

**INSTITUTO DE ENSINO SUPERIOR THATHI**

**ALEXANDRE LUIS BINI**

**NEURALIZE.ME – IMPLEMENTAÇÃO DE REDE NEURAL ARTIFICIAL DE  
KOHONEN EM PLATAFORMA WEB**

**ARAÇATUBA  
2009**

**INSTITUTO DE ENSINO SUPERIOR THATHI**

**ALEXANDRE LUIS BINI**

**NEURALIZE.ME – IMPLEMENTAÇÃO DE REDE NEURAL ARTIFICIAL DE  
KOHONEN EM PLATAFORMA WEB**

Monografia realizada para apresentação  
de Trabalho de Conclusão do Curso de  
Ciência da Computação, orientada pelo  
professor Dr. Renato Ferrari Pacheco.

**ARAÇATUBA  
2009**

**INSTITUTO DE ENSINO SUPERIOR THATHI**

**ALEXANDRE LUIS BINI**

**NEURALIZE.ME – IMPLEMENTAÇÃO DE REDE NEURAL ARTIFICIAL DE  
KOHONEN EM PLATAFORMA WEB**

**MONOGRAFIA DE TRABALHO DE CONCLUSÃO DE CURSO**

Presidente e Orientador: Dr. Renato Ferrari Pacheco

2º Examinador: Ms. Diogo Sobral Fontes

3º Examinador: Ms. Lucilena de Lima

Araçatuba, 15 dezembro de 2009.

## **AGRADECIMENTOS**

A todos aqueles que de alguma maneira me acompanharam por todo esse processo.

Ao meu orientador, Professor Dr. Renato Ferrari Pacheco, pela idealização, apoio e prestatividade.

A meu amigo e companheiro Adriano Tadao Sabadini Matsumoto, que me ajudou a desenvolver esse projeto.

A minha namorada Nayara Cristina Jorge, pelo apoio moral, ilustrações e revisão ortográfica;

A minha mãe, Gislene de Fátima Bini, por ter cultivado as condições para que eu chegassem até aqui.

A minha avó, Dirce Alvarenga Bini, que vê o primeiro de seus netos se formar.

*“a ciência avançou alheia às críticas, e  
seu avanço se deu justamente pela  
capacidade do homem de ignorar as  
críticas e ousar.”*

*Ana Maria da Rocha Fernandes*

## RESUMO

Redes Neurais Artificiais possuem a capacidade de aprender, o que para qualquer programador algorítmico convencional é no mínimo fascinante. Este trabalho tem como objetivo a implementação de uma RNA capaz de treinar quaisquer tipos de dados utilizando a plataforma web para o desenvolvimento, o que não é comum para esse tipo de aplicação, mas que o torna acessível a qualquer um que pretenda usá-lo. O modelo de rede neural utilizado foi o SOM (self-organizing maps) do finlandês Kohonen desenvolvido na década de 1980.

**Palavras-Chave:** Inteligência Artificial; Redes Neurais Artificiais; Redes SOM (Kohonen); Ruby; Rails; Flex; MongoDB.

## ABSTRACT

Artificial Neural Networks has the capacity to learn, what is for anyone conventional algorithmic programmer, in the minimum, fascinating. This project have as objective the implementation of one ANN able to train any datasets using the web platform to development, what is not common to this type of application, but makes accessible to anyone wishing to use it. The network model used was the SOM (self-organizing maps) of the Finnish Kohonen developed in 1980s.

**Keywords:** Artificial Intelligence; Artificial Neural Network; Kohonen Map (SOM); Ruby; Rails; Flex; MongoDB

# SUMÁRIO

<b>LISTA DE FIGURAS.....</b>	<b>9</b>
<b>LISTA DE TABELAS.....</b>	<b>10</b>
<b>LISTA DE GRÁFICOS.....</b>	<b>11</b>
<b>LISTA DE QUADROS.....</b>	<b>12</b>
<b>CÓDIGO FONTE E LICENÇA.....</b>	<b>12</b>
<b>1 INTRODUÇÃO.....</b>	<b>13</b>
<b>2 REDES NEURAIS ARTIFICIAIS.....</b>	<b>15</b>
2.1 Redes SOM.....	16
<b>3 TECNOLOGIAS.....</b>	<b>23</b>
3.1 Banco de dados.....	24
3.2 Ruby.....	25
3.3 Rails.....	27
3.4 Flex.....	28
3.5 JSON.....	29
3.6 C++.....	30
<b>4 DESENVOLVIMENTO.....</b>	<b>31</b>
4.1 O cadastro dos conjuntos de dados.....	31
4.2 O treinamento da rede.....	33
4.3 Os resultados.....	41
<b>5 DEMONSTRAÇÃO.....</b>	<b>45</b>
5.1 Outros resultados.....	51
<b>6 CONCLUSÃO.....</b>	<b>56</b>
6.1 Trabalhos Futuros.....	56
<b>BIBLIOGRAFIA.....</b>	<b>58</b>
Indicações de leitura.....	59
<b>ANEXOS.....</b>	<b>61</b>
ANEXO A – Conjunto de dados de bandeiras de países: Colunas.....	62
ANEXO B – Conjunto de dados de bandeiras de países: Linhas.....	63
ANEXO C – Imagens das bandeiras.....	66

## LISTA DE FIGURAS

Figura 1: Esquema de funcionamento do neurônio artificial segundo FERNANDES.....	15
Figura 2: Representação genérica da arquitetura de uma rede neural artificial.....	16
Figura 3: Camadas de uma rede SOM.....	17
Figura 4: Exemplo da amplitude da vizinhança.....	18
Figura 5: Demonstração do treinamento de uma Rede SOM: apresentação do conjunto de dados.....	19
Figura 6: Demonstração do treinamento de uma Rede SOM: apresentação da primeira entrada à rede.....	20
Figura 7: Demonstração do treinamento de uma Rede SOM: repetindo o treinamento para a próxima entrada.....	21
Figura 8: Demonstração do treinamento de uma Rede SOM: fim do treinamento da rede.....	22
Figura 9: Esquematização da comunicação entre as linguagens utilizadas.....	23
Figura 10: Imagem da tela de cadastro dos conjuntos de dados.....	33
Figura 11: Parametrização do treinamento.....	34
Figura 12: Chapéu mexicano.....	40
Figura 13: Demonstração da exibição do resultado do treinamento.....	42
Figura 14: Comparação de elementos.....	43
Figura 15: Alteração do rótulo de exibição.....	44
Figura 16: Demonstração dos resultados: Passo 1/500.....	46
Figura 17: Demonstração dos resultados: Passo 150/500.....	47
Figura 18: Demonstração dos resultados: Passo 300/500.....	48
Figura 19: Demonstração dos resultados: Passo 450/500.....	49
Figura 20: Demonstração dos resultados: Passo 500/500.....	50
Figura 21: Outros resultados: Teste 1 - Passo 1/200.....	52
Figura 22: Outros resultados: Teste 2 - Passo 1/200.....	52
Figura 23: Outros resultados: Teste 3 - Passo 1/200.....	53
Figura 24: Outros resultados: Teste 1 - Passo 200/200.....	54
Figura 25: Outros resultados: Teste 2 - Passo 200/200.....	54
Figura 26: Outros resultados: Teste 3 - Passo 200/200.....	55

## **LISTA DE TABELAS**

Tabela 1: Evolução do índice TIOBE das 10 linguagens de programação melhor qualificadas em 2009.....	26
Tabela 2: Dados para exemplificação da escolha do intervalo de pesos.....	35
Tabela 3: Influência da taxa de aprendizado relativa à distância.....	39

## **LISTA DE GRÁFICOS**

Gráfico 1: Relação entre tempo e dimensão da matriz de pesos no treinamento da rede quando utilizado Ruby 1.9 e 1.8.....	27
Gráfico 2: Relação entre tempo e dimensão da matriz de pesos no treinamento da rede quando feito e não o uso de métodos em C.....	30
Gráfico 3: Distribuição linear com a quantidade de treinamentos igual a 20, vizinhança inicial 5 e final 1.....	36
Gráfico 4: Comparaçao entre o uso da vizinhança através de um array previamente criado e o cálculo em tempo de execução.....	37

## LISTA DE QUADROS

Quadro 1: Cópia do projeto via git.....	12
Quadro 2: Algoritmo de treinamento SOM.....	18
Quadro 3: Exemplo (em Ruby) do modelo dataset no banco MongoDB.....	24
Quadro 4: Trecho de código (em Ruby) que adiciona um atributo a um objeto dinamicamente.	
.....	25
Quadro 5: Atribuição de laço de repetição a um número.....	26
Quadro 6: Exemplo de um array de objetos no formato JSON.....	30
Quadro 7: Conversão das entradas não numéricas.....	35
Quadro 8: Criação da matriz de pesos.....	36
Quadro 9: Criação de array através da distribuição de um intervalo.....	37
Quadro 10: Treinamento da rede.....	38
Quadro 11: Método que busca o ponto mais próximo na matriz de pesos “array” em relação a entrada “x”.....	38
Quadro 12: Aplicação da taxa de aprendizado.....	40
Quadro 13: Método que encontra a posição cartesiana de cada entrada e a salva no banco de dados.....	41

## CÓDIGO FONTE E LICENÇA

Todo código fonte é Open Source e está sob a licença MIT (Massachusetts Institute of Technology License), permitindo assim que seja utilizado e licenciado em programas proprietários ou não, de forma livre, tal qual grande parte das tecnologias aqui utilizadas.

É possível visualizar e baixar o todo o código fonte e exemplos utilizados aqui, no website do projeto Neuralize.me no Github: <http://github.com/voraz/neuralize.me> ou através do controlador de versões git:

```
git clone git://github.com/voraz/neuralize.me.git
```

*Quadro 1: Cópia do projeto via git.*

## 1 INTRODUÇÃO

A inteligência pode ser definida como o conjunto de todas as faculdades intelectuais (memória, imaginação, juízo, raciocínio, abstração e conceção)<sup>1</sup> dos seres vivos. Já a Inteligência Artificial, segundo POOLE (1998) é o estudo do design de agentes inteligentes<sup>2</sup>, ou seja, um sistema que atua com inteligência. Esses agentes podem aprender e usar seu conhecimento para alcançar seus objetivos. Nesse sentido surgiram inúmeros resultados, dentre eles as redes neurais artificiais, que tem o conceito do *aprender* como um de seus principais fundamentos.

Esse projeto tem por finalidade permitir o uso na internet de um dos modelos de redes neurais artificiais, o de Kohonen, que seja capaz de agrupar quaisquer conjuntos de dados e que seja usável a qualquer pessoa, tenha ela conhecimento sobre redes neurais artificiais ou não.

A motivação para desenvolvimento de tal programa se fez em uma das aulas ministradas pelo Professor Dr. Renato Ferrari Pacheco, orientador desse projeto, onde fora apresentado um trabalho de conclusão de curso de seu ex-aluno: Márcio Renan Nunes Gomes, intitulado “Análise de desempenho acadêmico utilizando redes neurais artificiais”, trabalho este que me chamou a atenção para a possibilidade de aplicações de uma rede neural, bem como a simplicidade da implementação do algoritmo.

Esta monografia está organizada em seis Capítulos. A seguir, uma visão geral sobre cada um deles:

- Tecnologias: explicações e justificativas de cada uma das tecnologias utilizadas nesse projeto;
- Redes Neurais Artificiais: descrição do funcionamento de uma rede neural, especialmente o modelo aqui utilizado;
- Desenvolvimento: explicação de todas as etapas do desenvolvimento do programa juntamente com trechos importantes de código;
- Demonstração: apresentação da execução do programa, utilizando como exemplo as bandeiras de países;
- Conclusão: relatamento dos resultados, os objetivos alcançados e os trabalhos futuros

<sup>1</sup> PRIBERAM. *Dicionário Proberam da Língua Portuguesa*. Disponível em: <<http://www.priberam.pt/DLPO/default.aspx?pal=intelig%C3%A3ncia>>. Acesso em: 6 dez. 2009.

<sup>2</sup> "Computational intelligence is the study of the design of intelligent agents" - Tradução nossa.

- a serem desenvolvidos;
- Bibliografia: relação de todas as obras utilizadas nesse projeto, bem como indicações de leitura sobre os temas aqui abordados.

## 2 REDES NEURAIS ARTIFICIAIS

Uma RNA (rede neural artificial) é um sistema computacional paralelo e distribuído, desenvolvido através de modelos matemáticos inspirados no sistema nervoso dos seres vivos. Acredita-se que a neurocomputação possa se beneficiar de modelos que tenham tal apelo.

O princípio de uma RNA, tal qual o da organização de neurônios do cérebro, baseia-se em aprender e tomar decisões através da aprendizagem (experiência). Tal capacidade é um dos grandes atrativos ao seu uso que vem sendo (potencialmente) feito para resolução de problemas de reconhecimento de padrões e categorização de dados distintos.

O neurônio é a célula fundamental do cérebro humano: estima-se que existam cerca de  $10^{11}$  delas. A representação artificial do seu modelo, desenvolvido por Warren McCulloch e Walter Pitts em 1943, é o elemento base das redes neurais artificiais sendo que seu funcionamento, segundo FERNANDES (2003), acontece na seguinte sequência:

1. Sinais são apresentados à entrada;
2. cada sinal é multiplicado por um número, ou peso, que indica a sua influência na saída da unidade;
3. é feita a soma ponderada dos sinais que produz um nível de atividade;
4. se este nível de atividade exceder um certo limite (*threshold*) a unidade produz uma determinada resposta de saída.

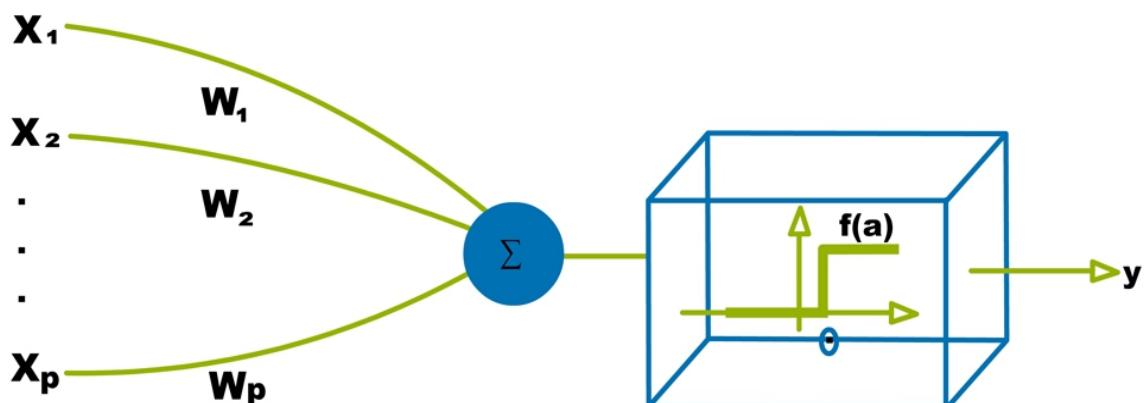
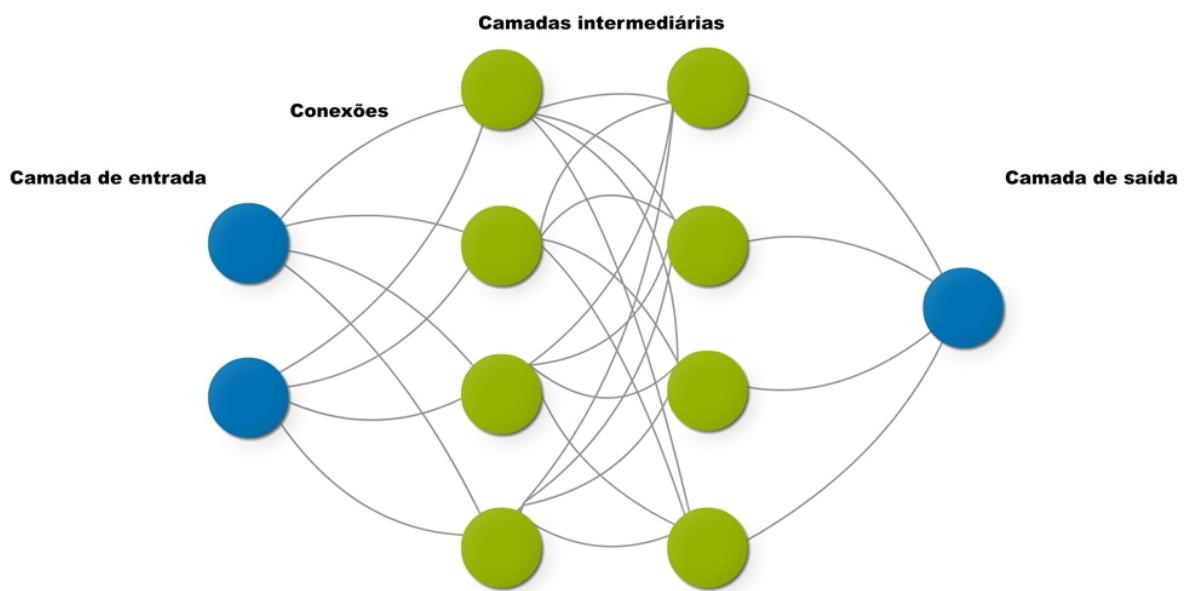


Figura 1: Esquema de funcionamento do neurônio artificial segundo FERNANDES.

Existem diversos modelos de RNA's, cada qual com suas peculiaridades, seja no aprendizado (supervisionado, não supervisionado, reforço); na característica predominante; nas camadas implementadas ou mesmo no método de treinamento. Porém elas são tipicamente arquitetadas em três camadas que podem estar ou não, conectadas as unidades da camada posterior: a de entrada, onde os padrões são apresentados à rede; as escondidas (ou intermediárias), que podem ser consideradas extratoras de características; e a de saída, onde o resultado é apresentado.



*Figura 2: Representação genérica da arquitetura de uma rede neural artificial.*

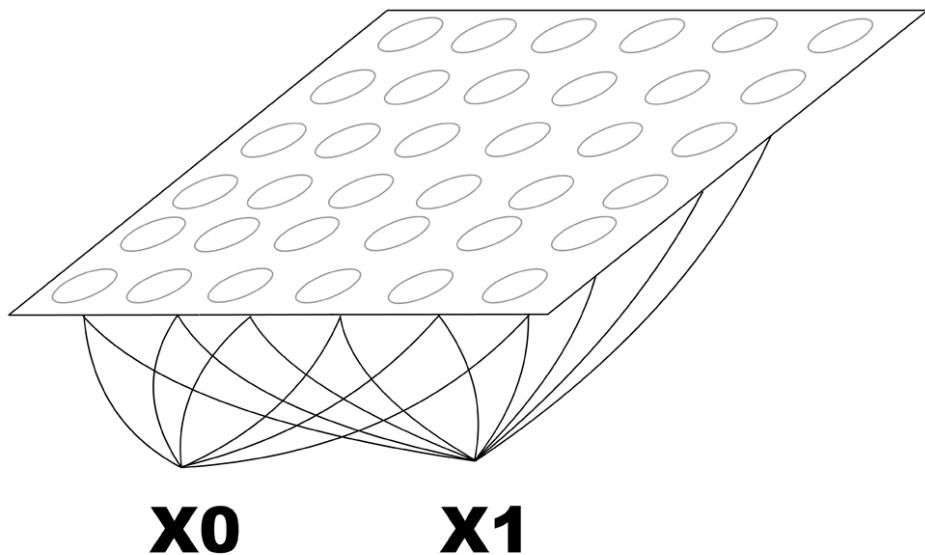
## 2.1 Redes SOM

As redes SOM (*Self-organizing maps*), que foram desenvolvidas na década de 1980 pelo finlandês Teuvo Kohonen, utilizam algoritmos baseados em conceitos biologicamente plausíveis, inspirados no mapeamento realizado pelo cérebro. Tais redes são competitivas e não supervisionadas, logo, não existe um supervisor para controlar o seu aprendizado que se faz sozinho através da exploração das características dos dados apresentados, permitindo “a representação de dados N-dimensionais em um espaço M-dimensional, onde  $M \ll N$ ” (BRAGA, 2000, p. 127), isto é, são capazes de representar um conjunto de dados em um plano cartesiano.

Sua principal área de atuação é no reconhecimento de padrões e

agrupamento, sendo que uma das primeiras aplicações a utilizá-la foi o datilógrafo fonético, desenvolvido pelo próprio Kohonen, que converte “linguagem falada em texto escrito” (BRAGA, 2000, p. 126).

O grande advento das redes SOM está na criação de um modelo composto por (pelo menos) dois subsistemas de naturezas diferentes interconectados: a camada de entradas e a de aprendizado.



