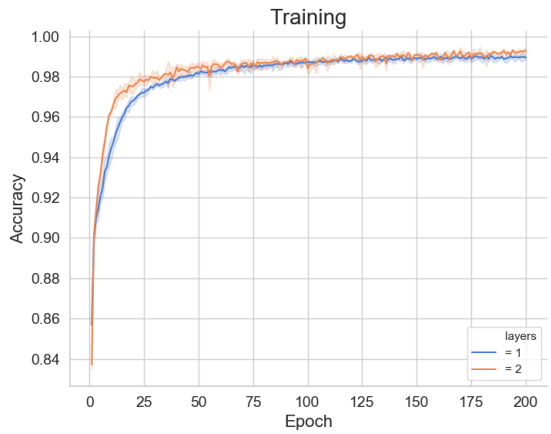


Figura 8: Valores de acurácia para diferentes números de camadas escondidas.

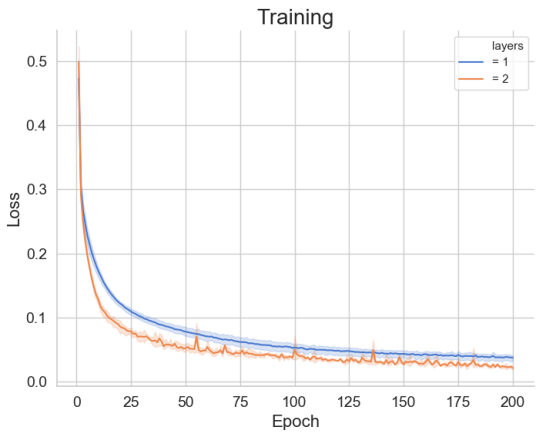


Figura 9: Valores de erro para diferentes números de camadas escondidas.

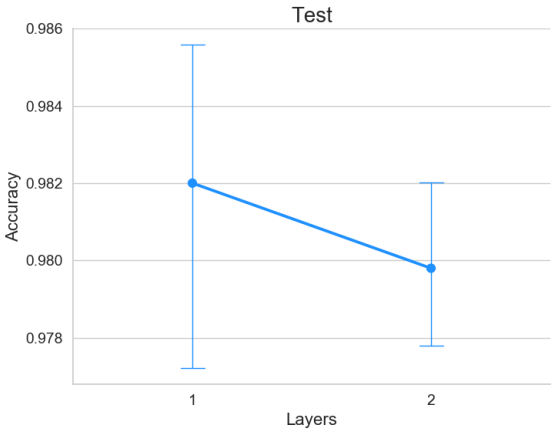


Figura 10: Valores de acurácia para diferentes números de camadas escondidas.

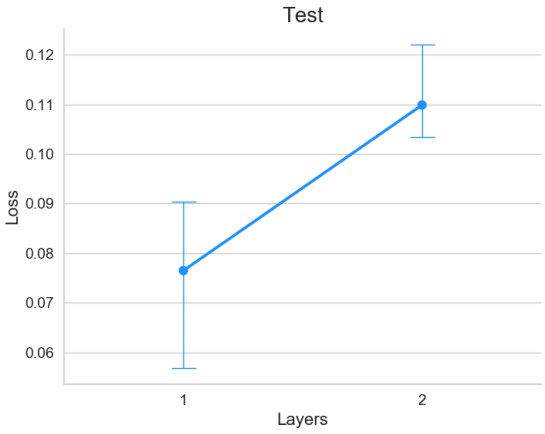


Figura 11: Valores de erro para diferentes números de camadas escondidas.

Nas Figuras 8 e 9 temos, respectivamente, os resultados de acurácia e erro obtidos ao longo das épocas na fase de treinamento utilizando diferentes número de camadas escondidas. Podemos observar que tanto os valores de acurácia como de erro são muito próxomos para 1 e 2 camadas escondidas. Além disso, podemos ver que aproximadamente na época 25 ocorre uma redução na acurácia e um aumento na taxa de erro.

Nas Figuras 10 e 11 temos, respectivamente, os resultados finais de acurácia e erro obtidos na fase de teste. Podemos observar que tanto os valores de acurácia como de erro são muito próxomos para 1 e 2 camadas escondidas. Porém é interessante observar que existe um maior desvio padrão quando consideramos 2 camadas escondidas, tanto para a acurácia como para a erro. Dessa forma, a melhor escolha é utilizar apenas uma camda escondida.

