

## Titanic Survival - Trabalho de Inteligência Computacional - UFU 2018-1

Grupo: Bruno Borges, Eduardo Freitas, Rafael Assis

### Resumo

Neste trabalho estamos apresentando de forma sucinta um dos desafios do site Kaggle onde o objetivo é ter a maior taxa de acerto depois da rede estar treinada.

Logo vemos todo o conteúdo do dataset percebemos que alguns atributos tinha mais relevância que outros (EX: Pclass, Age, Sex, Parch e SibSp) e foi acima disso que começamos a desenvolver nosso neurônio artificial para a construção da rede.

### Introdução

Redes Neurais Artificiais são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência. Uma grande rede neural artificial pode ter centenas ou milhares de unidades de processamento; já o cérebro de um mamífero pode ter muitos bilhões de neurônios.

O sistema nervoso é formado por um conjunto extremamente complexo de células, os neurônios. Eles têm um papel essencial na determinação do funcionamento e comportamento do corpo humano e do raciocínio. Os neurônios são formados pelos dendritos, que são um conjunto de terminais de entrada, pelo corpo central, e pelos axônios que são longos terminais de saída.

Cada rede neural tem suas camadas de entrada onde cada entrada tem um peso que é reajustado de acordo com o treinamento a ser determinado, além disso temos a função de transferência que determina em qual classe o meu dado está e por fim a saída da minha rede neural.

Com isso, concluímos que através de análises quais tipos de pessoas provavelmente sobreviveriam. Em particular, aplicamos as ferramentas de aprendizado de máquina para prever quais destes passageiros sobreviveram à tragédia.

### Apresentação do Problema abordado

Para uso de rede neural foi escolhido o problema do naufrágio do Titanic do site da Kaggle, um página que publica datasets e faz competição a partir de problemas, onde os melhores modelos que resolvem o problema recebem prêmios.

Da Kaggle foi obtido um conjunto de dados (dataset) com 891 passageiros com as seguintes características:

- **PassengerId:** Número de identificação do passageiro;
- **Survived:** Indica se o passageiro sobreviveu ao desastre. É atribuído o valor de 0 para aqueles que não sobreviveram, e 1 para quem sobreviveu;
- **Pclass:** Classe na qual o passageiro viajou. É informado 1 para primeira classe; 2 para segunda; e 3 para terceira;
- **Name:** Nome do passageiro;

- **Sex:** Sexo do passageiro;
- **Age:** Idade do passageiro em anos;
- **SibSp:** Quantidade de irmãos e cônjuges a bordo ;
- **Parch:** Quantidade de pais e filhos a bordo;
- **Ticket:** Número da passagem;
- **Fare:** Preço da passagem;
- **Cabin:** Número da cabine do passageiro;
- **Embarked:** Indica o porto no qual o passageiro embarcou. Há somente três valores possíveis: Cherbourg ('C'), Queenstown ('Q') e Southampton ('S')

Eles Podem ser melhor observados na imagem abaixo

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

O objetivo do desafio é prever o 'Survived' tendo em posse somente em os outros atributos

## Arquitetura da Rede Neural

**Nodos de Entrada:** Para as entradas da rede foi escolhidos as colunas: 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch' sendo esses os nodos de entrada. Pois quanto aos outros atributos:

- Name, PassengerID, Ticket e Embarked são irrelevantes.
- Cabin em boa parte dos dados não se tinha registro, então, não dava para utilizá-lo.

## Representação das amostras:

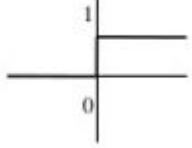
Em grande maioria, os dados são inteiros e com isso fica simples de trabalhar, porém, houve necessidade de tratamento para os seguintes dados:

- **Sex:** Foi convertido 'male' (homem) para ser 1 e 'female' (mulher) para ser 0;
- **Age:** No dataset, alguns passageiros não tinham idade. Para podermos usá-la, foi feito a média das idades de cada 'Pclass' e assim associar aos que não tinham esse valor. Dessa forma para os que não tinha 'Age':
  - Quem é da 1° Classe recebia 37 (média de idade das pessoas da 1° Classe)
  - Quem é da 2° Classe recebia 29 (média de idade das pessoas da 2° Classe)
  - Quem é da 3° Classe recebia 24 (média de idade das pessoas da 3° Classe)

**Saída da Rede:** Como o problema é uma predição da coluna 'Survived' (0 ou 1) então somente foi usado um neurônio de saída para predição binária.