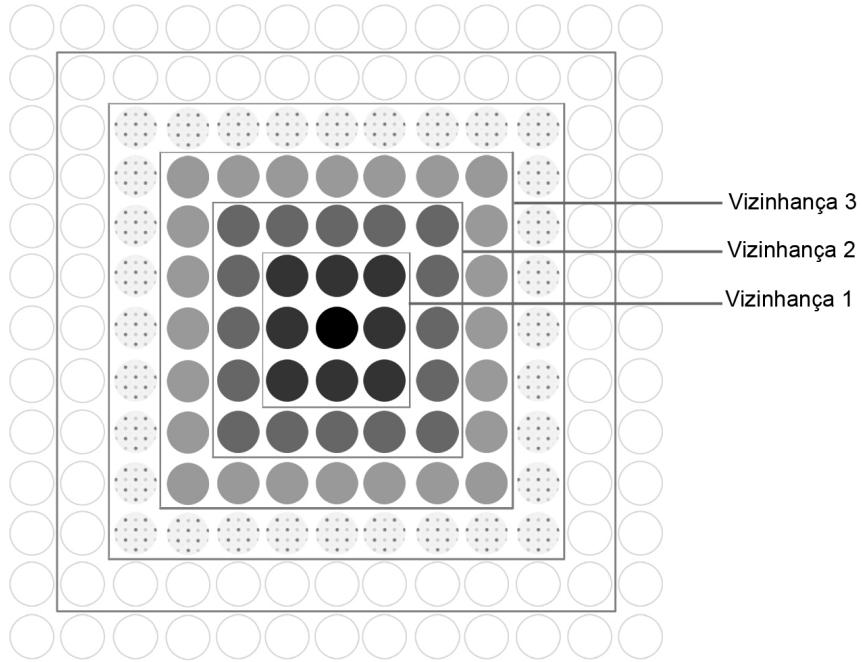
*Figura 3: Camadas de uma rede SOM.*

A camada de entradas representa os neurônios a serem treinados pela rede e a camada de aprendizado representa, analogicamente, os do córtex cerebral. O seu funcionamento é bastante simples: apresenta-se um neurônio da camada entrada para a de aprendizado, que faz uma varredura afim de encontrar qual dos seus é o mais parecido com a entrada apresentada, isto é, qual deles responde mais fortemente ao estímulo. Tal como é feito pelo córtex.

Observa-se que neurônios estão espacialmente ordenados dentro dessas áreas, e, assim neurônios topologicamente próximos tendem a responder a padrões ou estímulos semelhantes. (BRAGA, 2000, p. 113)

Essa competição entre os neurônios é chamada *winner-takes-all* (o vencedor leva tudo, tradução nossa) pois ele [*o vencedor*] tende a convergir mais fortemente para a entrada apresentada durante o treinamento do que seus vizinhos, que também convergem,

porém com uma intensidade menor, o chamado “chapéu mexicano”, onde cada neurônio influencia o estado de ativação de seus neurônios vizinhos.



*Figura 4: Exemplo da amplitude da vizinhança.*

Dessa maneira a camada de aprendizado é um mapa topológico que reflete a posição do neurônio de entrada no plano cartesiano.

Um algoritmo básico desse processo seria:

```

Iniciar pesos e parâmetros
Repetir
    Para cada padrão de treinamento x faça
        Definir o nodo vencedor
        Atualizar os pesos deste nodo e de seus vizinhos
        Se o número do ciclo for múltiplo de N
            reduzir a taxa de aprendizado e a área de vizinhança
    Até o mapa de características não mudar

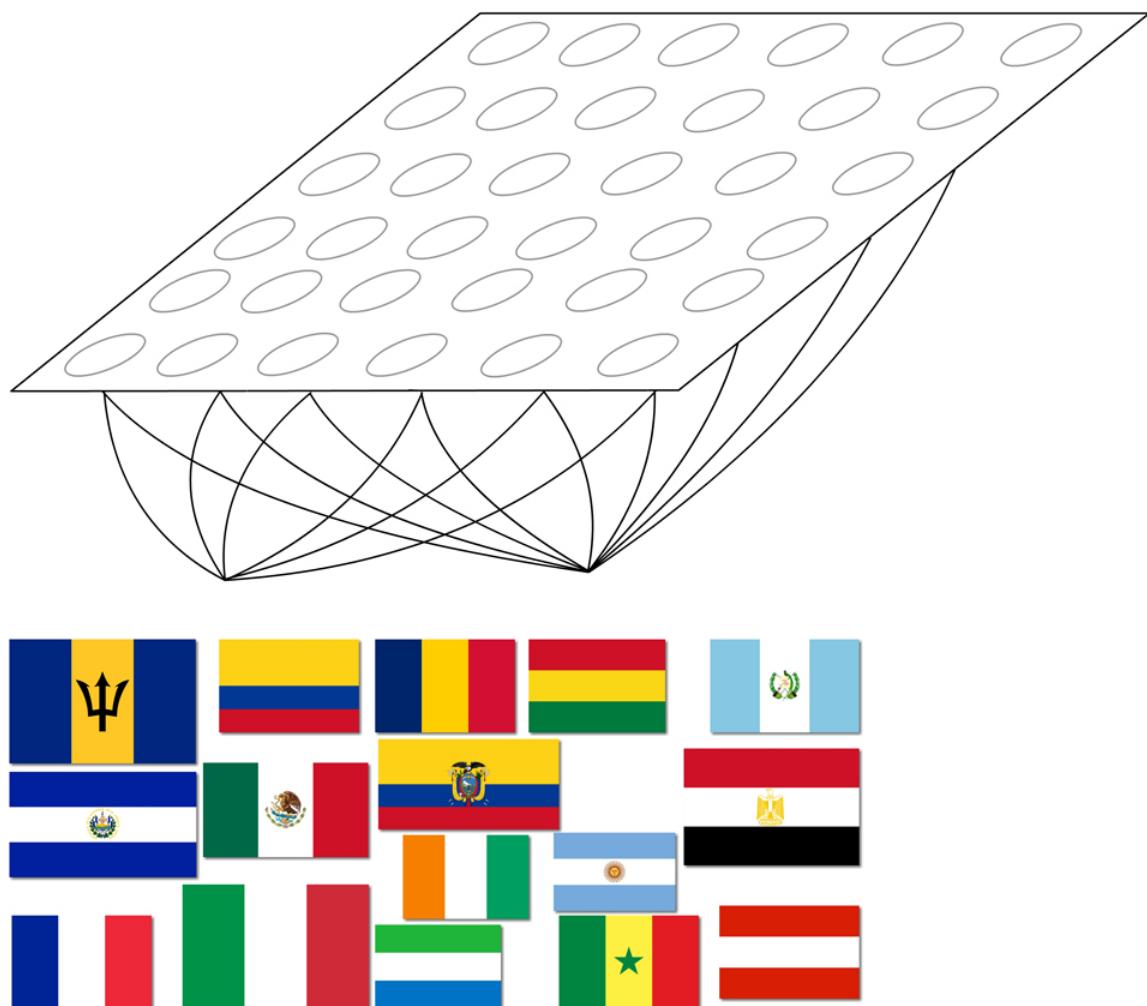
```

*Quadro 2: Algoritmo de treinamento SOM.*

O treinamento da rede SOM pode ser dividido em duas fases sequenciais: a de ordenação e a de convergência. A fase de ordenação tem por objetivo descobrir quantos *clusters* (agrupamentos) a rede provavelmente terá. Esse mapeamento é de certa forma grosso, pois a taxa de aprendizagem e de vizinhança são geralmente altas, o que acarreta

em uma grande mudança na camada de aprendizado. A fase de convergência objetiva sofisticar o agrupamento descoberto, sendo assim, as taxas de aprendizado e vizinhança são geralmente baixas, o que resulta em um ajuste fino, com poucas mudanças.

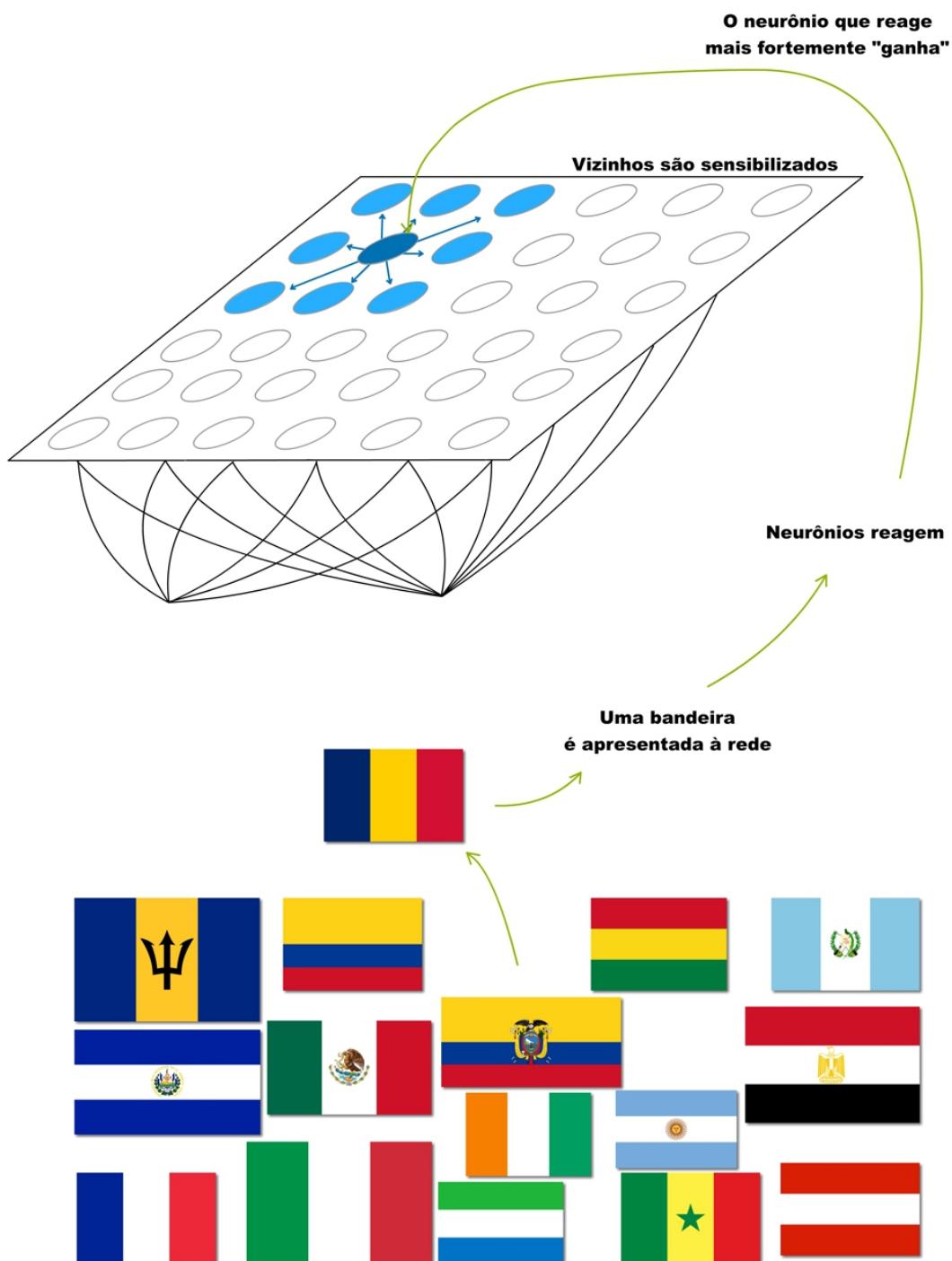
Para exemplificar o treinamento da rede, foi criada uma demonstração, baseada na de Francisco Aranha<sup>3</sup>, onde, após inicializar a rede com o valor dos seus pesos aleatórios, apresenta-se o conjunto de dados a ser treinado, no exemplo, bandeiras de países. São levadas em consideração aqui a direção das listras (horizontais e verticais) e se possuem símbolo no centro ou não.



*Figura 5: Demonstração do treinamento de uma Rede SOM: apresentação do conjunto de dados.*

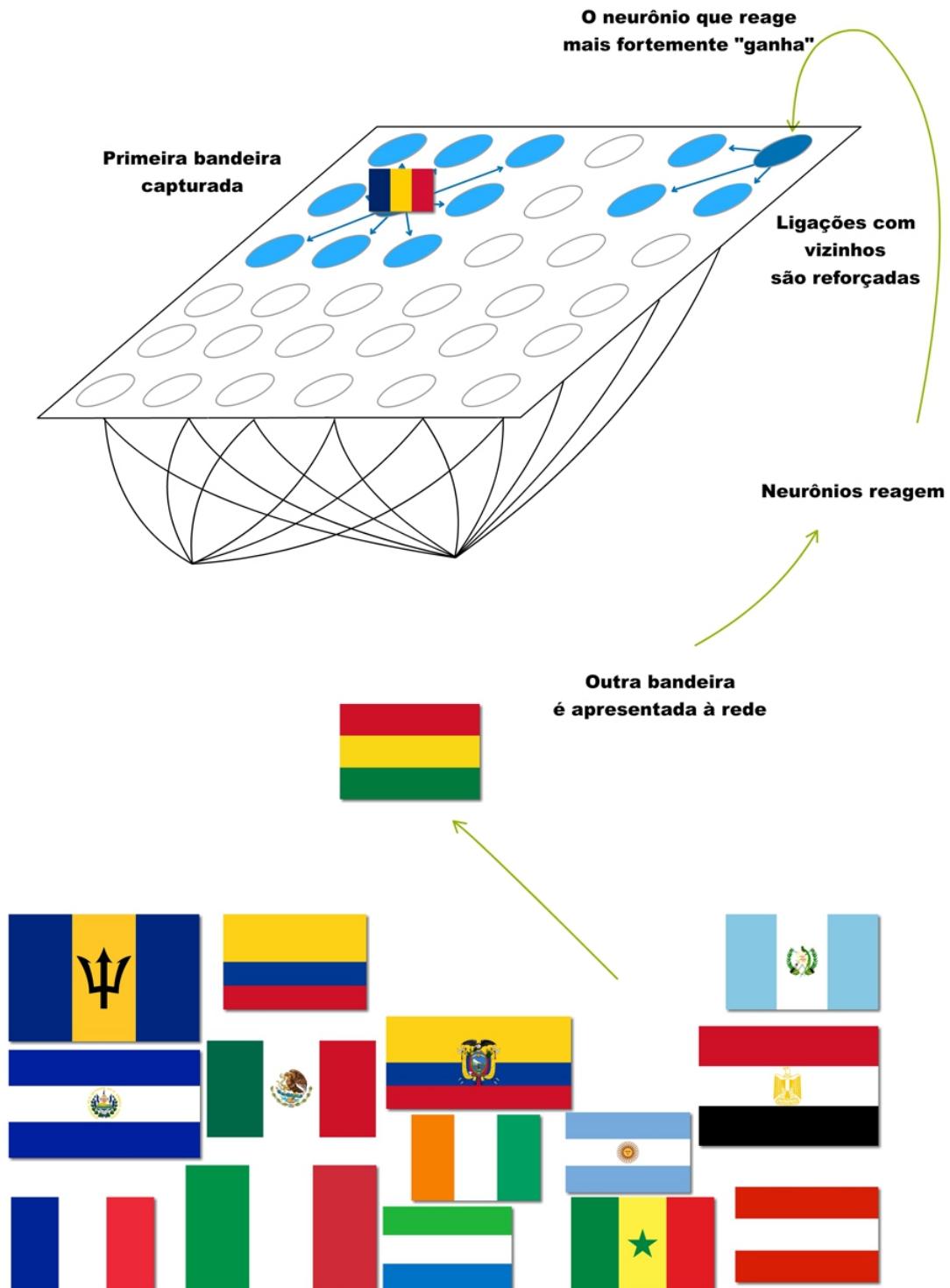
<sup>3</sup> UFMS. Segmentação com mapas neurais de Kohonen. Disponível em: <<http://www.dct.ufms.br/~mzanusso/Kohonen.htm>>. Acesso em: 03 dez. 2009.

A seguir, é apresentada a primeira das bandeiras à camada de aprendizado. O neurônio que reagir com a maior intensidade é tido como vencedor e seu peso e de seus vizinhos são ajustados afim de convergir para o valor da entrada apresentada.



*Figura 6: Demonstração do treinamento de uma Rede SOM: apresentação da primeira entrada à rede.*

O processo anterior é repetido para todas as entradas do conjunto de dados apresentado.



*Figura 7: Demonstração do treinamento de uma Rede SOM: repetindo o treinamento para a próxima entrada.*

Ao final do processo, a rede está organizada em 4 grupos: os que possuem listras horizontais e não símbolo; os que possuem listras horizontais e símbolo; os que possuem listras verticais e não símbolo; e os que possuem listras verticais e símbolo.

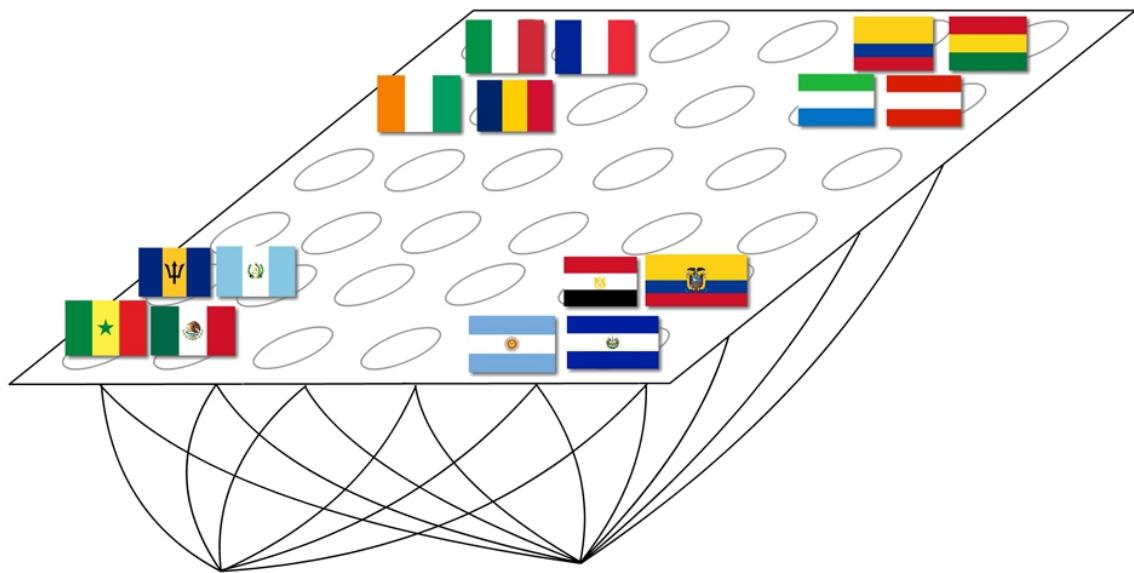
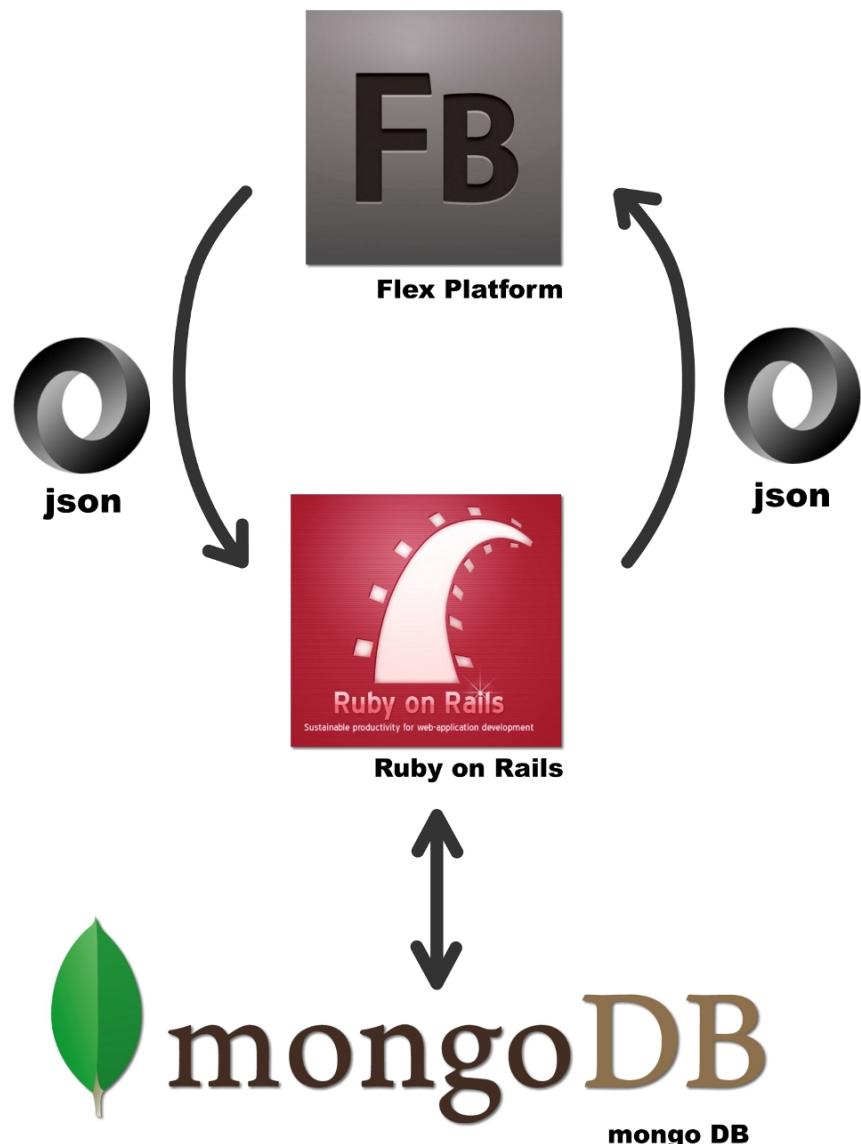


Figura 8: Demonstração do treinamento de uma Rede SOM: fim do treinamento da rede.