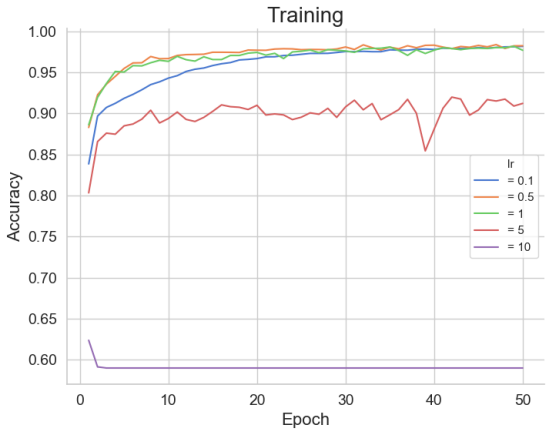


Figura 12: Valores de acurácia para diferentes taxas de aprendizado.

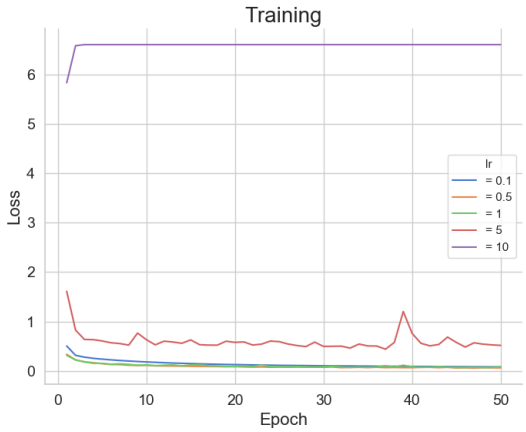


Figura 13: Valores de erro para diferentes taxas de aprendizado.

### 4.3.3. Taxa de aprendizado

Para gerar os resultados utilizando diferentes taxas de aprendizado $lr$, foram mantidos o tamanho do mini-batch $b$ = 50, o número de épocas $e$ = 50 e o número de neurônios na camada escondida $h$ = 50. Além disso, não foi realizado *oversampling* dos dados. Foram gerados resultados utilizando taxas iguais a 0.1, 0.5, 1, 5 e 10. A taxa de aprendizado controla o grau de atualização dos pesos da rede neural considerando o erro calculado pelo *gradient descent* estocástico. Quanto menor o valor da taxa de aprendiza, mais tempo gastamos para chegar até o mínimo local, ou seja, mais tempo gastamos para convergir. Porém, se a taxa de aprendizado for muito alta a convergência pode não ocorre, pois a taxa de a atualização dos pesos será sempre muito alta.

Nas Figuras 12 e 13 temos, respectivamente, os resultados de acurácia e erro obtidos ao longo das épocas na fase de treinamento utilizando diferentes taxas de aprendizado. Podemos observar que os piores valores de acurácia e erro foram obtidos utilizando uma taxa de aprendizado igual a 10 e que, a partir da época de número 20, os valores de acurácia e erro são muito próximos para as taxas de aprendizado 0.1, 0.5 e 1. É importante notar também que para as taxas de aprendizado iguais a 0.5 e 1 a convergência acontece mais rápido em relação a taxa igual 0.1.

Nas Figuras 14 e 15 temos, respectivamente, os resultados finais de acurácia e erro obtidos na fase de teste utilizando diferentes taxas de aprendizado. Podemos observar que os melhores valores de acurácia e erro foram obtidos utilizando uma taxa de aprendizado igual a 0.1 e que com o aumento da taxa os valores de acurácia e erro pioram. É importante que a taxa de aprendizado não seja nem muito alta, nem muito baixa, visto que queremos alcançar o mínimo local e obter os melhores valores de acurácia. Dessa forma, uma boa escolha para taxa de aprendizado é 0.1.