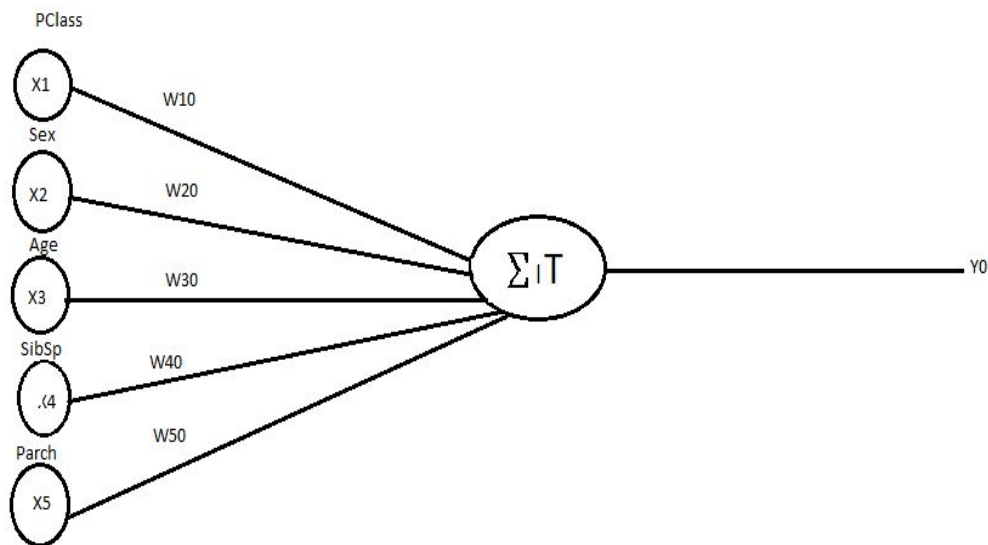
**Função de ativação e Saída da Rede:** Como o problema é uma predição Binária (Survived = 0 ou 1) 'então foi escolhido a função degrau. Aplicando sobre a soma dos dados vezes o vetor de peso, a função degrau retorna:

- 0 se se não atingiu 0 ou mais (neurônio inibido)
- 1 se é maior ou igual a 0 (neurônio ativado)

$$step_t(x) = \begin{cases} 1 & x \geq 0 \\ 0 & otherwise \end{cases}$$


### Arquitetura da Rede Neural

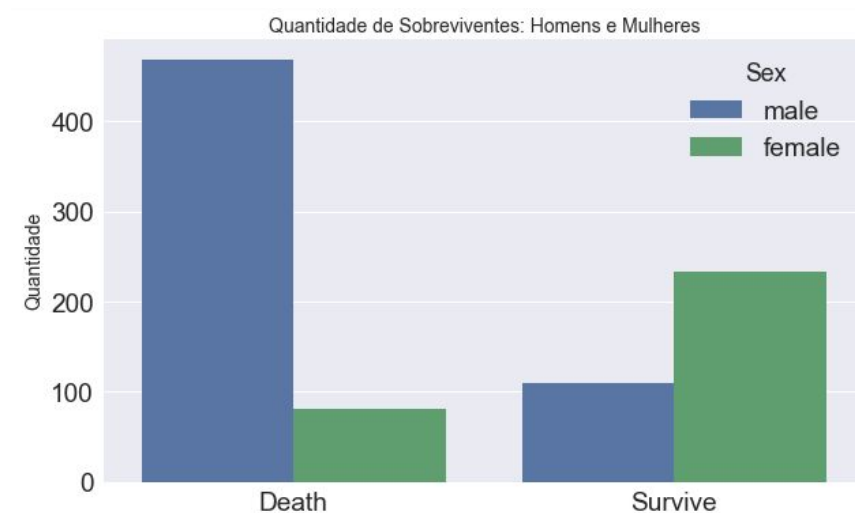
Por ser um Perceptron simples, temos somente a camada de nodos de entrada dos dados e uma camada de saída com um único neurônio. Esse esquema pode ser visto na seguinte imagem:



### Análise do Problema, da compatibilidade e da metodologia utilizada

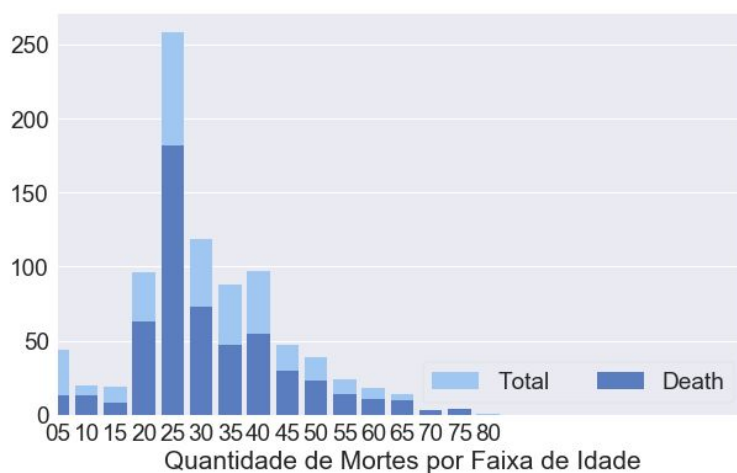
#### Análise do atributo 'Sex'

Fazendo um gráfico, observou-se que : os homens morreram muito mais do que as mulheres. Isso é uma estatística importante, que indica um padrão que será identificado pela rede (como visto nos resultados)



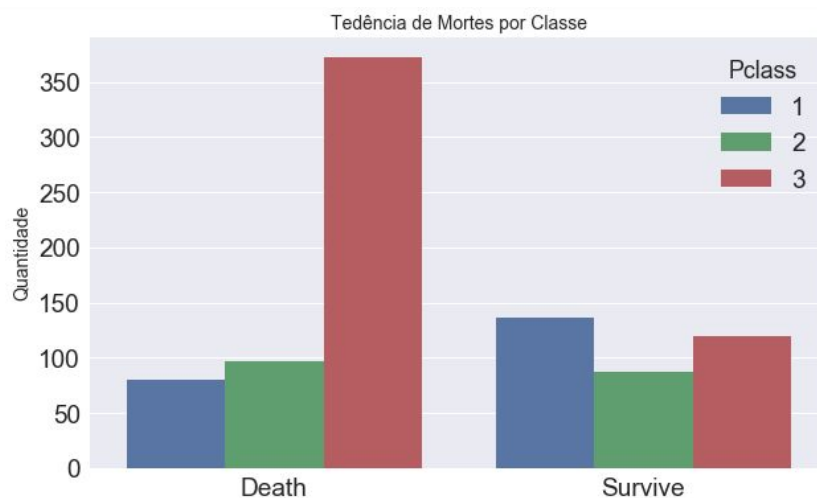
### Análise de do Atributo 'Age'

A faixa de idade é em torno de 20 à 40 anos e em geral, uma boa parte morreu para todas as idades, não se mostrando assim muito relevante.