### 3 TECNOLOGIAS

A implementação da rede neural proposta neste projeto é feita na plataforma web, tendo como base as linguagens de programação ActionScript e Ruby com seus respectivos framework's<sup>4</sup>: Flex e Rails, integrando-os através de requisições REST que transitam objetos JSON e MongoDB para banco de dados. A sequência das requisições entre tais ferramentas se faz da seguinte forma:



*Figura 9: Esquematização da comunicação entre as linguagens utilizadas.*

<sup>4</sup> Framework é um conjunto de soluções genéricas altamente confiáveis desenvolvidas a partir de padrões de projetos com o intuito de agilizar o desenvolvimento de software.

### 3.1 Banco de dados

Com o objetivo de implementar uma rede neural capaz de treinar quaisquer dados, houve a imprevisibilidade em relação aos tipos de dados que poderiam ser armazenados, bem como sobre a necessidade ou não de escalabilidade. Portanto, um banco de dados relacional poderia não ser uma boa escolha, pois neles geralmente há a necessidade de definir um tipo para o atributo, além da complexidade aliada ao alto custo de serem escalados.

Motivado por esses e outros aspectos [*dos bancos de dados relacionais*], surgiu em 2009 (apesar da criação desse tipo de banco de dados ser anterior) um movimento denominado "NoSQL" - *not only sql*<sup>5</sup> - que basicamente sugere o uso de banco de dados não relacionais. Pode-se considerar que boa parte do sucesso do movimento está na ligação entre a estrutura dos bancos de dados não relacionais e a realidade do desenvolvimento de softwares nos dias de hoje. Nesse sentido existem diversas implementações sendo feitas por grandes empresas, como Google Bigtable (Google), Hadoop (Yahoo!), Dynamo (Amazon), Cassandra (Digg, Facebook), Voldemort (LinkedIn), Tokyo Cabinet (Mixi), MongoDB (Engine Yard), etc. Todos eles tem como características: a alta performance, o armazenamento de dados no formato *key-value* (chave-valor), sendo que o valor pode ser de qualquer tipo de dado (o que soluciona o primeiro aspecto do projeto); a facilidade e o baixo custo de escalabilidade, visto que se escala horizontalmente ao invés de verticalmente - como nos banco de dados relacionais.

Por sua simplicidade de implementação, instalação, performance e *plug-ins* disponíveis para Ruby, o banco de dados não relacional orientado a documentos escolhido foi o MongoDB. O trecho de código (em Ruby) a seguir demonstra a simplicidade na representação dos conjuntos de dados, chamados aqui de "Datasets":

```
class Dataset
  include MongoMapper::Document

  key :lines, Array
  key :columns, Array
  key :title
  key :description

end
```

Quadro 3: Exemplo (em Ruby) do modelo dataset no banco MongoDB.

---

5 “*Não somente sql*” – tradução nossa.

Note que foi definido o tipo *array* para os atributos *lines* e *columns*, mas não para *title* e *description*. O motivo de tal definição é única e exclusivamente por uma questão de visualização, não sendo necessária sequer a própria definição dos atributos. Partindo do modelo acima poderíamos criar um atributo “*test*” em tempo de execução para um determinado Dataset:

```
>> data = Dataset.new
=> #<Dataset _id: nil, lines: [], columns: [], title: nil, description: nil>

>> data[:test] = 100
=> 100

>> data.save
=> true

>> data.test
=> 100
```

*Quadro 4: Trecho de código (em Ruby) que adiciona um atributo a um objeto dinamicamente.*

### 3.2 Ruby

A linguagem de programação Ruby foi desenvolvida por Yukihiro Matsumoto, inspirada no que de melhor havia nas linguagens: Perl, SmallTalk, Eiffel, Ada e Lisp – “*Eu queria uma linguagem interpretada que fosse mais poderosa que Perl e mais orientada a objetos que Python*”<sup>6</sup>(MATSUMOTO, 2001). Seu foco é a simplicidade e produtividade, envoltos por uma sintaxe elegante e intuitiva de fácil compreensão:

Eu acredito que as pessoas querem se expressar quando programam. Elas não querem lutar com a linguagem. Linguagens de programação devem fluir naturalmente para os programadores<sup>7</sup>. (MATSUMOTO, 2001)

Ruby é uma linguagem totalmente orientada a objetos, isso significa que mesmo classes e tipos de dados que algumas linguagens tratariam como primitivos, são objetos em Ruby. Isso facilita sua utilização visto que regras aplicadas em objetos aplicam-se

<sup>6</sup> "I wanted a scripting language that was more powerful than Perl, and more object-oriented than Python." - Tradução nossa. MATSUMOTO, Yukihiro. *An Interview with the Creator of Ruby*. 29 nov. 2001. Disponível em <<http://linuxdevcenter.com/pub/a/linux/2001/11/29/ruby.html>> Acesso em: 20 nov. 2009

<sup>7</sup> "I believe people want to express themselves when they program. They don't want to fight with the language. Programming languages must feel natural to programmers." - Tradução nossa. MATSUMOTO, Yukihiro. *An Interview with the Creator of Ruby*. 29 nov. 2001. Disponível em <<http://linuxdevcenter.com/pub/a/linux/2001/11/29/ruby.html>> Acesso em: 20 nov. 2009

a tudo. Um simples exemplo dessa diferença é a atribuição de um método a um número:

```
100.times { puts "Hello World" }
```

*Quadro 5: Atribuição de laço de repetição a um número.*

Tal linguagem destaca-se ainda pela grande ascensão nos últimos anos, sendo a que mais cresceu entre 2005 e 2009, segundo o índice TIOBE<sup>8</sup>, vinda da 24<sup>a</sup> em 2005 para a 10<sup>a</sup> posição em 2009, o que de certa maneira a torna uma das linguagens mais promissoras para os próximos anos:

Linguagem	Posição Novembro/2009	Posição Novembro/2005	Evolução (posições)
Java	1	1	0
C	2	2	0
PHP	3	4	1
C++	4	3	-1
(Visual) Basic	5	5	0
C#	6	7	1
Python	7	8	1
Perl	8	6	-2
JavaScript	9	9	0
<b>Ruby</b>	<b>10</b>	<b>24</b>	<b>14</b>

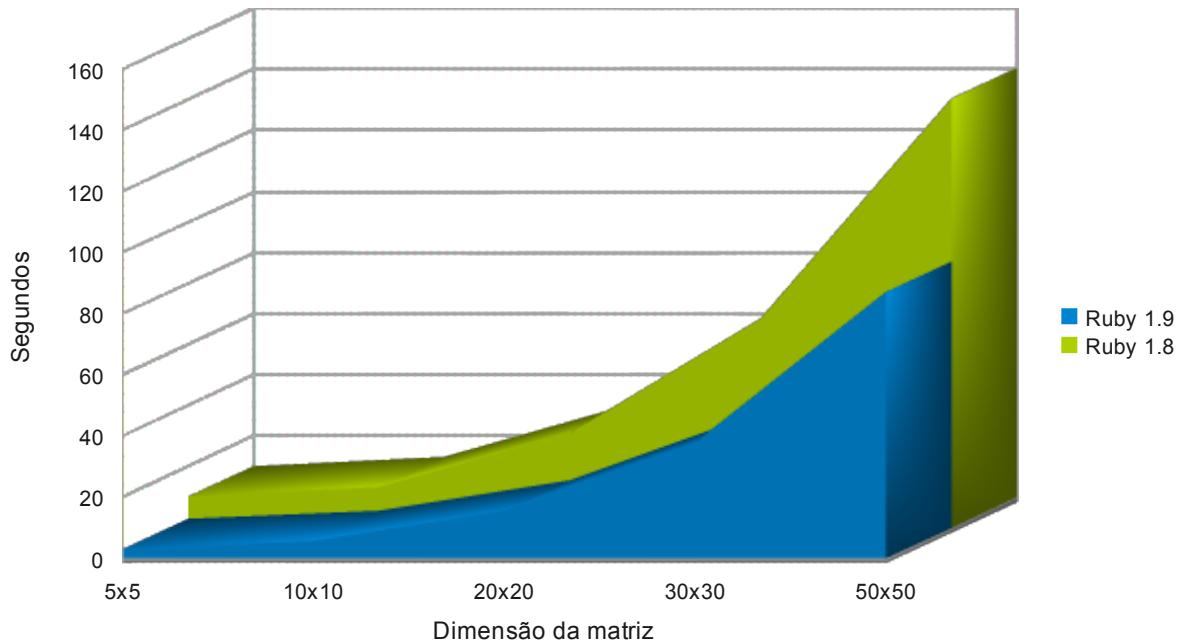
*Tabela 1: Evolução do índice TIOBE das 10 linguagens de programação melhor qualificadas em 2009.*

A versão do Ruby utilizada inicialmente foi a 1.8 devido a sua grande difusão, mas ao desenrolar da implementação do algoritmo proposto, percebeu-se que a aplicação demandaria de uma grande capacidade de processamento, o que acarretou na substituição pela versão 1.9 visto que sua performance é superior<sup>9</sup>, como mostra a

8 O Índice TIOBE é calculado através da popularidade de buscas do termo “<linguagem> programming” em diversos sites: Google, Google Blogs, MSN, Yahoo!, Wikipédia e Youtube. TIOBE SOFTWARE. *TIOBE Programming Community Index Definition.* 2009. Disponível em: <[http://www.tiobe.com/index.php/content/paperinfo/tpci/tpci\\_definition.htm](http://www.tiobe.com/index.php/content/paperinfo/tpci/tpci_definition.htm)>. Acesso em: 20 nov. 2009. O índice TIOBE envolve, por sua (possível) superficialidade metodológica, uma série de discussões entre os desenvolvedores. Uma argumentação desse tipo pode ser vista em <<http://www.akitaonrails.com/2008/4/13/off-topic-nunca-confie-no-Tiobe>>.

9 Essa comparação pode ser vista mais profundamente em <<http://blog.pluron.com/2009/05/ruby-19-performance.html>> e <[http://antoniocangiano.com/2009/08/03/performance-of-ironruby-ruby-on-windows](http://antoniocangiano.com/2009/08/03/performance-of-ironruby-ruby-on-windows/)>

comparação a seguir:



*Gráfico 1: Relação entre tempo e dimensão da matriz de pesos no treinamento da rede quando utilizado Ruby 1.9 e 1.8.*

### 3.3 Rails

Ruby possui uma série de frameworks web, dentre eles: Merb, Sinatra e Rails. Este, utilizado no projeto, fora extraído por David Heinemeier Hansson em 2003, de um software de gerenciamento de projetos chamado Basecamp, da empresa 37Signals. A popularidade do Ruby se deve em parte ao Rails, tanto que pouco após sua disponibilização ao público, o Ruby recebeu o prêmio de linguagem do ano de 2006 pela TIOBE<sup>10</sup>. O Rails também foi bastante premiado mundo afora, e no Brasil recebeu o prêmio Info 2008 na categoria desenvolvimento<sup>11</sup>.

A arquitetura organizacional do Rails é baseada no muito conhecido MVC (*model-view-controller*) que tem como característica a separação da camada de negócios da camada visual. O Rails fundamenta-se em diversas filosofias, tais como: Convention Over Configuration (COC), que implementa diversas convenções, diminuindo ou eliminando a

<sup>10</sup> TIOBE SOFTWARE. *TIOBE Programming Community Index Definition*. 2009. Disponível em: <[http://www.tiobe.com/index.php/content/paperinfo/tpci/tpci\\_definition.htm](http://www.tiobe.com/index.php/content/paperinfo/tpci/tpci_definition.htm)>. Acesso em: 20 nov. 2009.

<sup>11</sup> INFO ONLINE. Abril ed. 2008. Disponível em <<http://info.abril.com.br/premioinfo/2008/software.shl>> Acesso em: 20 nov. 2009.

necessidade de configurações específicas; Don't Repeat Yourself (DRY), que visa a não duplicação do código, fato que facilita as alterações e diminui a possibilidade de inconsistências; *Representational State Transfer* (REST), que organiza a aplicação em torno de recursos e padrões HTTP, o que o tornou um dos mais utilizados padrões de projeto para aplicações web; Plug-ins e extensões que existem em vasta quantidade e são desenvolvidos por seus usuários com a finalidade de implementar funcionalidades à suas aplicações.

Com todas essas considerações e, principalmente pela intenção de utilizar a plataforma web, foi escolhido tal framework pois ele [*o Rails*] ainda proporcionou um ambiente agradável, produtivo e extensível.

### 3.4 Flex

A linguagem de programação ActionScript, orientada a objetos e amplamente difundida em aplicações web e sites, é utilizada por diversas IDE's (ambiente integrado de desenvolvimento) e framework's tais como o Flash, Air e o Flex que possibilitam o desenvolvimento de *Rich Internet Applications* (RIA's)<sup>12</sup> - aplicações web com características de softwares tradicionais desenvolvidos para desktop - baseadas na *Flash Platform*. É possível a implementação de diversos *Design Pattern's* (padrões de projeto de software): MVC, Facade, Proxy, Mediator, etc; sendo estes aplicados nesse projeto através do framework PureMVC<sup>13</sup>, afim de padronizar o desenvolvimento facilitando possíveis futuras incrementações.

As aplicações desenvolvidas no framework Flex são compiladas em um arquivo de extensão swf (*small web format*) e executadas em máquina virtual chamada Flash Player, cujas principais características são: disseminação<sup>14</sup> - aproximadamente 99% dos computadores com acesso à internet o possuem; multi-plataforma<sup>15</sup> - compatibilidade com diversos sistemas operacionais (Linux, Mac, Solaris e Windows); além da alta performance das aplicações em tempo de execução<sup>16</sup>.

---

12 Termo sugerido pela Macromedia Inc (posteriormente adquirida pela Adobe Inc) em Março de 2002 em <<http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf>>. Acesso em 28 nov. 2009.

13 PUREMVC FRAMEWORK. *Code at the speed of thought.* Disponível em: <[http://trac.puremvc.org/PureMVC\\_AS3](http://trac.puremvc.org/PureMVC_AS3)>. Acesso em 28 nov. 2009.

14 ADOBE. *Flash Player Statistics.* Disponível em: <[http://www.adobe.com/products/player\\_census/flashplayer](http://www.adobe.com/products/player_census/flashplayer)>. Acesso em 28 nov. 2009.

15 ADOBE. *Flash Player 10: System requirements.* Disponível em: <<http://www.adobe.com/products/flashplayer/systemreqs/#os>>. Acesso em 28 nov. 2009.

16 ADOBE FLASH PLAYER 10. *Deliver breakthrough, cross-platform web experiences.* Formato PDF.

A escolha do framework Flex para o desenvolvimento do *front-end* (camada visual) desse projeto, é feita pela valorização das características supra citadas e pela possibilidade de demonstrar a execução da rede neural proposta através de transições dos objetos em tempo de execução, fato este que consolidou a *Flash Platform* no mercado atual, tanto que o Flex recebeu vários prêmios<sup>17</sup> incluindo o de “Tecnologia do ano de 2009”<sup>18</sup> pela InfoWorld.

### 3.5 JSON

No projeto foram implementados separadamente o *Front-end* (Flex) e *Back-end* (camada de processamento, em Rails) por considerar que para este caso o Flex ofereça mais recursos visuais que o HTML. Tais camadas podem ser integradas de 3 maneiras<sup>19</sup>: *Remote Object Services* (amf<sup>20</sup>), *Web Services* (xml) e *HTTP Services* (object, array, xml, e4x, flashvars, text)<sup>21</sup>. Destes, apenas o amf não é nativo para o Ruby, mas pode ser implementado através do plug-in RubyAMF<sup>22</sup>.

Fora usado inicialmente o *Remote Object*, mas devido a uma incompatibilidade com a versão 1.9 do Ruby, precisou ser trocado. Os *Web Services* suportam apenas o uso de xml, o que de certa maneira limitaria as nossas possibilidades. Foi escolhido então integrar as camadas através de *HTTP Services*, utilizando objetos JSON (*JavaScript Object Notation*), um formato de intercâmbio de dados baseado em texto legível para humanos, como demonstrado a seguir:

Disponível em: <[http://www.adobe.com/products/flashplayer/pdfs/Flash\\_Player\\_10\\_Datasheet.pdf](http://www.adobe.com/products/flashplayer/pdfs/Flash_Player_10_Datasheet.pdf)>. Acesso em 28 nov. 2009.

17 ADOBE. *Flex 3: Awards*. Disponível em <<http://www.adobe.com/products/flex/buzz/awards/>>. Acesso em 28 nov. 2009.

18 InfoWorld: *Adobe Flash, Flex and Air*. Disponível em: <<http://www.infoworld.com/node/62963>>. Acesso em 28 nov. 2009.

19 ADOBE FLEX 4. *Acessing data services overview*. Disponível em: <[http://help.adobe.com/en\\_US/Flex/4.0/AccessingData/WSbde04e3d3e6474c4-2c1457f3121e70c6c33-8000](http://help.adobe.com/en_US/Flex/4.0/AccessingData/WSbde04e3d3e6474c46c45e7b4120d413dc14-8000.html#WSbde04e3d3e6474c4-2c1457f3121e70c6c33-8000)>. Acesso em 28 nov. 2009.

20 ADOBE SYSTEMS INCORPORATED. *AMF 3 Specification*. Formato PDF. Disponível em <[http://opensource.adobe.com/wiki/download/attachments/1114283/amf3\\_spec\\_05\\_05\\_08.pdf](http://opensource.adobe.com/wiki/download/attachments/1114283/amf3_spec_05_05_08.pdf)>. Acesso em 28 nov. 2009.

21 HTTPSERVICE. *Adobe Flex 4 Beta Language Reference*. Disponível em: <[http://help.adobe.com/en\\_US/Flex/4.0/langref/mx/rpc/http/HTTPService.html#resultFormat](http://help.adobe.com/en_US/Flex/4.0/langref/mx/rpc/http/HTTPService.html#resultFormat)>. Acesso em 28 nov. 2009.

22 RUBYAMF. *Flash and Flex Remoting for Ruby on Rails*. Disponível em <<http://code.google.com/p/rubyamf>>. Acesso em 28 nov. 2009.

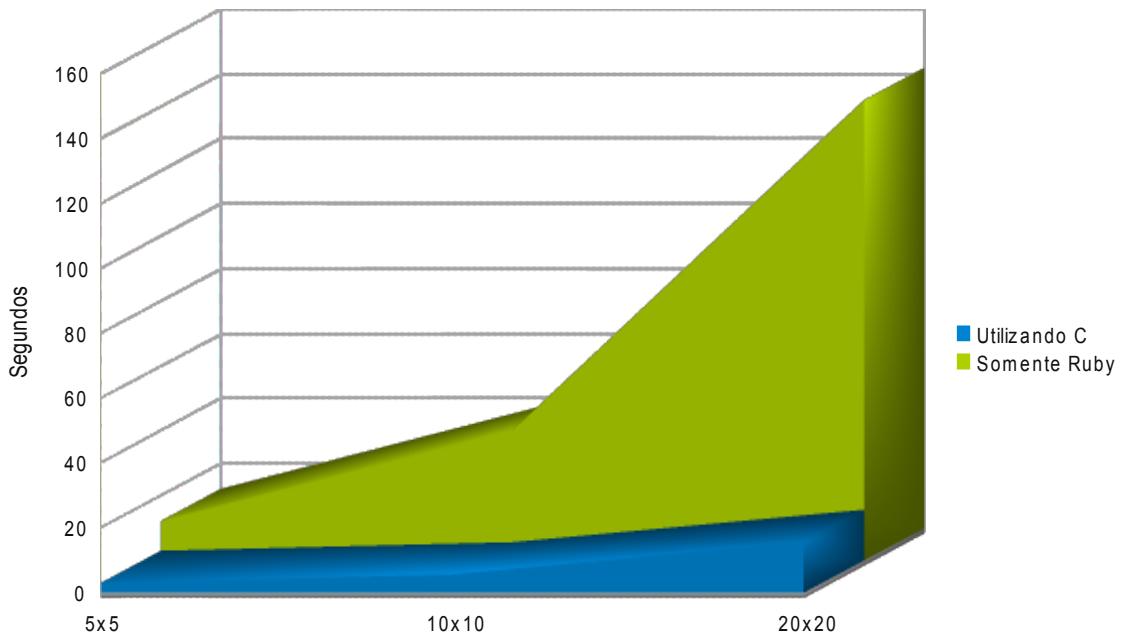
```
[  
  {  
    "id": "4afdb0f49c59153f1f000001",  
    "title": "Flags"  
  },  
  {  
    "id": "4afdec7c9c59150a04000001",  
    "title": "Clouds"  
  }  
]
```

*Quadro 6: Exemplo de um array de objetos no formato JSON.*

### 3.6 C++

Notou-se que apesar de utilizada a versão 1.9 do Ruby a aplicação ainda não possuía um desempenho adequado. Vale ressaltar que Ruby é uma linguagem de programação interpretada e não compilada, o que por via de senso, entende-se ser mais lenta do que, por exemplo, Java e C, que são exemplos de linguagens compiladas.