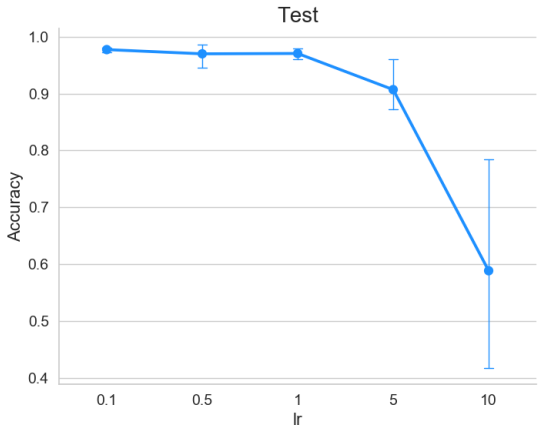


Figura 14: Valores de acurácia para diferentes taxas de aprendizado.

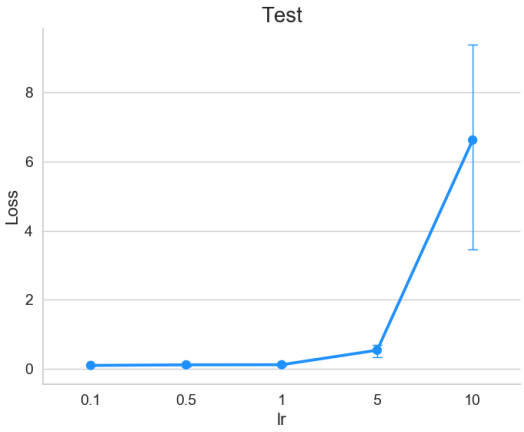


Figura 15: Valores de erro para diferentes taxas de aprendizado.

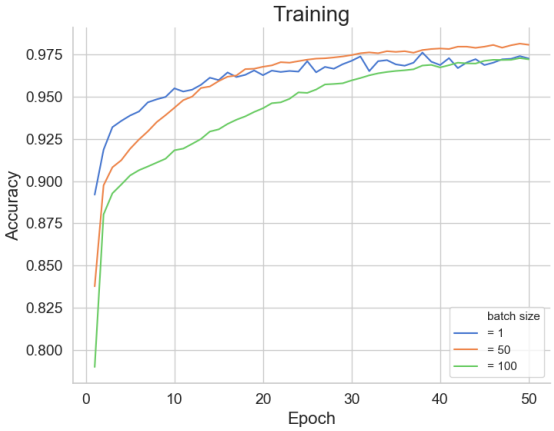


Figura 16: Valores de acurácia para diferentes tamanhos de *mini-batch*.

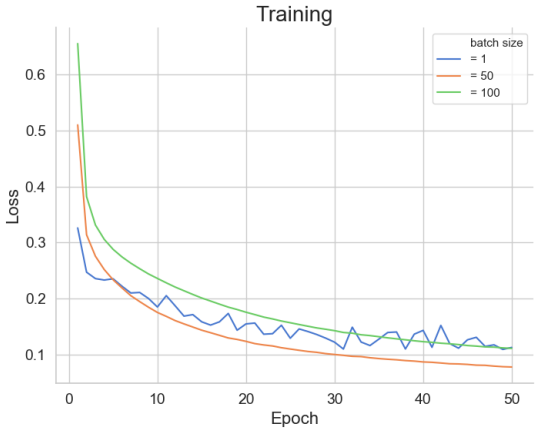


Figura 17: Valores de acurácia para diferentes tamanhos de *mini-batch*.

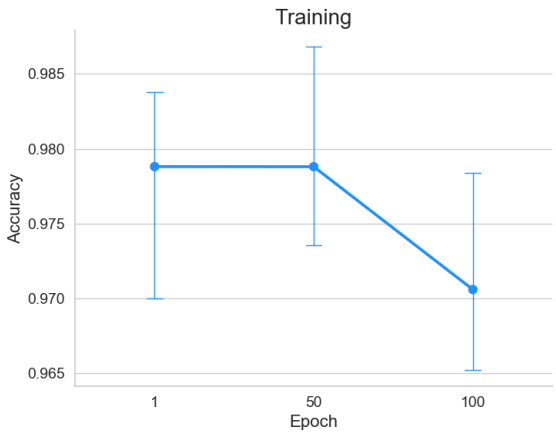


Figura 18: Valores de acurácia para diferentes tamanhos de *mini-batch*.