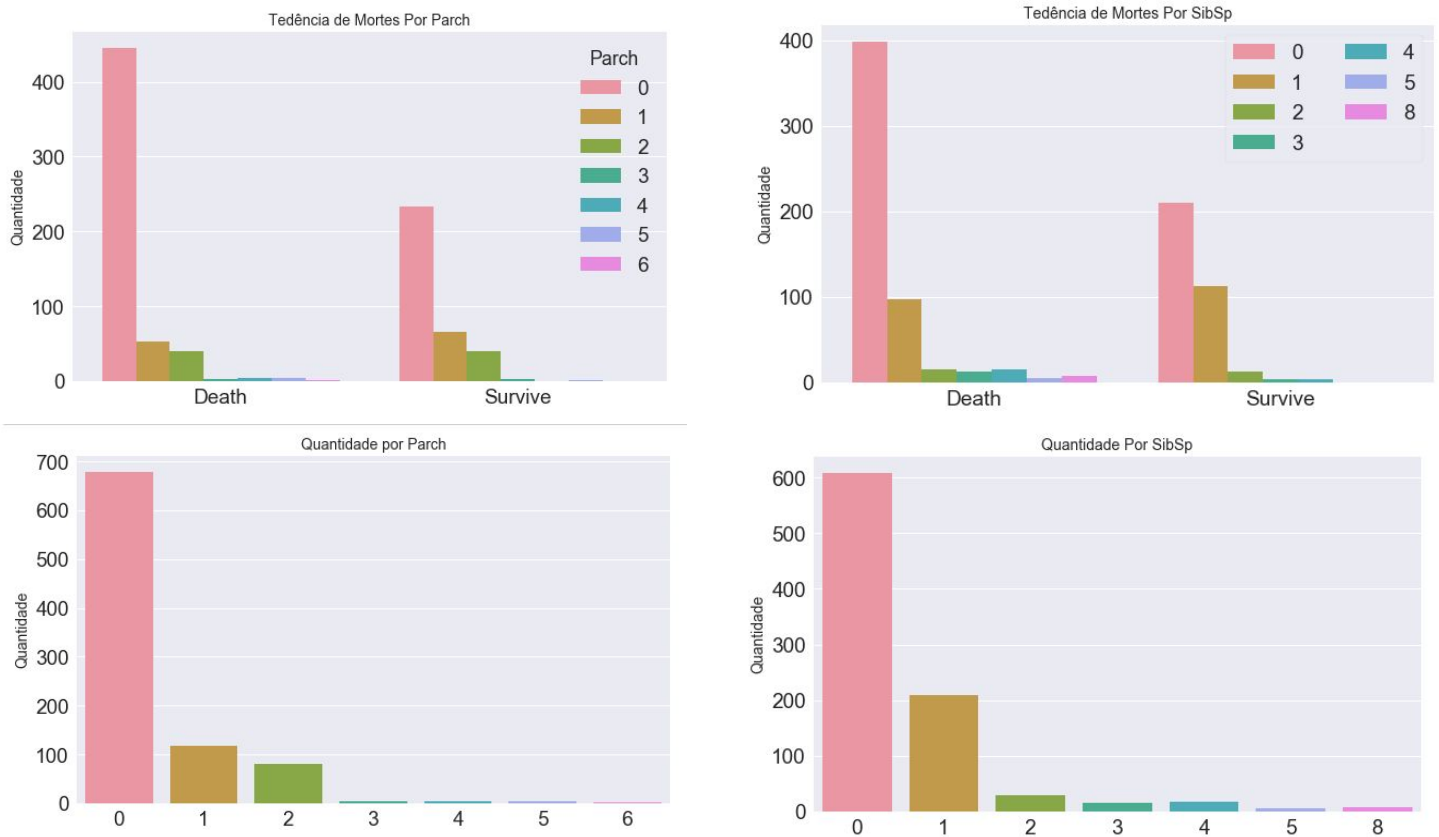
### Análise de Pclass

Ao observar o gráfico vemos a tendência de que quanto menor a classe maior é a tendência a morrer.



## Análise de SibSp e Parch

Os dois não tiveram muita influência na decisão se morreu ou não. Como é possível ver nos gráficos a seguir, a taxa de sobrevivência/morte é grande a média que a quantidade de sibSp e Parch é menor, isso porque, valores mais baixos são os mais frequentes.



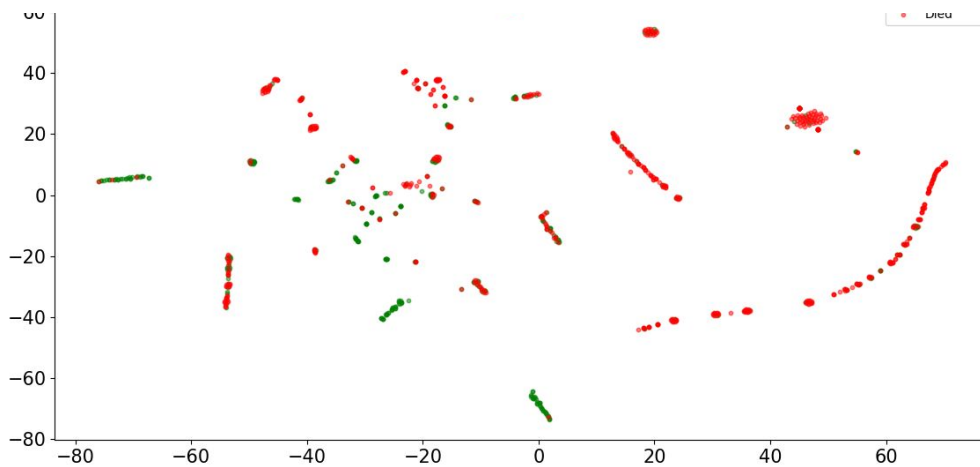
## t-SNE e análise da Separabilidade do problema

O t-SNE é uma técnica de redução de dimensionalidade não linear adequada para visualização de dados que estão em alta dimensão.

O objetivo do t-SNE é encontrar uma representação fiel de dados de um espaço com várias dimensões para um espaço com poucas dimensões, um plano (X,Y) por exemplo. Especificamente ele modela dados de forma que dados com características comuns sejam mapeados para pontos próximos e dados com características díspares para pontos distantes, com alta probabilidade.