Tal problema foi solucionado através do uso da extensão RubyInline que permite a escrita de métodos C/C++ diretamente em arquivos Ruby, proporcionando assim um ganho significativo no desempenho, como pode ser visto no gráfico abaixo que compara a execução de treinamentos quando se utiliza e não alguns métodos em C.



*Gráfico 2: Relação entre tempo e dimensão da matriz de pesos no treinamento da rede quando feito e não o uso de métodos em C.*

## 4 DESENVOLVIMENTO

A implementação da rede de Kohonen foi feita de modo a viabilizar o treinamento de qualquer conjunto de dados, motivo pelo qual foi complexo chegar a uma arquitetura satisfatória. Outro aspecto considerado é a capacidade demonstrativa da troca de posições dos elementos durante o treinamento da rede, para que mesmo alguém que esteja iniciando os estudos ou não seja da área pudesse compreender, ainda que superficialmente, o processo da aprendizagem. Também o fato do aplicativo ser direcionado a essas pessoas foi ponto fundamental nas escolhas tanto da plataforma (web) quanto das tecnologias (citadas no Capítulo 3 - Tecnologias), permitindo ao usuário uma boa experiência de usabilidade, “[...] *atributo mais importante [...]. No final, o valor da sua aplicação é somente o valor do que é entregue ao usuário final.*” (GALPERIN, Eran. *What makes a good programmer?*. Tecfounder, Israel, jul. 2009. Disponível em: <<http://www.techfounder.net/2009/07/22/what-makes-a-good-programmer>>. Acesso em 10 dez. 2009.)

Para a demonstração da aplicação, foi utilizado um conjunto de dados provenientes da Universidade da Califórnia<sup>23</sup>, sobre as bandeiras de países<sup>24</sup>. Qualquer um poderia ter sido escolhido, mas a escolha deste se fez por serem de simples comparação visual, pela disponibilidade de todas as imagens e por já termos grande parte delas em nossa memória.

A aplicação possui duas seções principais: a de cadastro dos conjuntos de dados e a de treinamento da rede.

### 4.1 O cadastro dos conjuntos de dados

Foi criada uma seção onde o usuário pudesse cadastrar em formato de tabela (linhas e colunas) os dados a serem treinados pelo sistema. A fim de automatizar a inserção de dados, o aplicativo possui a capacidade de importar tanto as linhas quanto as colunas de dados vindos de quaisquer arquivos de texto. Essas tabelas de dados são facilmente manipuladas através de controles avançados de ordenação bem como de inserção e deleção de linhas e

---

<sup>23</sup> UCI MACHINE LEARNING REPOSITORY. *Center for Machine Learning and Intelligent Systems*. Disponível em: <<http://archive.ics.uci.edu/ml>>. Acesso em: 2 nov. 2009.

<sup>24</sup> Pelos dados serem coletados em 1986, alguns países que constam nos arquivos já não existem mais, como por exemplo a USRR.

colunas.

As colunas representam atributos dos elementos, que na demonstração são: quantidade de estrelas, de cores, etc., podendo ainda ser inserida uma descrição, que é de grande valia para o entendimento dos elementos. É possível ainda selecionar quais das colunas farão parte do treinamento, considerando que algumas podem não ser relevantes, como a coluna “nome” da demonstração, que serviria apenas para identificar as bandeiras.

Já as linhas, representam os elementos propriamente ditos, que são as bandeiras: Brasil, Argentina, etc. Nelas pode-se cadastrar uma imagem, independentemente do seu tamanho (pois ela é automaticamente redimensionada pelo Ruby para adaptar-se aos padrões do aplicativo), e o valor de seus atributos, que podem ser de quaisquer tipos, pois todos são transformados em números no treinamento da rede<sup>25</sup> garantindo assim seu pleno funcionamento.

Vale ressaltar que um dos grandes fatores da qualidade do treinamento da rede está relacionado a coesão das entradas apresentadas. *“Para que seja desenvolvida uma função de processamento útil durante o processo de aprendizagem, é necessário que haja redundância nos padrões de entrada”*. (BRAGA, 200, p. 101)

---

<sup>25</sup> Essa conversão é necessária pois o algoritmo proposto por Kohonen realiza operações matemáticas, logo, apenas números são válidos.

The screenshot shows a software application window for managing datasets. On the left, there are input fields for 'Title' (set to 'Flags') and 'Description' (empty), along with buttons for 'Line' and 'Column' selection. At the bottom left is a button for 'Import from .txt'. On the right is a large table containing data for various countries, with a navigation bar at the bottom.

	<b>name</b>	<b>landmass</b>	<b>area</b>	<b>language</b>	<b>religion</b>	<b>blue</b>	<b>bars</b>	<b>stripes</b>	<b>colours</b>
	Algeria	4	2388	8	2	0	2	0	3
	American-Samoa	6	0	1	1	1	0	0	5
	Andorra	3	0	6	0	1	3	0	3
	Angola	4	1247	10	5	0	0	2	3
	Anguilla	1	0	1	1	1	0	1	3
	Antigua-Barbuda	1	0	1	1	1	0	1	5
	Argentina	2	2777	2	0	1	0	3	2
	Australia	6	7690	1	1	1	0	0	3
	Austria	3	84	4	0	0	0	3	2
	Bahamas	1	19	1	1	1	0	3	3
	Bahrain	5	1	8	2	0	0	0	2
	Bangladesh	5	143	6	2	0	0	0	2

Figura 10: Imagem da tela de cadastro dos conjuntos de dados.

## 4.2 O treinamento da rede

A segunda seção relevante é a de treinamento da rede, onde se faz tanto a sua parametrização quanto a sua visualização. Para viabilizar a compreensão, todos os parâmetros possuem uma prática explicação sobre sua influência no treinamento. Escolhe-se qual é o conjunto de dados a ser treinado, a quantidade de treinamentos, a taxa de atualização da visualização, as taxas de aprendizagem e vizinhanças iniciais e finais e a dimensão da matriz de pesos. Tais parâmetros são geralmente conseguidos através de tentativa e erro.

**Training**

Data: Flags

Times: 50

Update at: 5 sec

**Exhibition**

Columns: 10

Lines: 20

**Weights**

Randomize?

**Neighbors** Randomize between minimum and maximum of each column  
Start: 15  
End: 1

**Learning (Alpha)**

Start: 0.01  
End: 0.0001

**Train**

Figura 11: Parametrização do treinamento.

Após a parametrização, o treinamento é salvo no banco de dados e inicia-se então o que pode-se chamar de primeira fase do treinamento: a criação das variáveis.

O primeiro passo dessa criação é o ajuste das entradas, onde as colunas irrelevantes são removidas e as entradas que não são numéricas (detectadas pelo uso de uma expressão regular) são convertidas através da soma dos valores ASCII<sup>26</sup> de cada caractere.

---

26 Sigla de “American Standard Code for Information Interchange” ou “Código Padrão Americano para o Intercâmbio de Informação” tradução nossa.

```

def create_inputs()
  @inputs = Dataset.find(dataset_id).lines_to_matrix
  @inputs.map! do |r|
    r.map! do |s|
      if s.to_s.match(/\A[+-]\d+?(.\d+)?\Z/) == nil
        s=s.unpack("C*").sum
      else
        s=s.to_f
      end
    end
  end
end

```

*Quadro 7: Conversão das entradas não numéricas.*

O segundo passo é a criação da matriz de pesos aleatórios.

Os pesos iniciais de uma rede do tipo SOM são definidos aleatoriamente. Por causa disso, muitos nodos podem apresentar vetores de pesos muito diferentes dos padrões de entrada. Dessa forma, pode ser que não haja o número necessário de nodos utilizáveis [...]. Em resultado, a rede pode ou não convergir ou apresentar ciclos muito lentos. (BRAGA, 2000, p. 116)

Embora existam diversas alternativas para a definição do valor desses pesos, não há um consenso sobre uma forma padrão de utilização. Partindo desse ponto, o usuário pode indicar o intervalo a ser utilizado ou mesmo deixar que o próprio sistema identifique o intervalo entre o mínimo e máximo de cada coluna. Levando em consideração a tabela abaixo, esse intervalo seria para *colors* entre 0 e 3, e para *stripes* entre 3 e 5.

Name	Colours	Stripes
Afghanistan	3	5
Albania	0	3
Algeria	0	3
American-Sam	0	5
Andorra	0	3
Angola	2	3

*Tabela 2: Dados para exemplificação da escolha do intervalo de pesos.*

O trecho de código a seguir demonstra como é gerada a matriz de pesos:

```

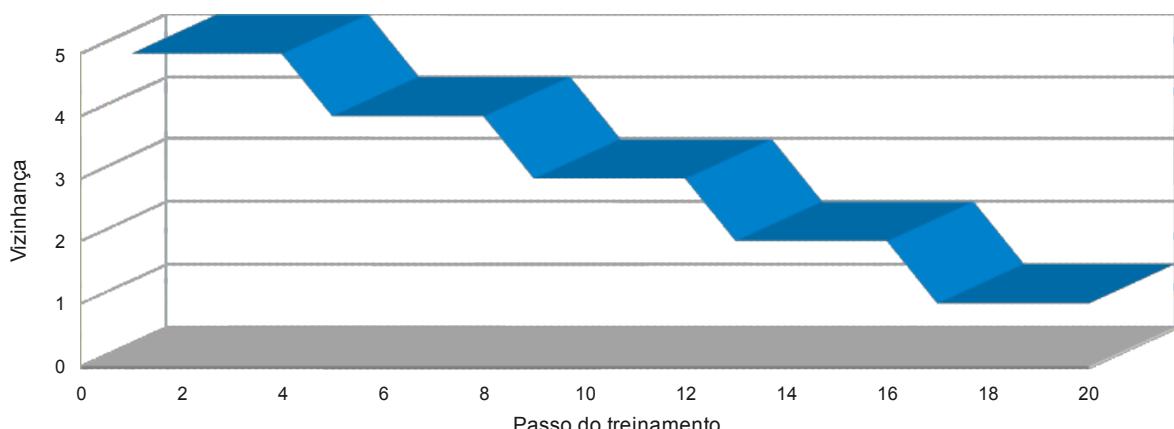
def create_weights
  if randomize_weights
    z_size = @inputs.max{|x,y| x.size<=>y.size}.size
    min_max = Array.new(z_size){[]}

    z_size.times do |i|
      @inputs.size.times do |j|
        min_max[i][0] = [ min_max[i][0], @inputs[j][i] ].min rescue @inputs[j][i]
        min_max[i][1] = [ min_max[i][1], @inputs[j][i] ].max rescue @inputs[j][i]
      end
    end
    @weights = Array.new(weight_columns){
      Array.new(weight_lines){
        Array.new(z_size){ |l| rand(min_max[i][1]-min_max[i][0])+min_max[i][0] }
      }
    }
  else
    @weights = Array.new(weight_columns){
      Array.new(weight_lines){
        Array.new(z_size){
          |l| rand(end_random_weights-start_random_weights)+start_random_weights
        }
      }
    }
  end
end

```

*Quadro 8: Criação da matriz de pesos.*

Após a criação da matriz de pesos cria-se um *array* de vizinhanças, de tamanho igual a quantidade de treinamentos, com os valores inteiros da distribuição linear entre a vizinhança final e inicial, tal como proposto por Kohonen ápud BRAGA (2000, p. 117), onde “[...] Durante o treinamento, a região de vizinhança é progressivamente reduzida até um limite pré-definido.”, com a ressalva de também haver a possibilidade de aumentar ao invés de diminuir a vizinhança; assim, existe uma posição no *array* de vizinhanças para cada passo. Um exemplo dessa distribuição pode ser notada no gráfico abaixo:



*Gráfico 3: Distribuição linear com a quantidade de treinamentos igual a 20, vizinhança inicial 5 e final 1.*

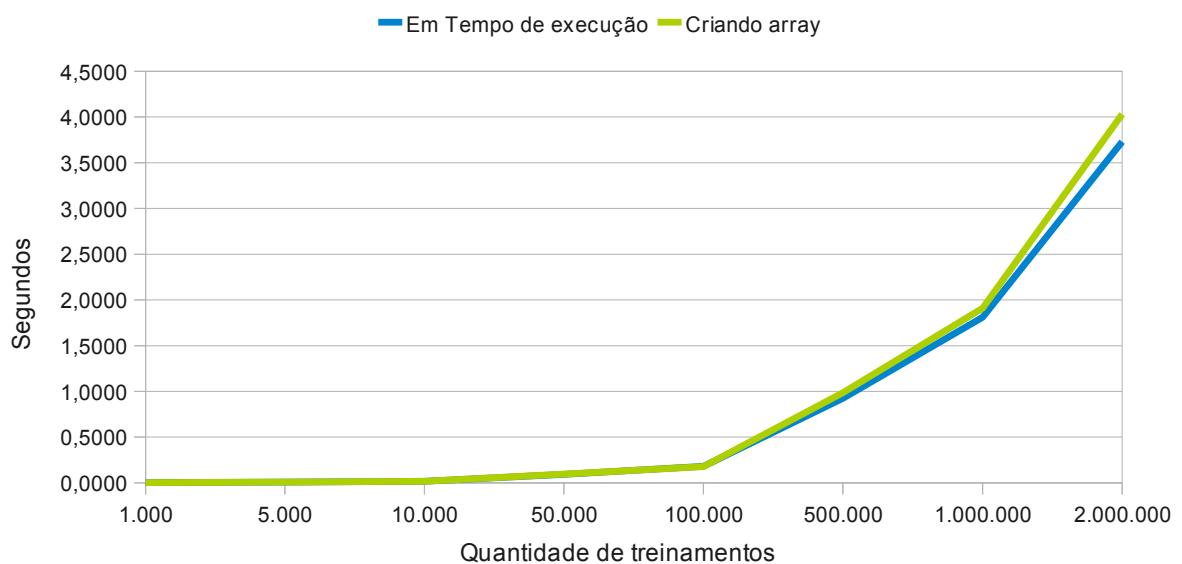
O último *array* a ser criado é o de taxas de aprendizado, que mantém a mesma ideia do de vizinhanças, utilizando o intervalo entre as taxas inicial e final e valores reais ao invés de inteiros. Uma importante restrição é que esse intervalo deve estar entre 0 e 1, visto que, “[...] a taxa de aprendizagem [...] é decrementada linearmente para zero durante o treinamento.” (KOHONEN, 2000, p. 322). Esse *array*, bem como o de vizinhanças, é criado da seguinte forma:

```
def distribute(start_value, end_value, array_size, want_ints)
    diff = 1.0 * (end_value - start_value)
    n = [array_size-1, 1].max

    (0..(array_size-1)).map { |i|
        v = start_value + i * diff / n
        want_ints ? v.round : v
    }
end
```

*Quadro 9: Criação de array através da distribuição de um intervalo.*

O motivo do salvamento, tanto das vizinhanças quanto das taxas de aprendizagem em um *array*, se faz pela viabilização da posterior criação dessas curvas de forma não linear (como será sugerido no item 6.1 - Trabalhos Futuros do Capítulo 6 - Conclusão), apesar do gráfico abaixo demonstrar que o cálculo do valor em tempo de execução ser ligeiramente mais rápido conforme a quantidade de treinamentos aumenta:



*Gráfico 4: Comparação entre o uso da vizinhança através de um array previamente criado e o cálculo em tempo de execução.*

Terminada a criação das variáveis, inicia-se o treinamento da rede que se repete na quantidade de vezes determinada pelo usuário. A cada passo avançado ajusta-se as respectivas taxas de aprendizado e vizinhança e executa-se a aprendizagem dos elementos.

```

def train
  @training_times.times do |time|
    @alpha = @alphas[time]
    @range = @ranges[time]

    @inputs.each do |input|
      closer = closer(input, @weights)
      learn(input, closer)
    end

    update_positions
  end
  finished = true
  save!
end

```

*Quadro 10: Treinamento da rede.*

A aprendizagem dos elementos é o momento culminante do treinamento, onde primeiramente acontece a busca do ponto mais próximo: é apresentada uma entrada e procura-se na matriz de pesos qual é o ponto com a menor distância euclidiana à ela. O cálculo é feito da seguinte maneira:

```

def closer(x, array)
  closer = [0, 0]
  array.each_index do |i|
    array[i].each_index do |j|
      p = [i, j]
      if array[i][j]==x
        closer = p
        break
      elsif
        euclidean_distance(x, array[p[0]][p[1]]) <
        euclidean_distance(x, array[closer[0]][closer[1]])
        closer = p
      end
    end
  end
  closer
end

```

*Quadro 11: Método que busca o ponto mais próximo na matriz de pesos “array” em relação a entrada “x”.*

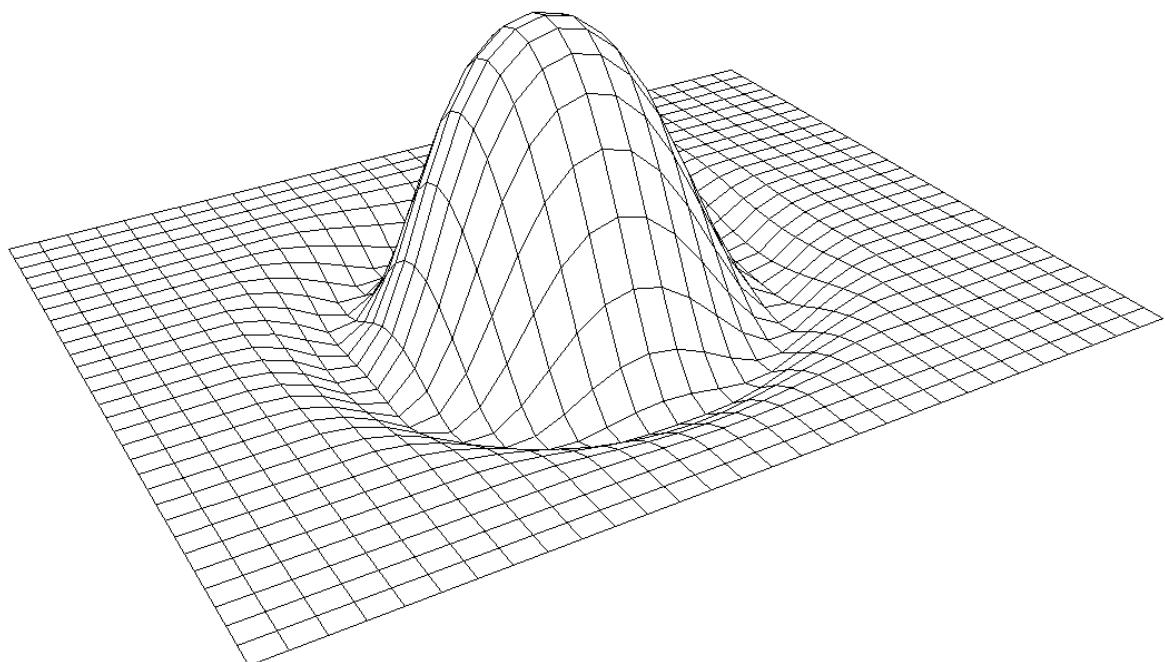
Ao encontrar o ponto mais próximo, inicia-se então a aprendizagem. Supondo que o taxa de aprendizagem atual seja 0.5 e a vizinhança 4, teríamos a seguinte

situação:

<b>Distância</b>	<b>Taxa de aprendizado aplicada</b>
0	0.5
1	0.25
2	0.17
3	0.13
4	0.1
5	-0.08
6	0
7	0
n...	0

*Tabela 3: Influência da taxa de aprendizado relativa à distância.*

A influência da taxa de aprendizagem sobre cada ponto é relativa à sua distância ao ponto mais próximo, ou seja, quanto mais próximo, maior a influência; quanto mais distante, menor a influência: o chamado chapéu mexicano. Aqueles que possuem a distância igual a vizinhança+1 também são influenciados, porém, negativamente, o que faz com que eles se distanciem da vizinhança afetada, resultando na diminuição da redundância da matriz de pesos. A tabela acima resultaria no mapeamento da seguinte forma:



*Figura 12: Chapéu mexicano.*

O método que aplica essa taxa de aprendizado ao neurônio pode ser visto a seguir:

```
def learn(input, closer)
  @weights.each_index do |i|
    @weights[i].each_index do |j|
      p = [i, j]

      if in_range(p, closer, @range)
        @weights[i][j].each_index do |k|
          @weights[i][j][k] += relative_alpha(closer, p) * (input[k] - @weights[i][j][k])
        end
      elsif index_distance(closer, p) == @range + 1
        @weights[i][j].each_index do |k|
          @weights[i][j][k] -= relative_alpha(closer, p) * (input[k] - @weights[i][j][k])
        end
      end
    end
  end
end
```

*Quadro 12: Aplicação da taxa de aprendizado.*

Ao final de cada ciclo de aprendizado, são salvos no banco de dados a posição cartesiana atual de cada um dos elementos bem como o passo em que se encontra o treinamento.

```
def update_positions(time)
  result = []
  @inputs.each_index do |i|
    point = closer(@inputs[i], @weights)
    result << { :index => i, :x => point[0], :y => point[1] }
  end
  results << result
  update_attributes(:results => results, :current_training_time => time)
end
```