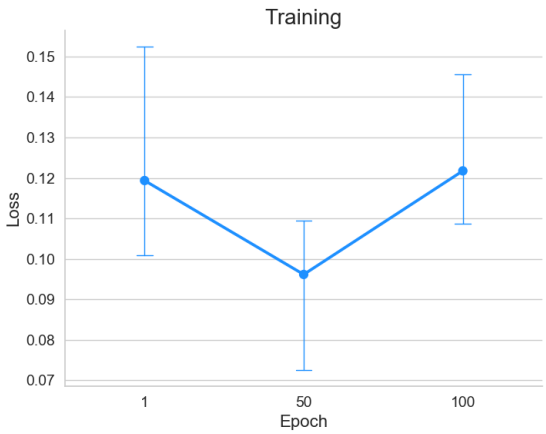


Figura 19: Valores de acurácia para diferentes tamanhos de *mini-batch*.

### 4.3.4. Tamanho do *mini-batch*

Para gerar os resultados utilizando diferentes tamanhos de *mini-batch* $b$, foram mantidos o número de épocas $e$ = 50, a taxa de aprendizado $lr$ = 0.1 e o número de neurônios na camada escondida $h$ = 50. Além disso, não foi realizado *oversampling* dos dados. Foram gerados resultados utilizando *mini-batch* de tamanho 50 e *mini-batch* de tamanho 1, que é equivalente a utilizar *gradient descent* estocástico sem *mini-batch*.

Nas figuras 16 e 17 temos os resultados de acurácia e erro, respectivamente, considerando o uso e o não uso do *mini-batch* na fase de treino. Podemos notar que quando utilizamos *gradient descent* estocástico com *mini-batch* a convergêcia da acurácia e do erro não ocorre tão rapidamente comparado a quando não utilizamos *mini-batch*. Além disso, temos que quanto maior o tamanho do *mini-batch* maioir é o número de épocas necessário para ocorrer a convergência. Observando os gráficos, temos que os melhores resultados finais foram obtidos utilizando tamanho igual a 50.

Nas figuras 18 e 19 temos os resultados finais de acurácia e erro, respectivamente. De forma análoga ao que ocorreu na fase de treino, os melhores resultados finais são obtidos com o uso de *mini-batch* de tamanho 50. Além disso, temos que o tempo computacional gasto para treinar a rede sem o uso de *mini-batch* é muito maior do que o tempo gasto quando o utilizamos um *mini-batch* de tamanho 50, aproximadamente **10 vezes maior**. Dessa forma, o uso do *gradient descent* com *mini-batch* é melhor forma de treinar a rede, visto que gastamos menos tempo e obtemos resultados melhores, tanto em relação à acurácia como em relação ao erro.

### 4.3.5. *Oversampling*

Para gerar os resultados utilizando *oversampling* foram mantidos o tamanho do mini-batch $b$ = 50, o número de épocas $e$ = 200, o número de neurônios na camada escondida $h$ = 50 e a taxa de apredizado $e$ = 0.1. Para balancear a amostra de dados utilizando a técnice

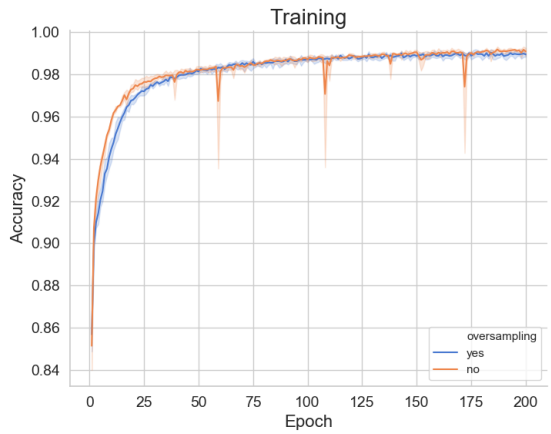


Figura 20: Valores de acurácia utilizando *oversampling*.