Neste trabalho, usamos o t-SNE de visualizar a natureza do problema e verificar se há formação de clusters com os dados de treinamento. O código com o uso do t-SNE e construção do gráfico está presente no arquivo tsneTitanic.py.

As variáveis usadas para tal análise são: "Pclass", "Sex", "Age", "SibSp", "Parch" e "Survived".



Claramente existem padrões, mas a distribuição dos pontos não é perfeitamente separáveis por algum limite, há sobreviventes misturados com mortos, e mortos misturados com sobreviventes.

Nesse caso, não haverá modelo que será 100% preciso no conjunto de treinamento.

De acordo com as discussões encontradas aqui, um perceptron com múltiplas camadas e backpropagation conseguiu em média, 80% de acerto com os dados de treinamento. Então o grupo propôs iniciar com um perceptron simples e se necessário faríamos um 'upgrade'. Porém como veremos no tópico de treinamento, conseguimos em média 75% de acerto no perceptron simples com os dados de treinamento e decidimos optar seguir com ele, pois a diferença para um perceptron mais refinado era pouca.

## Treinamento da rede

A RNA foi treinada usando a regra de Hebb com modificação no critério de parada. Resumidamente o processo funciona com ajustes excitatórios e inibitórios, ou seja, após calcular o potencial de ativação e aplicar a função degrau, se a saída produzida no neurônio coincide com a saída desejada, os pesos serão incrementados; caso contrário os pesos são decrementados. Esse processo é repetido, cada repetição é denominada de período ou época. Para cada época todos os dados de treinamento são passado pela RNA. O critério de parada do treinamento é um número "x" de épocas.

O algoritmo usado para treinamento comentado:

```
def train(xk, djk, eta, features, epocas):
    wj = {}

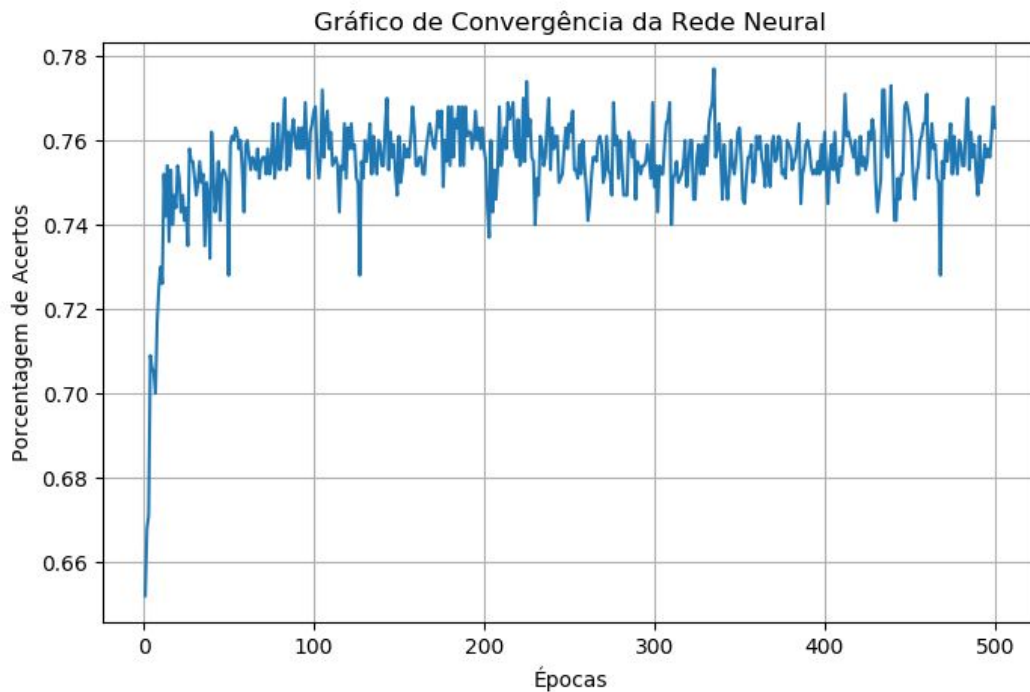
    for feature in features:
        wj[feature] = uniform(0,1) # inicializa o vetor de pesos aleatório com valores entre 0 e 1

    period = 0 # inicializa o número de épocas
    while True:
        for passenger in xk.keys(): # para cada entrada de dados de treinamento
            features = xk[passenger]
            uj = sumAttr(features, wj) # aplica a função soma nas features para obter o potencial de ativação
            yjk = degrau(uj) # aplica o potencial de ativação na função degrau
            if yjk != djk[passenger]: # se o resultado do treinamento for diferente do desejado
                for feature, value in features.items():
                    wj[feature] += eta * (djk[passenger] - yjk) * value # ajusta os pesos para cada feature
            period += 1
        if(period == epocas):
            return wj
```