*Quadro 13: Método que encontra a posição cartesiana de cada entrada e a salva no banco de dados.*

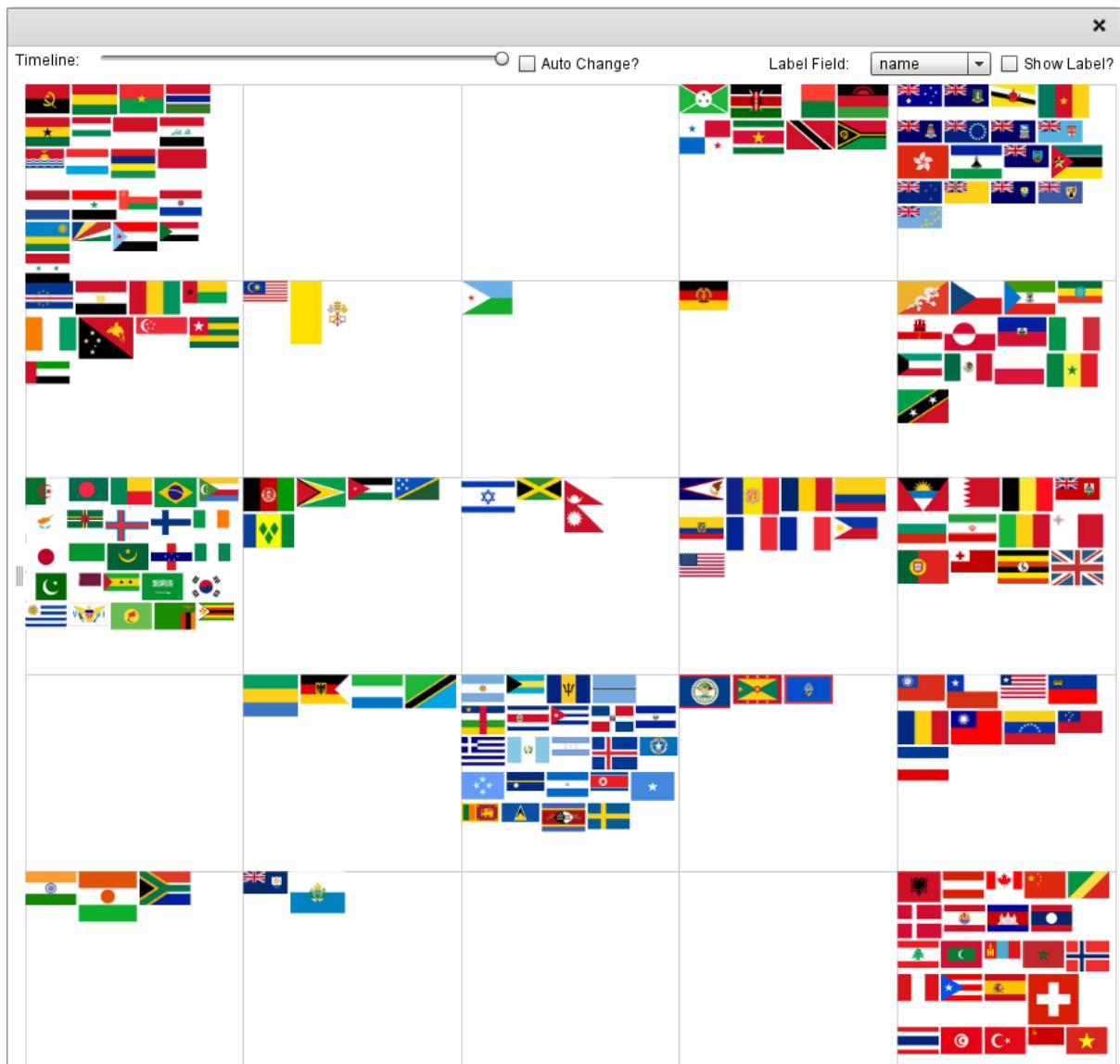
Quando todos os ciclos estiverem completos, o atributo “*finished*” recebe o valor “*true*” e é salvo no banco, finalizando assim o treinamento.

### 4.3 Os resultados

A apresentação dos resultados obtidos se faz de maneira interativa, sendo

que a partir da taxa de atualização da visualização (escolhida pelo usuário), o sistema faz uma verificação constante para descobrir qual a posição cartesiana de cada elemento e os posiciona para visualização. Vale ressaltar que esta verificação é feita enquanto a rede está em treinamento, pois ele [*o treinamento*] é executado em segundo plano permitindo assim ao programa responder a outras requisições ao invés de aguardar a finalização do processamento da rede (o que não é rápido).

Durante essa verificação se houver alterações nas posições, a movimentação é feita com uma transição, possibilitando ao usuário identificar visualmente quando algum elemento está trocando de grupo ou mesmo a formação dos grupos.



*Figura 13: Demonstração da exibição do resultado do treinamento.*

Quando se treina um conjunto de dados pequeno e conhecido, conseguimos entender o comportamento da rede visualmente, porém, quando o conjunto de dados é complexo, se torna interessante ao usuário comparar determinados elementos para entender assim porque um está próximo ou distante de outro. É possível fazer tal comparação através de uma tabela, que exibe todos os atributos de cada elemento.

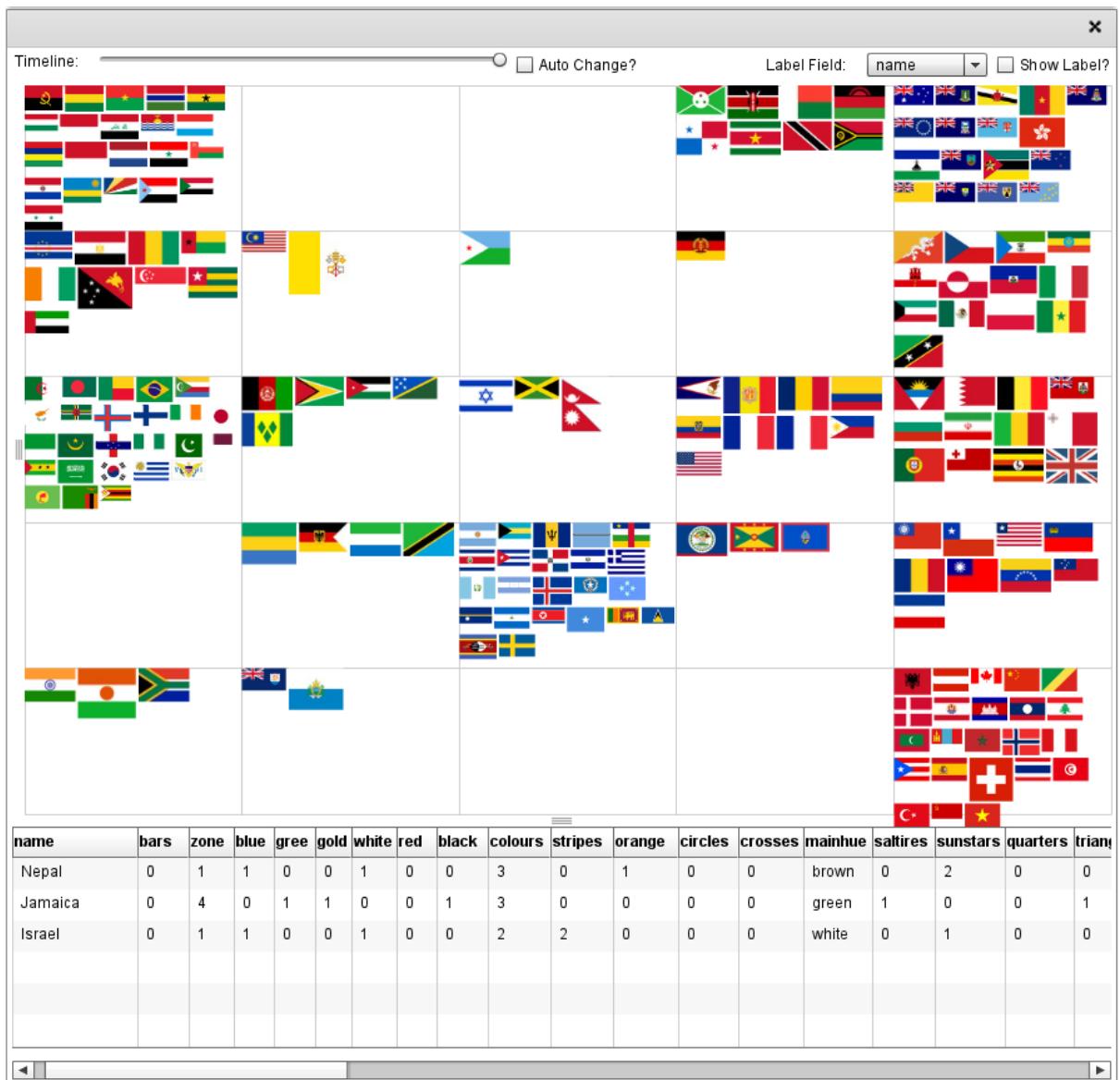


Figura 14: Comparação de elementos.

É possível também alterar o rótulo de exibição dos elementos em tempo de

execução, permitindo que (no caso das bandeiras) seja exibido nome do país ou qualquer outro atributo.

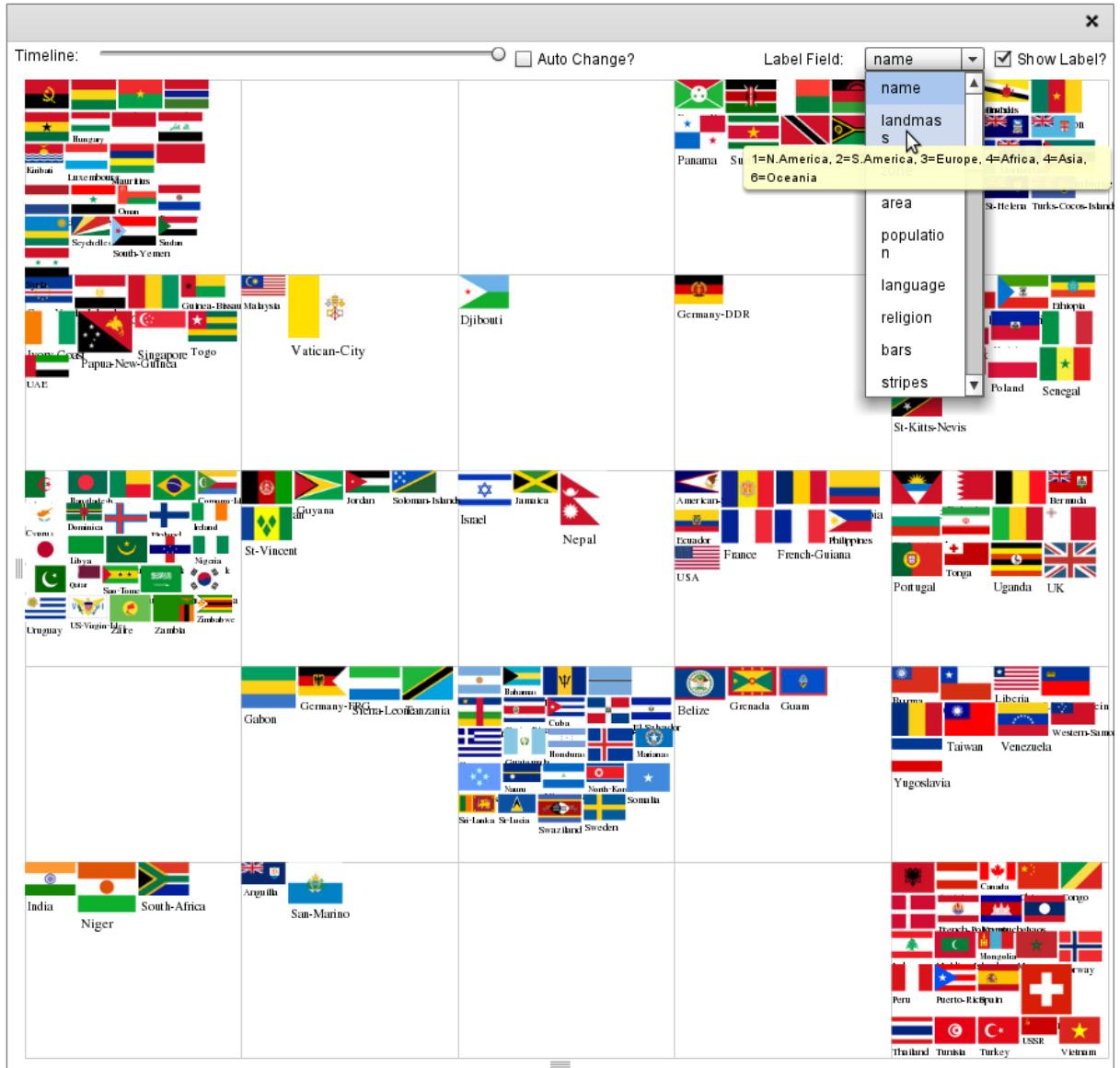


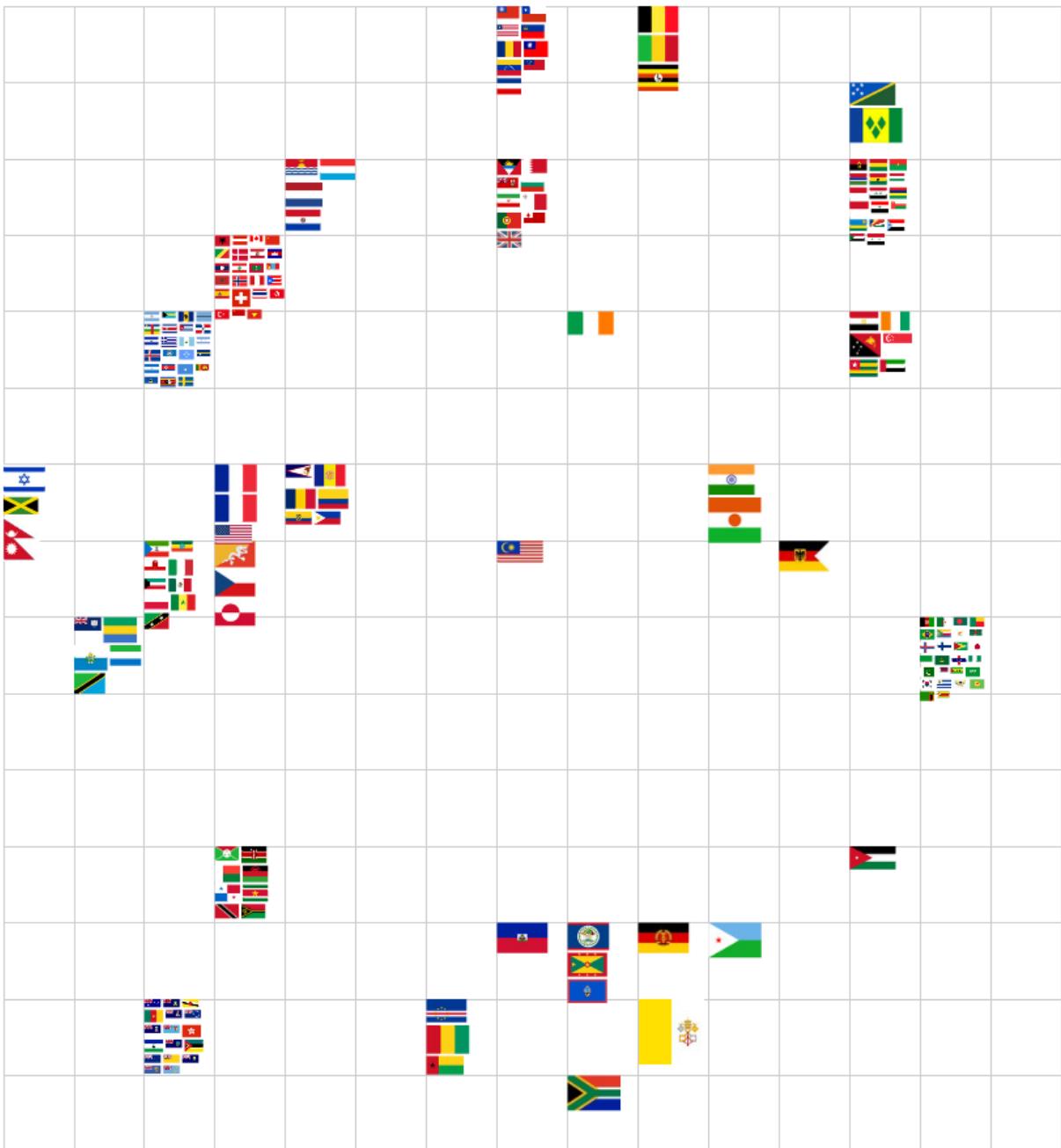
Figura 15: Alteração do rótulo de exibição.

## 5 DEMONSTRAÇÃO

Para demonstração dos resultados obtidos pelo treinamento, foi utilizado o conjunto de dados de bandeiras de países (como dito no Capítulo 4 - Desenvolvimento) e os seguintes parâmetros:

- Dimensão da matriz de pesos: 15x15
- Quantidade de treinamentos: 500
- Taxa de aprendizagem: 0.1 a 0.01
- Taxa de vizinhança: 7 a 2

Seria interessante, mas não necessário, que pudéssemos supor quais os agrupamentos a rede poderia encontrar ao final do treinamento. Contudo, essa suposição seria difícil pois a quantidade de bandeiras é grande (193) e muitas são as características analisadas (22). Esperou-se então que a rede apresentasse o melhor agrupamento visual possível, desconsiderando atributos que não tenham relevância visual no treinamento, como população, área, etc.



*Figura 16: Demonstração dos resultados: Passo 1/500.*

Nos primeiros passos do treinamento, pode-se identificar diversos grupos, dentre eles: França e Guiana Francesa - pois tratam-se da mesma bandeira; Taiwan, Chile, Burma, Samoa, etc. - pois tem basicamente a mesma estrutura: símbolo no canto superior esquerdo e predominância da cor vermelha; Níger e Índia – pois são praticamente idênticas, apenas o símbolo central se difere. Vale ressaltar que nesse ponto do treinamento os grupos que a rede consegue identificar não necessariamente se manterão até o final do treinamento.

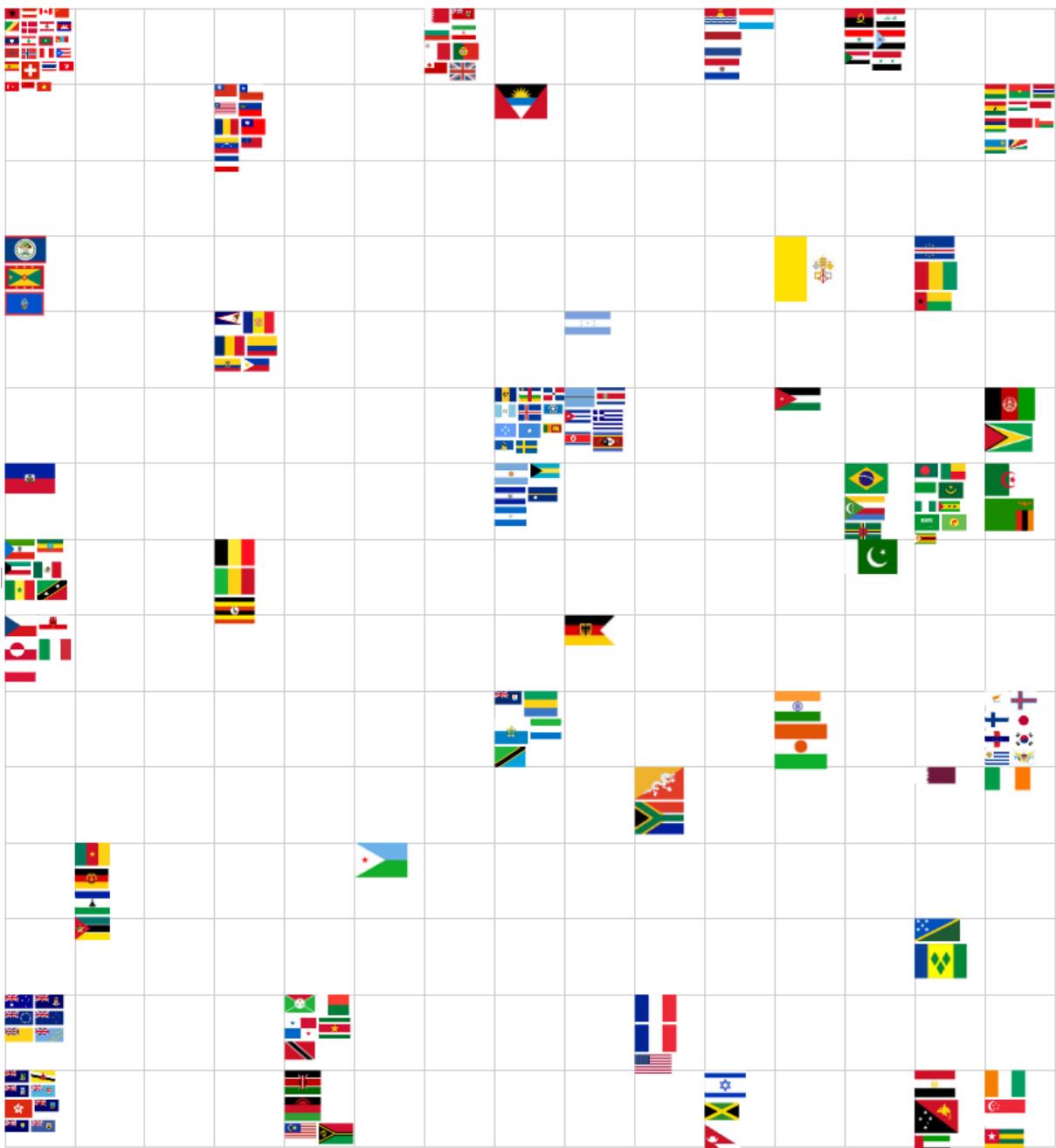


Figura 17: Demonstração dos resultados: Passo 150/500.