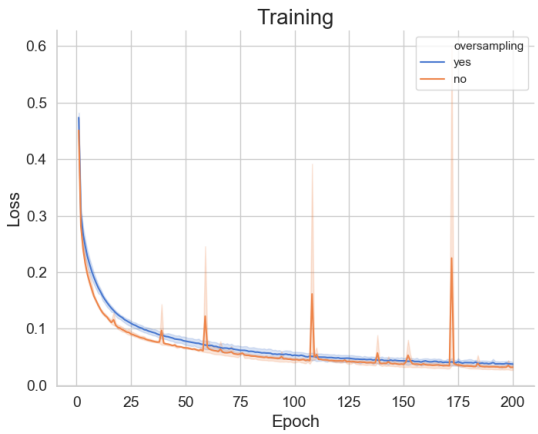


Figura 21: Valores de erro utilizando *oversampling*.

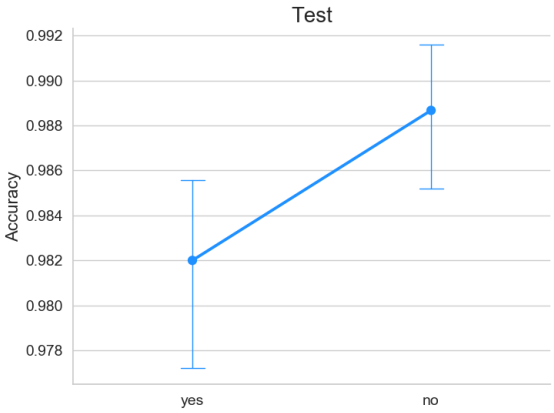


Figura 22: Valores de acurácia utilizando *oversampling*.

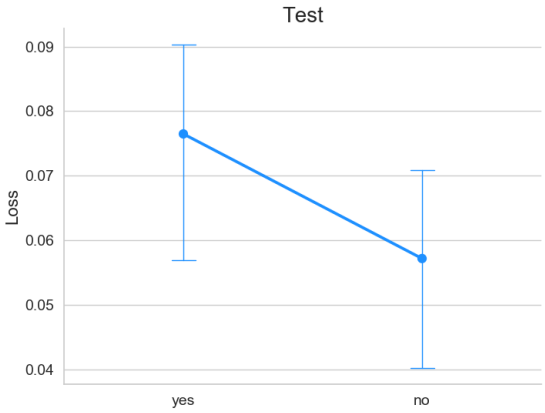


Figura 23: Valores de erro utilizando *oversampling*.

de *oversampling* foi utlizado o método **RandomOverSampler** do pacote *imblearn*[3]. A partir do *oversampling* todas as classes passam a ter o mesmo número de intâncias. Como a classe predominante possui inicialmente 2501 instâncias, temos que após o balanceamento todas as classes passa a ter 2501 instâncias.

As figuras 20 e 21 apresentam os valores de acurácia e erro, respectivamente, com e sem o uso de *oversampling*. A partir das figuras, podemos notar que o valores de acurácia e erro com e sem *oversampling* são, no geral, muito próximos ao longo das épocas. Porém, quando não utilizamos *oversampling* ocorrem algumas quedas de acurácia e picos do erros durante a fase de treinamento.

As figuras 22 e 23 apresentas os resultados finais de acurácia e erro, respectivamente. Podemos notar que quando utilizamos *oversampling* tanto o valor de acurácia como o valor de erro piora.

[3] `https://imbalanced-learn.readthedocs.io/en/stable/index.html`

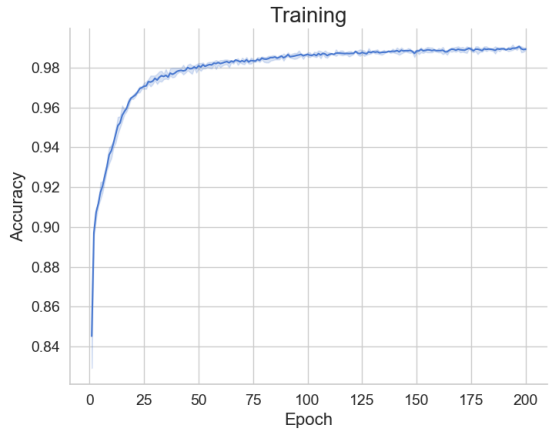


Figura 24: Valores de acurácia obtidos utilizando os parâmetros finais.