## Resultados

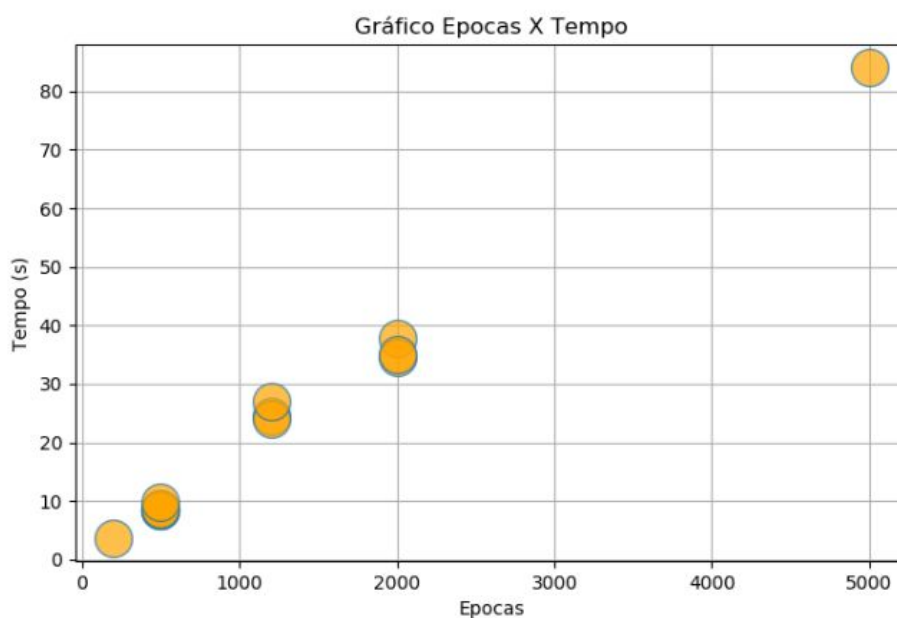
Testamos o treinamento com várias épocas. Os resultados em tempo e precisão são mostrados nos gráficos abaixo.

Gráfico de precisão, com épocas variando entre 1 e 500.



Após 50 épocas o gráfico se estabilizou entre 73 e 77% de precisão, mesmo para mais de 500 épocas de treinamento.

Gráfico de tempo, com épocas variando 1 e 1000:



Notamos claramente que o tempo é linear ao número de épocas. Como o maior processamento é feito no cálculo de peso, e a quantidade máxima que pode acontecer por época é o valor da quantidade de amostras então, o treinamento usando a regra de hebb tem complexidade linear.

## Conclusão

Resultado da execução:

```
Execucao da Rede Treinada:
Epocas : 2000
Acertos : 672 / 891
Erros : 219 / 891
Tempo de Treinamento : 34.513 segundos
Porcentagem de Acerto : 75.42%
```

```
Pesos:
w[Pclass] = -1.5022098491835865
w[Sex]     = -72.61221203516632
w[Age]     = 0.136114936322699
w[SibSp]   = -2.5182183766087247
w[Parch]   = 4.001350790195022
```

MATRIZ DE CONFUSAO  
(0 = Death, 1 = Survived)

	1 PREVISTO	0 PREVISTO
1 REAL	152 17.06%	190 21.32%
0 REAL	29 3.25%	520 58.36%

A rede obteve mais de 75% de acerto, melhor do que um simples 'chute' humano. Como dito pela análise o 'Sex' se mostrou bastante relevante para a decisão.

## Referências

**t-SNE**, disponível em <<http://dfalbel.github.io/2016/12/tsne.html>>

**t-SNE**, disponível em

<[https://en.wikipedia.org/wiki/T-distributed\\_stochastic\\_neighbor\\_embedding](https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding)>

<http://conteudo.icmc.usp.br/pessoas/andre/research/neural/>

**Prof. Dr. Rita Maria da Silva Julia, Material de curso, Redes neurais**, disponível em

<<https://www.moodle.ufu.br/mod/folder/view.php?id=155943>>