No passo 150, o objetivo ainda é a identificação dos grupos. Pode-se notar que os citados anteriormente se mantém, porém em posições distintas, e que a formação de novos é perceptível: Iraque, Síria, Yemen do Norte, etc. – tem a mesma disposição (horizontal) e cores (vermelho, branco e preto) das listras; Argentina, Nicarágua e El Salvador – que se diferem do grupo anterior basicamente pelas cores.

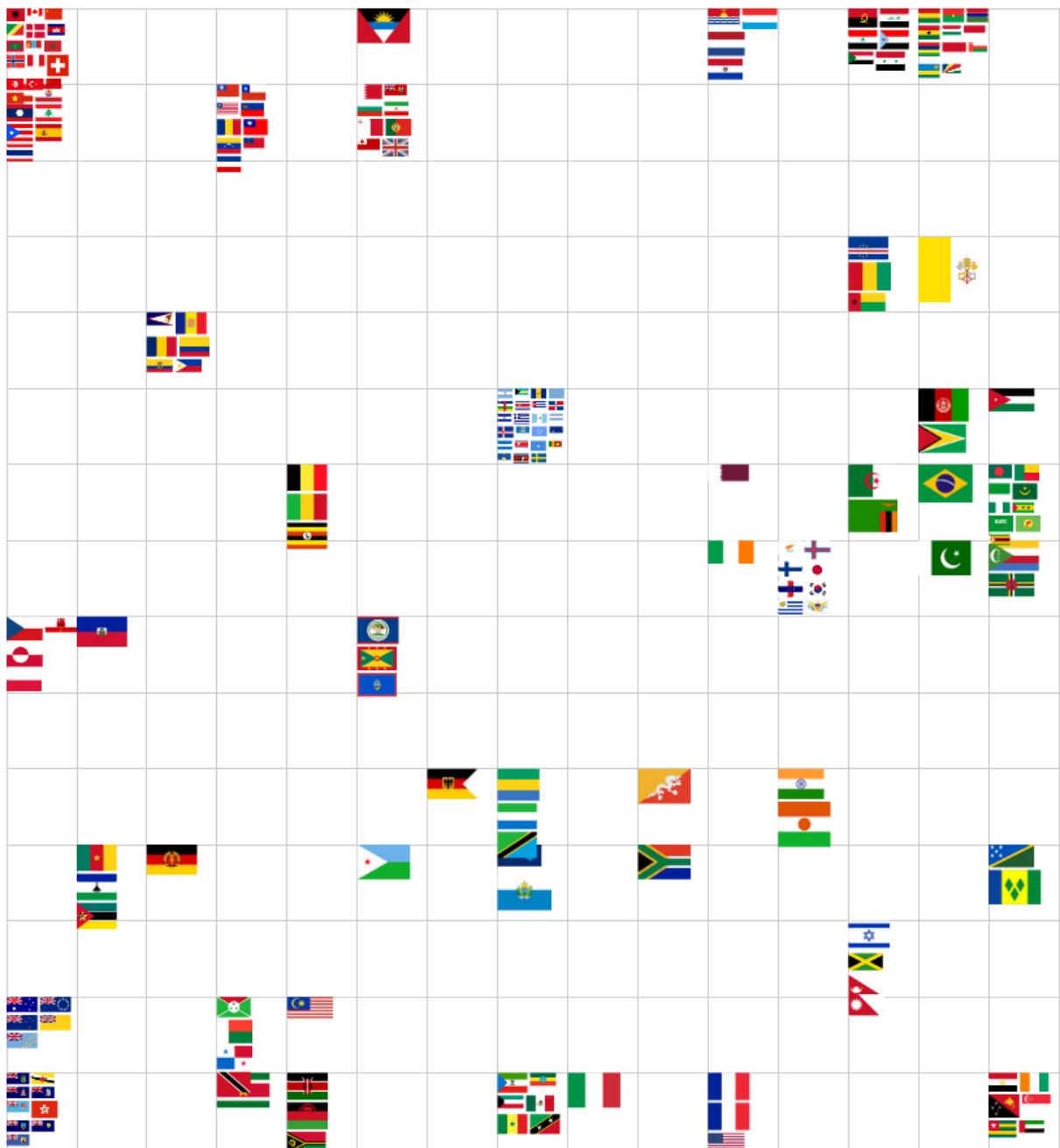


Figura 18: Demonstração dos resultados: Passo 300/500.

Ao passo 300 estamos um pouco à frente da metade do treinamento, logo, temos praticamente a total definição dos agrupamentos sendo que próximo a esse passo o objetivo passa a ser o de sofisticação desses grupos. Assim, pode-se notar a presença de um outro forte grupo: Austrália, Ilhas Caimã, Nova Zelândia, Niue, etc. pois contém parte da bandeira do Reino Unido (isso porque a maioria desses países são ex-colônias do Império Britânico).

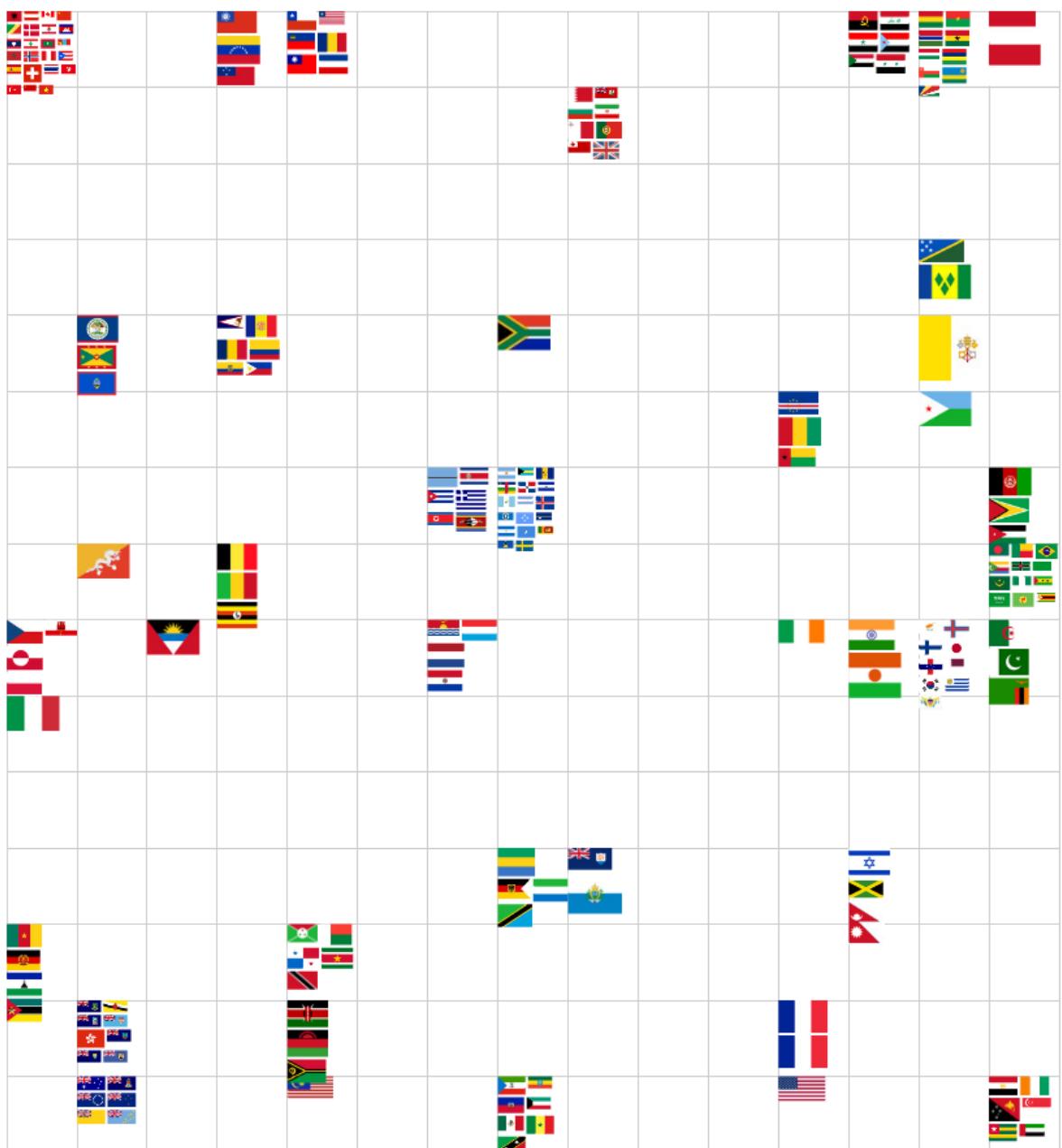


Figura 19: Demonstração dos resultados: Passo 450/500.

Do passo 300 ao 450 não existem grandes mudanças, pois muitas das bandeiras nesse ponto já tem a sua posição praticamente definida. O que se faz aqui é apenas um melhoramento que resulta em uma distribuição dos grupos dentro de uma vizinhança próxima.

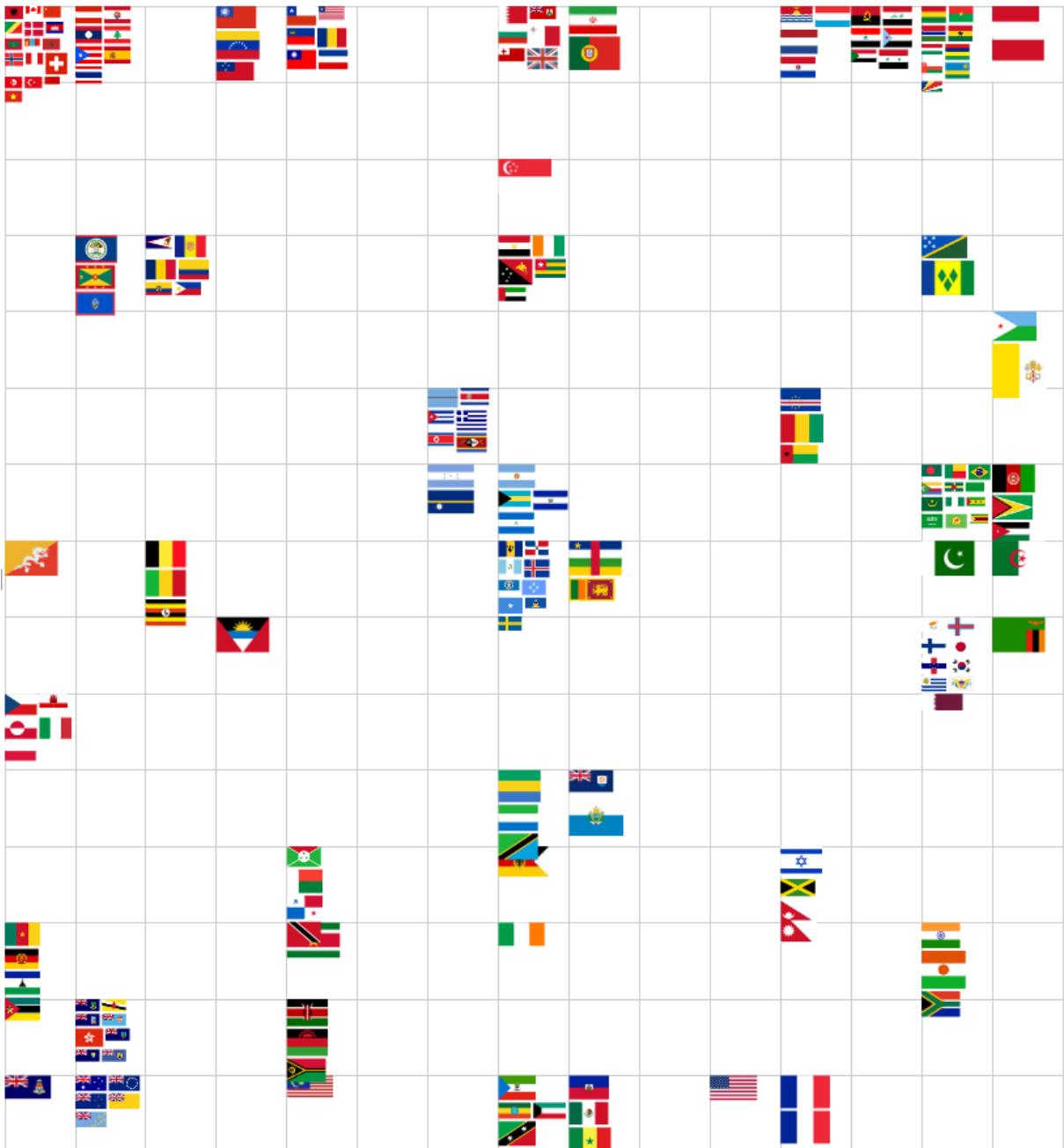


Figura 20: Demonstração dos resultados: Passo 500/500.

Ao final do treinamento conseguimos identificar diversos aspectos: ao centro fincaram-se em geral as bandeiras com predominância azul e ao seu lado direito, as com predominância verde; ao topo, bandeiras com grande incidência de cor vermelha: Suíça, Albânia, Áustria, etc; abaixo das verdes, um pequeno grupo onde a principal característica visual é a predominância da cor banca. Notou-se ainda que as bandeiras do Vaticano e de Butão ficaram isoladas em praticamente todos os treinamentos e que as bandeiras do Nepal e

de Israel estiveram sempre isoladas das demais, porém próximas uma da outra.

O desenrolar do treinamento acaba por exemplificar as duas fases teóricas citadas no Capítulo 2 - Redes Neurais Artificiais, pois no início do treinamento a rede identificou os grupos das bandeiras e os sofisticou no decorrer do mesmo.

## 5.1 Outros resultados

Afim de averiguar qual poderia ser o comportamento da rede quando treinada com parâmetros diferentes do teste anterior, e qual a variação desse treinamento quando executados por diversas vezes com os mesmos parâmetros, foram executadas uma sequência de 3 testes, com os seguintes parâmetros:

- Dimensão da matriz de pesos: 10x10
- Quantidade de treinamentos: 200
- Taxa de aprendizagem: 0.1 a 0.01
- Taxa de vizinhança: 10 a 1

Nota-se que no primeiro passo, todos os treinamentos descobriram praticamente os mesmos grupos, porém, em posições cartesianas diferentes, como se pode ver nas figuras abaixo:

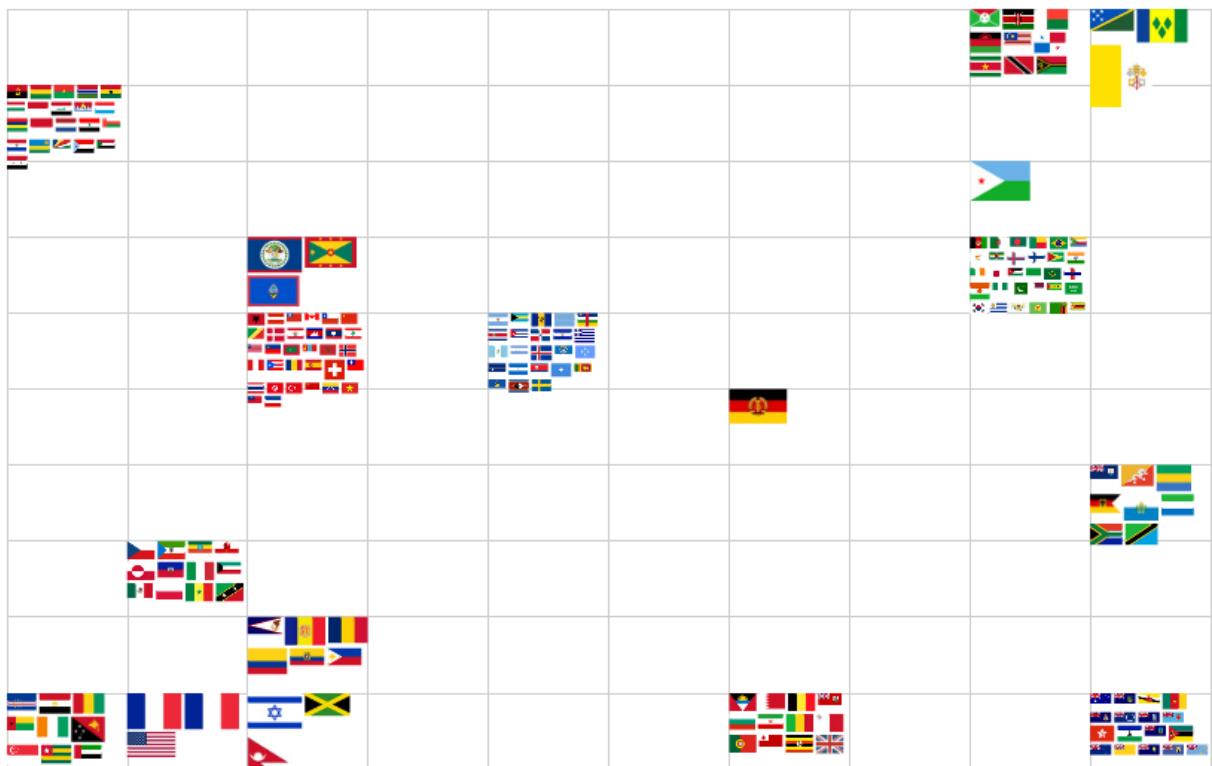


Figura 21: Outros resultados: Teste 1 - Passo 1/200.

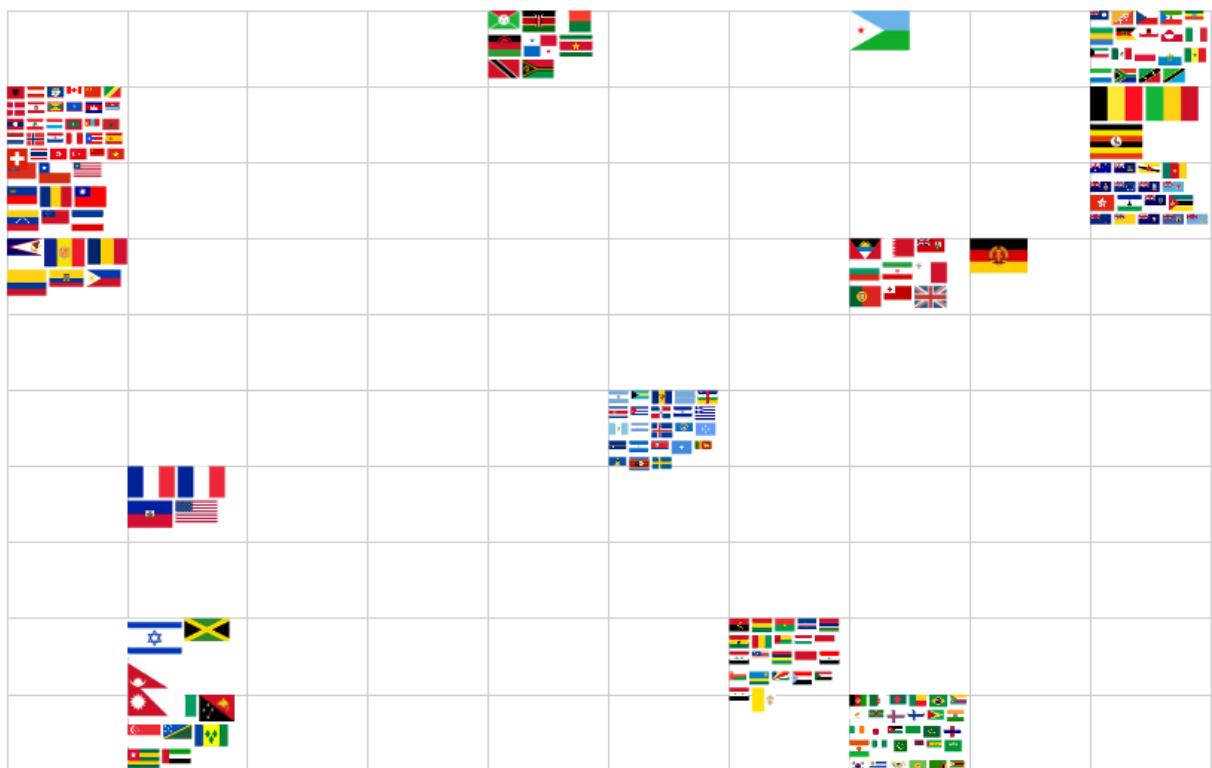
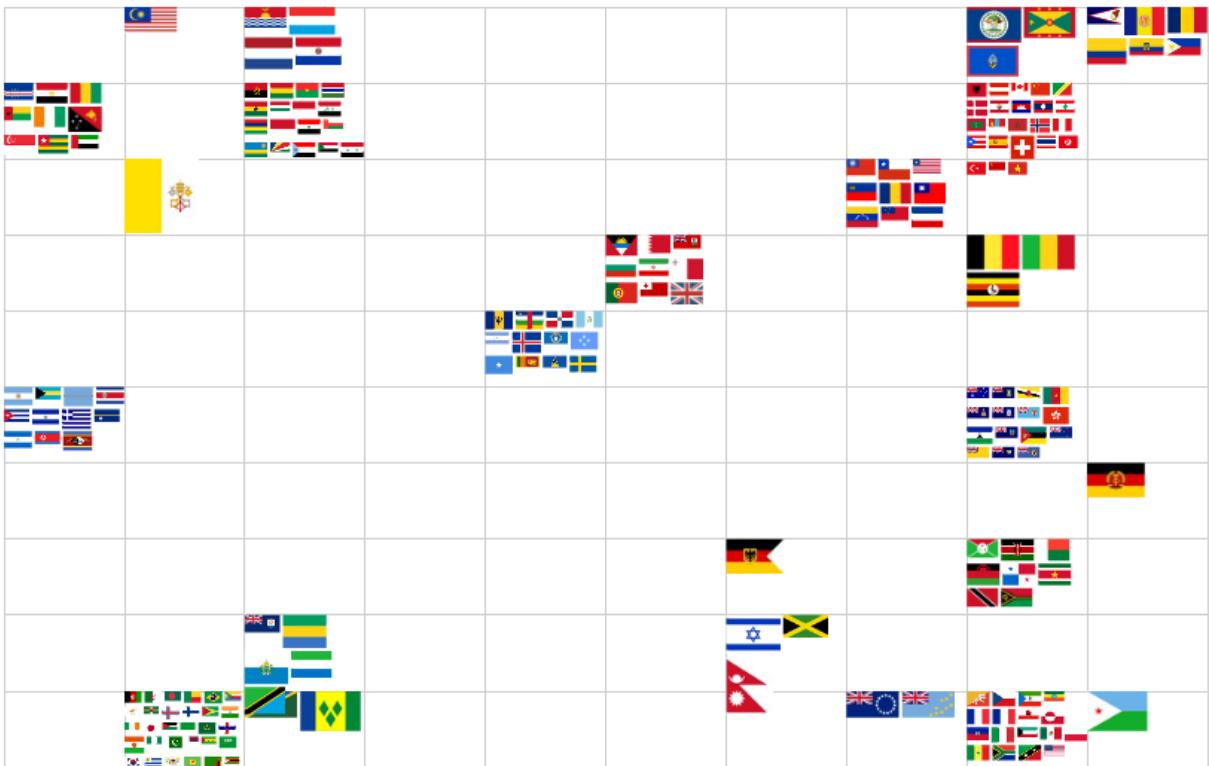


Figura 22: Outros resultados: Teste 2 - Passo 1/200.