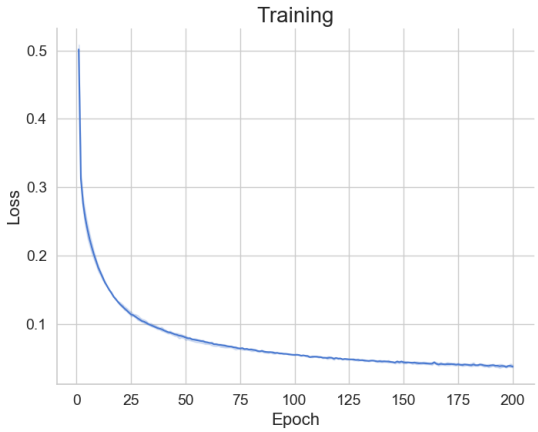


Figura 25: Valores de erro obtidos utilizando os parâmetros finais.

## 5. Resultados finais

Como resultado final temos as escolhas dos valores de cada um dos parâmetros da rede neural. Os melhores valores encontrados foram:

- Número de época: 200
- Tamanho do *mini-batch*: 50
- Taxa de aprendizado: 0.1
- Número de camadas escondidade: 1
- Número de neurônios na cam ada escondida: 30

Além disso, optou-se por não aplicar uma técnica de *oversapling*, uma vez que o erro pode aumentar. As figuras 24 e 25 apresentam os valores de acurácia e erro, respectivamente, ao longo das épocas considerando o conjunto de treinamento. As figuras 26 e 27 apresentam os valores de acurácia e erro, respectivamente, ao longo das épocas considerando o conjunto de teste. Pelos resultados apresentados nessa seção e nas seções de análise, temos que a diferença nos valores de erro na fase de treinamento e na fase de teste não é muito grande ao longo das épocas e ca última época o resultado é muito próximo.

## 6. Conclusão

Neste trabalho foi implementada uma rede neural capaz de resolver um problema de classificação. Os parâmetros da rede neural são número de épocas gastas no treinamento, taamanho do $mini - batch$, taxa de aprendizado, número de neurônios nas camadas escondidas e número de camadas escondidas. Para escolher os melhores parâmentros para a rede foi realizado um estudo considerando cada um deles individualmente. A escolha desses parâmtros não é uma tarefa fácil, visto que a modificação de apenas um deles pode alterar drasticamente o resultado. A realização de testes foi fundamental para entender a efeito de cada um dos parâmentros nos valores finais encontrados. Com a escolha dos parâmetros foram gerados os resultados para a amostra de dados *Sloan Digital Sky Survey*.

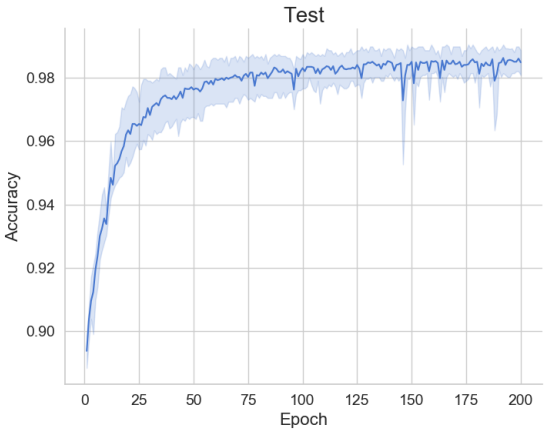


Figura 26: Valores de acurácia obtidos utilizando os parâmetros finais.

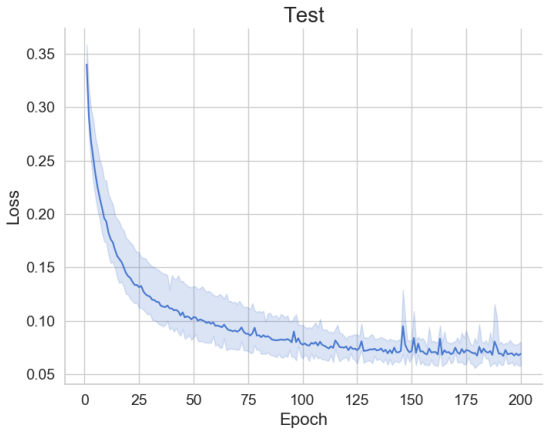


Figura 27: Valores de erro obtidos utilizando os parâmetros finais.