*Figura 23: Outros resultados: Teste 3 - Passo 1/200.*

No final do treinamento confirmamos que os grupos são, como suposto, praticamente os mesmos. Contudo, existem diferenças em relação a primeira demonstração: as bandeiras do Vaticano e de Butão, que sempre se diferiam das outras, acabaram se aproximando ou mesmo se enquadrando em determinados grupos.

Pela matriz de pesos ter uma dimensão menor que na primeira demonstração, conseguimos distinguir melhor os grupos ao final do treinamento, porém, em geral os todos resultados foram similares à primeira demonstração.



Figura 24: Outros resultados: Teste 1 - Passo 200/200.



Figura 25: Outros resultados: Teste 2 - Passo 200/200.



Figura 26: Outros resultados: Teste 3 - Passo 200/200.

## 6 CONCLUSÃO

As redes de Kohonen são de grande auxílio na solução de problemas que o homem sozinho demoraria a solucionar com qualidade satisfatória. A exemplo das bandeiras, dificilmente conseguiríamos analisar todas as suas características e agrupá-las tão bem quanto o programa, isso porque, enquanto nós analisamos a bandeira apenas visualmente, o programa tem uma visão profunda das mesmas.

O mais surpreendente dessa rede, e talvez uma quebra de paradigmas a olhares comuns, é a sua capacidade de aprender tal qual o cérebro humano aprenderia de maneira computacionalmente trivial. Essa aparente trivialidade aliada a uma série de possíveis utilizações, despertou o interesse de implementar o algoritmo na plataforma web, o que acarretou no uso de tecnologias não convencionais para esse tipo de aplicação, haja visto o seu alto custo computacional. Embora essas tecnologias não tenham um bom desempenho (para esse tipo de aplicação), elas são extensíveis, o que nos permitiu implementar alguns métodos em C++, tornando assim o desempenho do software satisfatório.

A aplicação desenvolvida, suporta o treinamento de uma ampla gama de conjuntos de dados permitindo assim seu uso em diversas áreas. Sua disponibilidade na internet aliada a uma interface simples e usável, almeja propiciar sua larga utilização e vislumbramento de resultados a qualquer tipo de pessoa, esteja ela ou não, ligada a Inteligência Artificial. Espera-se assim, que professores possam a utilizar para descobrir características de seus alunos e ajudar sua melhora acadêmica; que gerentes de empresas consigam agrupar os elementos de sua equipe afim de melhorar a sua produção; que gestores de estoque consigam melhorar e tornar mais efetiva sua disposição; que centros de controle de vetores e zoonoses possam identificar semelhanças entre bairros com focos de doenças; e tantos outros quanto a imaginação permitir.

### 6.1 Trabalhos Futuros

Dezenas de melhorias podem ser implementadas pois as Redes SOM possuem diversas possibilidades, e as tecnologias aqui utilizadas, permitem que esse software atenda a elas. Como sugestão dessas possibilidades, pode-se citar:

- Adição de mais um eixo para tornar a visualização tridimensional;

- possibilidade de criação gráfica da curva de aprendizagem e da vizinhança do treinamento;
- implementações de outras fórmulas de cálculo de vizinhança, como a de hexágono ou círculo (gaussiana);
- ajustes para melhoria de performance, como por exemplo a tradução de outros métodos relacionados ao treinamento para C++;
- criação de um aplicativo utilizando o framework Air, possibilitando assim a execução independente de conexões com a internet;
- comparação dos treinamentos de cada conjunto de dados.

## BIBLIOGRAFIA

BRAGA, Antônio P.; LUDEMIR, Teresa B.; CARVALHO, André Carlos P. L. F. *Redes Neurais Artificiais: Teoria e aplicações*. Rio de Janeiro: LTC Editora. 2000.

FERNANDES, Anita M. R. *Inteligência Artificial: Noções gerais*. Florianópolis: Visual Books. 2003.

GOMES, Marcio R. N. *Análise de desempenho acadêmico utilizando redes neurais artificiais*. 2006. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Instituto de Ensino Superior Thathi, Araçatuba, 2006.

HAINES, Kirk. *A Light in the Darkness! (Key Value Stores Part 5)*. **Engine Yard**, San Fransico, set. 2009. Seção Blog. Disponível em: <<http://www.engineyard.com/blog/2009/mongodb-a-light-in-the-darkness-key-value-stores-part-5>>. Acesso em: 29 nov. 2009.

HANSSON, David H. *Broadcasting Brain Loud Thinking*. Disponível em: <<http://www.loudthinking.com>>. Acesso em: 20 nov. 2009.

KOHONEN, Teuvo. *Self-Organizing Maps*. 3 ed. New York: SPRINGER. 2000.

KRUG, Steve. *Não me faça pensar: Uma Abordagem de Bom Senso à Usabilidade na Web*. 2 ed. Rio de Janeiro: ALTA BOOKS. 2006.

MONGODB. *Combining the best features of document databases, key-value stores, and RDBMSes*. Disponível em: <<http://www.mongodb.org>>. Acesso em: 28 nov. 2009.

POOLE, David; MACKWORTH, Alan; GOEBEL, Randy. *Computational Intelligence: A Logical Approach*. New York: Oxford University Press. 1998. Disponível em: <<http://www.cs.ubc.ca/spider/poole/ci/ch1.pdf>>. Acesso em: 7 dez. 2009.

RAILS GUIDES. *Getting Started with Rails.* Disponível em: <[http://guides.rubyonrails.org/getting\\_started.html](http://guides.rubyonrails.org/getting_started.html)>. Acesso em: 28 nov. 2009.

RUBY: *A Programmer' s Best Friend.* Seção About. Disponível em: <<http://www.ruby-lang.org/en/about/>> Acesso em: 20 nov. 2009.

SCOHOLARPEDIA. *Kohonen network.* Disponível em: <[http://www.scholarpedia.org/article/Kohonen\\_network](http://www.scholarpedia.org/article/Kohonen_network)>. Acesso em 7 dez. 2009.

SEPPAT, Nico: *Bancos de dados não relacionais e o movimento NoSQL.* **Caelum**, São Paulo, out. 2009. Seção Blog. Disponível em: <<http://blog.caelum.com.br/2009/10/30/bancos-de-dados-nao-relacionais-e-o-movimento-nosql>>. Acesso em: 29 nov. 2009.

YOVITS, Marshall C. *Advances in Computers.* Direção geral de Marvin V. Zelkowitz. San Diego: Academic Press Inc. 1993. v. 77, n. 36.

WIKIPEDIA. *NoSQL.* Disponível em: <<http://en.wikipedia.org/wiki/NoSQL>>. Acesso em 7 dez. 2009.

## Indicações de leitura

RUBY. *A Programmer' s Best Friend.* Disponível em: <<http://www.ruby-lang.org>>.

BASECAMP. *Project managment, collaboration and task software.* Disponível em: <<http://basecamphq.com/>>.

GOOGLE RESEARCH PUBLICATION. *Bigtable.* Disponível em: <<http://labs.google.com/papers/bigtable.html>>.

VOGELS, Werner. *Amazon's Dynamo. All Things Distributed*, Seattle, out. 2007. Disponível em: <[http://www.allthingsdistributed.com/2007/10/amazons\\_dynamo.html](http://www.allthingsdistributed.com/2007/10/amazons_dynamo.html)>.

APACHE. *Hadoop*. Disponível em: <<http://hadoop.apache.org>>.

CASSANDRA. *A highly scalable, eventually consistent, distributed, structured key-value store*. Disponível em: <<http://incubator.apache.org/cassandra/>>.

PROJECT VOLDEMORT. *A distributed database*. Disponível em: <<http://project-voldemort.com>>.

TOKYO CABINET. *A modern implementation of DBM*. Disponível em: <<http://1978th.net/tokyocabinet>>.

**ANEXOS**

## ANEXO A – Conjunto de dados de bandeiras de países: Colunas

Attribute	Description	Included in training?
name	Name of the country concerned	false
landmass	1=N.America, 2=S.America, 3=Europe, 4=Africa, 4=Asia, 6=Oceania	false
zone	Geographic quadrant, based on Greenwich and the Equator 1=NE, 2=SE, 3=SW, 4=NW	false
area	in thousands of square km	false
population	in round millions	false
language	1=English, 2=Spanish, 3=French, 4=German, 5=Slavic, 6=Other Indo-European, 7=Chinese, 8=Arabic, 9=Japanese/Turkish/Finnish/Magyar, 10=Others	false
religion	0=Catholic, 1=Other Christian, 2=Muslim, 3=Buddhist, 4=Hindu, 5=Ethnic, 6=Marxist, 7=Others	false
bars	Number of vertical bars in the flag	true
stripes	Number of horizontal stripes in the flag	true
colours	Number of different colours in the flag	true
red	0 if red absent, 1 if red present in the flag	true
gree	same for green	true
blue	same for blue	true
gold	same for gold (also yellow)	true
white	same for white	true
black	same for black	true
orange	same for orange (also brown)	true
mainhue	predominant colour in the flag (tie-breaks decided by taking the topmost hue, if that fails then the most central hue, and if that fails the leftmost hue)	false
circles	Number of circles in the flag	true
crosses	Number of (upright) crosses	true
saltires	Number of diagonal crosses	true
quarters	Number of quartered sections	true
sunstars	Number of sun or star symbols	true
crescent	1 if a crescent moon symbol present, else 0	true
triangle	1 if any triangles present, 0 otherwise	true
icon	1 if an inanimate image present (e.g., a boat), otherwise 0	true
animate	1 if an animate image (e.g., an eagle, a tree, a human hand) present, 0 otherwise	true
text	1 if any letters or writing on the flag (e.g., a motto or slogan), 0 otherwise	true

## ANEXO B – Conjunto de dados de bandeiras de países: Linhas

Afghanistan,5,1,648,16,10,2,0,3,5,1,1,0,1,1,1,0,green,0,0,0,0,1,0,0,1,0,0,black,green  
 Albania,3,1,29,3,6,6,0,0,3,1,0,0,1,0,1,red,0,0,0,0,1,0,0,0,1,0,red,red  
 Algeria,4,1,2388,20,8,2,2,0,3,1,1,0,0,1,0,0,green,0,0,0,0,1,1,0,0,0,0,green,white  
 American-Samoa,6,3,0,0,1,1,0,0,5,1,0,1,1,1,0,1,blue,0,0,0,0,0,0,1,1,1,0,blue,red  
 Andorra,3,1,0,0,6,0,3,0,3,1,0,1,1,0,0,0,gold,0,0,0,0,0,0,0,0,0,0,blue,red  
 Angola,4,2,1247,7,10,5,0,2,3,1,0,0,1,0,1,red,0,0,0,0,1,0,0,1,0,0,red,black  
 Anguilla,1,4,0,0,1,1,0,1,3,0,0,1,0,1,white,0,0,0,0,0,0,1,0,white,blue  
 Antigua-Barbuda,1,4,0,0,1,1,0,1,5,1,0,1,1,1,0,red,0,0,0,0,1,0,1,0,0,0,black,red  
 Argentina,2,3,2777,28,2,0,0,3,2,0,0,1,0,1,0,0,blue,0,0,0,0,0,0,0,0,0,blue,blue  
 Australia,6,2,7690,15,1,1,0,0,3,1,0,1,0,1,0,0,blue,0,1,1,1,0,0,0,0,0,white,blue  
 Austria,3,1,84,8,4,0,0,3,2,1,0,0,0,1,0,0,red,0,0,0,0,0,0,0,0,0,0,red,red  
 Bahamas,1,4,19,0,1,1,0,3,3,0,0,1,1,0,1,blue,0,0,0,0,0,1,0,0,0,blue,blue  
 Bahrain,5,1,1,0,8,2,0,0,2,1,0,0,0,1,0,0,red,0,0,0,0,0,0,0,0,0,white,red  
 Bangladesh,5,1,143,90,6,2,0,0,2,1,1,0,0,0,0,0,green,1,0,0,0,0,0,0,0,0,green,green  
 Barbados,1,4,0,0,1,1,3,0,3,0,0,1,1,0,1,blue,0,0,0,0,0,0,1,0,0,blue,blue  
 Belgium,3,1,31,10,6,0,3,0,3,1,0,0,1,0,1,0,gold,0,0,0,0,0,0,0,0,0,black,red  
 Belize,1,4,23,0,1,1,0,2,8,1,1,1,1,1,1,blue,1,0,0,0,0,0,0,1,1,1,red,red  
 Benin,4,1,113,3,3,5,0,0,2,1,1,0,0,0,0,green,0,0,0,0,1,0,0,0,0,0,green,green  
 Bermuda,1,4,0,0,1,1,0,0,6,1,1,1,1,1,0,red,1,1,1,1,0,0,1,1,0,white,red  
 Bhutan,5,1,47,1,10,3,0,0,4,1,0,0,0,1,1,1,orange,4,0,0,0,0,0,0,0,1,0,orange,red  
 Bolivia,2,3,1099,6,2,0,0,3,3,1,1,0,1,0,0,0,red,0,0,0,0,0,0,0,0,0,red,green  
 Botswana,4,2,600,1,10,5,0,5,3,0,0,1,0,1,1,0,blue,0,0,0,0,0,0,0,0,0,blue,blue  
 Brazil,2,3,8512,119,6,0,0,0,4,0,1,1,1,1,0,0,green,1,0,0,0,22,0,0,0,0,1,green,green  
 British-Virgin-Isles,1,4,0,0,1,1,0,0,6,1,1,1,1,1,0,1,blue,0,1,1,1,0,0,0,1,1,1,white,blue  
 Brunei,5,1,6,0,10,2,0,0,4,1,0,0,1,1,1,0,gold,0,0,0,0,0,0,1,1,1,1,white,gold  
 Bulgaria,3,1,111,9,5,6,0,3,5,1,1,1,1,0,0,red,0,0,0,0,1,0,0,1,1,0,white,red  
 Burkina,4,4,274,7,3,5,0,2,3,1,1,0,1,0,0,red,0,0,0,0,1,0,0,0,0,0,red,green  
 Burma,5,1,678,35,10,3,0,0,3,1,0,1,0,1,0,0,red,0,0,0,1,14,0,0,1,1,0,blue,red  
 Burundi,4,2,28,4,10,5,0,0,3,1,1,0,0,1,0,0,red,1,0,1,0,3,0,0,0,0,0,white,white  
 Cameroon,4,1,474,8,3,1,3,0,3,1,1,0,1,0,0,0,gold,0,0,0,0,1,0,0,0,0,0,green,gold  
 Canada,1,4,9976,24,1,1,2,0,2,1,0,0,0,1,0,0,red,0,0,0,0,0,0,0,1,0,red,red  
 Cape-Verde-Islands,4,4,4,0,6,0,1,2,5,1,1,0,1,0,1,1,0,1,1,0,1,0,1,0,red,green  
 Cayman-Islands,1,4,0,0,1,1,0,0,6,1,1,1,1,0,1,blue,1,1,1,1,4,0,0,1,1,1,white,blue  
 Central-African-Republic,4,1,623,2,10,5,1,0,5,1,1,1,1,0,0,gold,0,0,0,0,1,0,0,0,0,0,blue,gold  
 Chad,4,1,1284,4,3,5,3,0,3,1,0,1,1,0,0,0,gold,0,0,0,0,0,0,0,0,0,0,blue,red  
 Chile,2,3,757,11,2,0,0,2,3,1,0,1,0,0,red,0,0,0,1,1,0,0,0,0,0,blue,red  
 China,5,1,9561,1008,7,6,0,0,2,1,0,0,1,0,0,0,red,0,0,0,0,5,0,0,0,0,0,0,red,red  
 Colombia,2,4,1139,28,2,0,0,3,3,1,0,1,1,0,0,0,gold,0,0,0,0,0,0,0,0,0,0,gold,red  
 Comoros-Islands,4,2,2,0,3,2,0,0,2,0,1,0,0,1,0,0,green,0,0,0,0,4,1,0,0,0,0,green,green  
 Congo,4,2,342,2,10,5,0,0,3,1,1,0,1,0,0,0,red,0,0,0,0,1,0,0,1,1,0,red,red  
 Cook-Islands,6,3,0,0,1,1,0,0,4,1,0,1,0,1,0,blue,1,1,1,1,15,0,0,0,0,0,white,blue  
 Costa-Rica,1,4,51,2,2,0,0,5,3,1,0,1,0,1,0,0,blue,0,0,0,0,0,0,0,0,0,blue,blue  
 Cuba,1,4,115,10,2,6,0,5,3,1,0,1,0,1,0,0,blue,0,0,0,0,1,0,1,0,0,blue,blue  
 Cyprus,3,1,9,1,6,1,0,0,3,0,1,0,1,1,0,0,white,0,0,0,0,0,0,1,1,0,white,white  
 Czechoslovakia,3,1,128,15,5,6,0,0,3,1,0,1,0,1,0,0,white,0,0,0,0,0,0,1,0,0,0,white,red  
 Denmark,3,1,43,5,6,1,0,0,2,1,0,0,0,1,0,0,red,0,1,0,0,0,0,0,0,0,0,red,red  
 Djibouti,4,1,22,0,3,2,0,0,4,1,1,1,0,1,0,0,blue,0,0,0,0,1,0,1,0,0,0,white,green  
 Dominica,1,4,0,0,1,1,0,0,6,1,1,1,1,1,0,green,1,0,0,0,10,0,0,0,1,0,green,green  
 Dominican-Republic,1,4,49,6,2,0,0,0,3,1,0,1,0,1,0,0,blue,0,1,0,0,0,0,0,0,0,blue,blue  
 Ecuador,2,3,284,8,2,0,0,3,3,1,0,1,1,0,0,0,gold,0,0,0,0,0,0,0,0,0,gold,red  
 Egypt,4,1,1001,47,8,2,0,3,4,1,0,0,1,1,1,0,black,0,0,0,0,0,0,0,0,1,1,red,black  
 El-Salvador,1,4,21,5,2,0,0,3,2,0,0,1,0,1,0,blue,0,0,0,0,0,0,0,0,0,0,blue,blue  
 Equatorial-Guinea,4,1,28,0,10,5,0,3,4,1,1,1,0,1,0,0,green,0,0,0,0,0,0,1,0,0,0,green,red  
 Ethiopia,4,1,1222,31,10,1,0,3,3,1,1,0,1,0,0,0,green,0,0,0,0,0,0,0,0,0,0,green,red  
 Faeroes,3,4,1,0,6,1,0,0,3,1,0,1,0,1,0,0,white,0,1,0,0,0,0,0,0,0,0,white,white  
 Falklands-Malvinas,2,3,12,0,1,1,0,0,6,1,1,1,1,0,0,blue,1,1,1,1,0,0,0,1,1,1,white,blue  
 Fiji,6,2,18,1,1,1,0,0,7,1,1,1,1,0,1,blue,0,2,1,1,0,0,0,1,1,0,white,blue  
 Finland,3,1,337,5,9,1,0,0,2,0,0,1,0,1,0,0,white,0,1,0,0,0,0,0,0,0,0,white,white  
 France,3,1,547,54,3,0,3,1,0,1,0,1,0,0,white,0,0,0,0,0,0,0,0,0,0,blue,red  
 French-Guiana,2,4,91,0,3,0,3,1,0,1,0,1,0,0,white,0,0,0,0,0,0,0,0,0,0,blue,red  
 French-Polynesia,6,3,4,0,3,0,0,3,5,1,0,1,1,1,0,red,1,0,0,0,1,0,0,1,0,0,red,red  
 Gabon,4,2,268,1,10,5,0,3,3,0,1,1,1,0,0,0,green,0,0,0,0,0,0,0,0,0,green,blue

Gambia,4,4,10,1,1,5,0,5,4,1,1,1,0,1,0,red,0,0,0,0,0,0,0,0,0,0,0,red,green  
 Germany-DDR,3,1,108,17,4,6,0,3,3,1,0,0,1,0,1,0,gold,0,0,0,0,0,0,1,0,0,black,gold  
 Germany-FRG,3,1,249,61,4,1,0,3,3,1,0,0,1,0,1,0,black,0,0,0,0,0,0,0,0,0,black,gold  
 Ghana,4,4,239,14,1,5,0,3,4,1,1,0,1,0,1,red,0,0,0,0,1,0,0,0,0,0,0,red,green  
 Gibraltar,3,4,0,0,1,1,0,1,3,1,0,0,1,1,0,0,white,0,0,0,0,0,0,0,1,0,0,white,red  
 Greece,3,1,132,10,6,1,0,9,2,0,0,1,0,1,0,blue,0,1,0,1,0,0,0,0,0,0,blue,blue  
 Greenland,1,4,2176,0,6,1,0,0,2,1,0,0,0,1,0,white,1,0,0,0,0,0,0,0,0,0,white,red  
 Grenada,1,4,0,0,1,1,0,0,3,1,1,0,1,0,0,gold,1,0,0,0,7,0,1,0,1,0,red,red  
 Guam,6,1,0,0,1,1,0,0,7,1,1,1,1,1,0,1,blue,0,0,0,0,0,0,0,0,1,1,1,red,red  
 Guatemala,1,4,109,8,2,0,3,0,2,0,0,1,0,1,0,0,blue,0,0,0,0,0,0,0,0,0,0,blue,blue  
 Guinea,4,4,246,6,3,2,3,0,3,1,1,0,1,0,0,0,gold,0,0,0,0,0,0,0,0,0,0,red,green  
 Guinea-Bissau,4,4,36,1,6,5,1,2,4,1,1,0,1,0,1,0,gold,0,0,0,0,1,0,0,0,0,0,red,green  
 Guyana,2,4,215,1,1,4,0,0,5,1,1,1,0,1,1,0,green,0,0,0,0,0,0,1,0,0,0,black,green  
 Haiti,1,4,28,6,3,0,2,0,2,1,0,0,0,0,1,0,black,0,0,0,0,0,0,0,0,0,0,black,red  
 Honduras,1,4,112,4,2,0,0,3,2,0,0,1,0,1,0,blue,0,0,0,0,5,0,0,0,0,0,blue,blue  
 Hong-Kong,5,1,1,5,7,3,0,0,6,1,1,1,1,1,0,1,blue,1,1,1,1,0,0,0,1,1,1,white,blue  
 Hungary,3,1,93,11,9,6,0,3,3,1,1,0,0,1,0,0,red,0,0,0,0,0,0,0,0,0,0,red,green  
 Iceland,3,4,103,0,6,1,0,0,3,1,0,1,0,1,0,0,blue,0,1,0,0,0,0,0,0,0,blue,blue  
 India,5,1,3268,684,6,4,0,3,4,0,1,1,0,1,0,1,orange,1,0,0,0,0,0,0,1,0,0,orange,green  
 Indonesia,6,2,1904,157,10,2,0,2,2,1,0,0,0,1,0,0,red,0,0,0,0,0,0,0,0,0,0,red,white  
 Iran,5,1,1648,39,6,2,0,3,3,1,1,0,0,1,0,0,red,0,0,0,0,0,0,1,0,1,green,red  
 Iraq,5,1,435,14,8,2,0,3,4,1,1,0,0,1,1,0,red,0,0,0,0,3,0,0,0,0,0,red,black  
 Ireland,3,4,70,3,1,0,3,0,3,0,1,0,1,0,1,white,0,0,0,0,0,0,0,0,0,0,green,orange  
 Israel,5,1,21,4,10,7,0,2,2,0,0,1,0,1,0,0,white,0,0,0,0,1,0,0,0,0,0,blue,blue  
 Italy,3,1,301,57,6,0,3,0,3,1,1,0,0,1,0,0,white,0,0,0,0,0,0,0,0,0,0,green,red  
 Ivory-Coast,4,4,323,7,3,5,3,0,3,1,1,0,0,1,0,0,white,0,0,0,0,0,0,0,0,0,0,red,green  
 Jamaica,1,4,11,2,1,1,0,0,3,0,1,0,1,0,1,0,green,0,0,1,0,0,0,1,0,0,0,0,gold,gold  
 Japan,5,1,372,118,9,7,0,0,2,1,0,0,0,1,0,0,white,1,0,0,0,1,0,0,0,0,0,white,white  
 Jordan,5,1,98,2,8,2,0,3,4,1,1,0,0,1,1,0,black,0,0,0,0,1,0,1,0,0,0,black,green  
 Kampuchea,5,1,181,6,10,3,0,0,2,1,0,0,1,0,0,red,0,0,0,0,0,0,0,1,0,0,red,red  
 Kenya,4,1,583,17,10,5,0,5,4,1,1,0,0,1,1,0,red,1,0,0,0,0,0,0,1,0,0,black,green  
 Kiribati,6,1,0,0,1,1,0,0,4,1,0,1,1,0,red,0,0,0,0,1,0,0,1,1,0,red,blue  
 Kuwait,5,1,18,2,8,2,0,3,4,1,1,0,0,1,1,0,green,0,0,0,0,0,0,0,0,0,0,green,red  
 Laos,5,1,236,3,10,6,0,3,3,1,0,1,0,1,0,0,red,1,0,0,0,0,0,0,0,0,0,red,red  
 Lebanon,5,1,10,3,8,2,0,2,4,1,1,0,0,1,0,1,red,0,0,0,0,0,0,0,0,1,0,red,red  
 Lesotho,4,2,30,1,10,5,2,0,4,1,1,1,0,1,0,blue,0,0,0,0,0,0,0,1,0,0,green,blue  
 Liberia,4,4,111,1,10,5,0,11,3,1,0,1,0,1,0,0,red,0,0,0,1,1,0,0,0,0,blue,red  
 Libya,4,1,1760,3,8,2,0,0,1,0,1,0,0,0,0,0,green,0,0,0,0,0,0,0,0,0,green,green  
 Liechtenstein,3,1,0,0,4,0,0,2,3,1,0,1,1,0,0,0,red,0,0,0,0,0,0,0,1,0,0,blue,red  
 Luxembourg,3,1,3,0,4,0,0,3,3,1,0,1,0,1,0,red,0,0,0,0,0,0,0,0,0,0,red,blue  
 Malagasy,4,2,587,9,10,1,1,2,3,1,1,0,0,1,0,0,red,0,0,0,0,0,0,0,0,0,white,green  
 Malawi,4,2,118,6,10,5,0,3,3,1,1,0,0,0,1,0,red,0,0,0,0,1,0,0,0,0,black,green  
 Malaysia,5,1,333,13,10,2,0,14,4,1,0,1,1,1,0,0,red,0,0,0,1,1,1,0,0,0,blue,white  
 Maldives,5,1,0,0,10,2,0,0,3,1,1,0,0,1,0,0,red,0,0,0,0,0,1,0,0,0,0,red,red  
 Mali,4,4,1240,7,3,2,3,0,3,1,1,0,1,0,0,0,gold,0,0,0,0,0,0,0,0,0,green,red  
 Malta,3,1,0,0,10,0,2,0,3,1,0,0,0,1,1,0,red,0,1,0,0,0,0,0,1,0,0,white,red  
 Marianas,6,1,0,0,10,1,0,0,3,0,0,1,0,1,0,0,blue,0,0,0,0,1,0,0,1,0,0,blue,blue  
 Mauritania,4,4,1031,2,8,2,0,0,2,0,1,0,1,0,0,0,green,0,0,0,0,1,1,0,0,0,0,green,green  
 Mauritius,4,2,2,1,1,4,0,4,4,1,1,1,1,0,0,red,0,0,0,0,0,0,0,0,0,0,red,green  
 Mexico,1,4,1973,77,2,0,3,0,4,1,1,0,0,1,0,1,green,0,0,0,0,0,0,0,1,0,green,red  
 Micronesia,6,1,1,0,10,1,0,0,2,0,0,1,0,1,0,0,blue,0,0,0,0,4,0,0,0,0,0,blue,blue  
 Monaco,3,1,0,0,3,0,0,2,2,1,0,0,0,1,0,0,red,0,0,0,0,0,0,0,0,0,0,red,white  
 Mongolia,5,1,1566,2,10,6,3,0,3,1,0,1,1,0,0,0,red,2,0,0,0,1,1,1,0,0,red,red  
 Montserrat,1,4,0,0,1,1,0,0,7,1,1,1,1,1,0,blue,0,2,1,1,0,0,0,1,1,0,white,blue  
 Morocco,4,4,447,20,8,2,0,0,2,1,1,0,0,0,0,0,red,0,0,0,0,1,0,0,0,0,0,red,red  
 Mozambique,4,2,783,12,10,5,0,5,5,1,1,0,1,1,1,0,gold,0,0,0,0,1,0,1,1,0,0,green,gold  
 Nauru,6,2,0,0,10,1,0,3,3,0,0,1,1,1,0,0,blue,0,0,0,0,1,0,0,0,0,0,blue,blue  
 Nepal,5,1,140,16,10,4,0,0,3,0,0,1,0,1,0,1,brown,0,0,0,0,2,1,0,0,0,0,blue,blue  
 Netherlands,3,1,41,14,6,1,0,3,3,1,0,1,0,1,0,0,red,0,0,0,0,0,0,0,0,0,0,red,blue  
 Netherlands-Antilles,1,4,0,0,6,1,0,1,3,1,0,1,0,1,0,0,white,0,0,0,0,6,0,0,0,0,0,white,white  
 New-Zealand,6,2,268,2,1,1,0,0,3,1,0,1,0,1,0,blue,0,1,1,1,4,0,0,0,0,0,white,blue  
 Nicaragua,1,4,128,3,2,0,0,3,2,0,0,1,0,1,0,0,blue,0,0,0,0,0,0,0,0,0,blue,blue  
 Niger,4,1,1267,5,3,2,0,3,3,0,1,0,0,1,0,1,orange,1,0,0,0,0,0,0,0,0,0,orange,green  
 Nigeria,4,1,925,56,10,2,3,0,2,0,1,0,0,1,0,0,green,0,0,0,0,0,0,0,0,0,0,green,green  
 Niue,6,3,0,0,1,1,0,0,4,1,0,1,1,1,0,0,gold,1,1,1,1,5,0,0,0,0,0,white,gold  
 North-Korea,5,1,121,18,10,6,0,5,3,1,0,1,0,1,0,0,blue,1,0,0,0,1,0,0,0,0,0,blue,blue  
 North-Yemen,5,1,195,9,8,2,0,3,4,1,1,0,0,1,1,0,red,0,0,0,0,1,0,0,0,0,0,red,black

Norway,3,1,324,4,6,1,0,0,3,1,0,1,0,1,0,red,0,1,0,0,0,0,0,0,0,red,red  
 Oman,5,1,212,1,8,2,0,2,3,1,1,0,0,1,0,red,0,0,0,0,0,0,1,0,0,red,green  
 Pakistan,5,1,804,84,6,2,1,0,2,0,1,0,0,1,0,green,0,0,0,0,1,1,0,0,0,white,green  
 Panama,2,4,76,2,2,0,0,0,3,1,0,1,0,1,0,red,0,0,0,4,2,0,0,0,0,white,white  
 Papua-New-Guinea,6,2,463,3,1,5,0,0,4,1,0,0,1,1,1,0,black,0,0,0,5,0,1,0,1,0,red,black  
 Paraguay,2,3,407,3,2,0,0,3,6,1,1,1,1,1,0,red,1,0,0,0,1,0,0,1,1,1,red,blue  
 Peru,2,3,1285,14,2,0,3,0,2,1,0,0,0,1,0,red,0,0,0,0,0,0,0,0,0,0,red,red  
 Philippines,6,1,300,48,10,0,0,0,4,1,0,1,1,1,0,blue,0,0,0,0,4,0,1,0,0,0,blue,red  
 Poland,3,1,313,36,5,6,0,2,2,1,0,0,0,1,0,white,0,0,0,0,0,0,0,0,0,white,red  
 Portugal,3,4,92,10,6,0,0,0,5,1,1,1,1,0,0,red,1,0,0,0,0,0,0,1,0,0,green,red  
 Puerto-Rico,1,4,9,3,2,0,0,5,3,1,0,1,0,1,0,red,0,0,0,0,1,0,1,0,0,0,red,red  
 Qatar,5,1,11,0,8,2,0,0,2,0,0,0,0,1,0,1,brown,0,0,0,0,0,0,0,0,0,white,brown  
 Romania,3,1,237,22,6,6,3,0,7,1,1,1,1,0,1,red,0,0,0,0,2,0,0,1,1,1,blue,red  
 Rwanda,4,2,26,5,10,5,3,0,4,1,1,0,1,0,1,0,red,0,0,0,0,0,0,0,0,0,1,red,green  
 San-Marino,3,1,0,0,6,0,0,2,2,0,0,1,0,1,0,0,white,0,0,0,0,0,0,0,0,0,white,blue  
 Sao-Tome,4,1,0,0,6,0,0,3,4,1,1,0,1,0,1,0,green,0,0,0,0,2,0,1,0,0,0,green,green  
 Saudi-Arabia,5,1,2150,9,8,2,0,0,2,0,1,0,0,1,0,0,green,0,0,0,0,0,0,0,1,0,1,green,green  
 Senegal,4,4,196,6,3,2,3,0,3,1,1,0,1,0,0,0,green,0,0,0,0,1,0,0,0,0,0,green,red  
 Seychelles,4,2,0,0,1,1,0,0,3,1,1,0,0,1,0,0,red,0,0,0,0,0,0,0,0,0,red,green  
 Sierra-Leone,4,4,72,3,1,5,0,3,3,0,1,1,0,1,0,0,green,0,0,0,0,0,0,0,0,0,green,blue  
 Singapore,5,1,1,3,7,3,0,2,2,1,0,0,0,1,0,0,white,0,0,0,0,5,1,0,0,0,0,red,white  
 Soloman-Islands,6,2,30,0,1,1,0,0,4,0,1,1,1,0,0,green,0,0,0,0,5,0,1,0,0,0,blue,green  
 Somalia,4,1,637,5,10,2,0,0,2,0,0,1,0,1,0,blue,0,0,0,0,1,0,0,0,0,0,blue,blue  
 South-Africa,4,2,1221,29,6,1,0,3,5,1,1,1,0,1,0,1,orange,0,1,1,0,0,0,0,0,0,orange,blue  
 South-Korea,5,1,99,39,10,7,0,0,4,1,0,1,0,1,0,white,1,0,0,0,0,0,1,0,0,white,white  
 South-Yemen,5,1,288,2,8,2,0,3,4,1,0,1,0,1,0,red,0,0,0,0,1,0,1,0,0,0,red,black  
 Spain,3,4,505,38,2,0,0,3,2,1,0,0,1,0,0,red,0,0,0,0,0,0,0,0,0,0,red,red  
 Sri-Lanka,5,1,66,15,10,3,2,0,4,0,1,0,0,1,gold,0,0,0,0,0,0,0,1,1,0,gold,gold  
 St-Helena,4,3,0,0,1,1,0,0,7,1,1,1,1,0,1,blue,0,1,1,1,0,0,0,1,0,0,white,blue  
 St-Kitts-Nevis,1,4,0,0,1,1,0,0,5,1,1,0,1,1,0,green,0,0,0,0,2,0,1,0,0,0,green,red  
 St-Lucia,1,4,0,0,1,1,0,0,4,0,0,1,1,1,0,blue,0,0,0,0,0,0,1,0,0,0,blue,blue  
 St-Vincent,1,4,0,0,1,1,5,0,4,0,1,1,1,0,0,green,0,0,0,0,0,0,0,1,1,1,blue,green  
 Sudan,4,1,2506,20,8,2,0,3,4,1,1,0,0,1,1,0,red,0,0,0,0,0,0,1,0,0,0,red,black  
 Surinam,2,4,63,0,6,1,0,5,4,1,1,0,1,1,0,0,red,0,0,0,0,1,0,0,0,0,0,green,green  
 Swaziland,4,2,17,1,10,1,0,5,7,1,0,1,1,1,1,blue,0,0,0,0,0,0,0,1,0,0,blue,blue  
 Sweden,3,1,450,8,6,1,0,0,2,0,0,1,1,0,0,blue,0,1,0,0,0,0,0,0,0,blue,blue  
 Switzerland,3,1,41,6,4,1,0,0,2,1,0,0,0,1,0,red,0,1,0,0,0,0,0,0,0,red,red  
 Syria,5,1,185,10,8,2,0,3,4,1,1,0,0,1,1,0,red,0,0,0,0,2,0,0,0,0,0,red,black  
 Taiwan,5,1,36,18,7,3,0,0,3,1,0,1,0,1,0,red,1,0,0,1,1,0,0,0,0,blue,red  
 Tanzania,4,2,945,18,10,5,0,0,4,0,1,1,1,0,1,0,green,0,0,0,0,0,0,1,0,0,0,green,blue  
 Thailand,5,1,514,49,10,3,0,5,3,1,0,1,0,1,0,red,0,0,0,0,0,0,0,0,0,red,red  
 Togo,4,1,57,2,3,7,0,5,4,1,1,0,1,1,0,0,green,0,0,0,1,1,0,0,0,0,0,red,green  
 Tonga,6,2,1,0,10,1,0,0,2,1,0,0,0,1,0,0,red,0,1,0,1,0,0,0,0,0,white,red  
 Trinidad-Tobago,2,4,5,1,1,1,0,0,3,1,0,0,0,1,1,0,red,0,0,0,0,0,0,0,1,0,0,white,white  
 Tunisia,4,1,164,7,8,2,0,0,2,1,0,0,0,1,0,red,1,0,0,0,1,1,0,0,0,red,red  
 Turkey,5,1,781,45,9,2,0,0,2,1,0,0,0,1,0,0,red,0,0,0,0,1,1,0,0,0,red,red  
 Turks-Cocos-Islands,1,4,0,0,1,1,0,0,6,1,1,1,1,0,1,blue,0,1,1,1,0,0,0,1,1,0,white,blue  
 Tuvalu,6,2,0,0,1,1,0,0,5,1,0,1,1,0,blue,0,1,1,9,0,0,0,0,0,0,white,blue  
 UAE,5,1,84,1,8,2,1,3,4,1,1,0,0,1,1,0,green,0,0,0,0,0,0,0,0,0,0,red,black  
 Uganda,4,1,236,13,10,5,0,6,5,1,0,0,1,1,1,0,gold,1,0,0,0,0,0,0,1,0,black,red  
 UK,3,4,245,56,1,1,0,0,3,1,0,1,0,1,0,0,red,0,1,1,0,0,0,0,0,0,0,white,red  
 Uruguay,2,3,178,3,2,0,0,9,3,0,0,1,1,1,0,0,white,0,0,0,1,1,0,0,0,0,0,white,white  
 US-Virgin-Isles,1,4,0,0,1,1,0,0,6,1,1,1,1,0,0,white,0,0,0,0,0,0,0,1,1,1,white,white  
 USA,1,4,9363,231,1,1,0,13,3,1,0,1,0,1,0,0,white,0,0,0,1,50,0,0,0,0,blue,red  
 USSR,5,1,22402,274,5,6,0,0,2,1,0,0,1,0,0,red,0,0,0,0,1,0,0,1,0,0,red,red  
 Vanuatu,6,2,15,0,6,1,0,0,4,1,1,0,1,0,red,0,0,0,0,0,0,1,0,1,0,black,green  
 Vatican-City,3,1,0,0,6,0,2,0,4,1,0,0,1,1,1,0,gold,0,0,0,0,0,0,0,1,0,0,gold,white  
 Venezuela,2,4,912,15,2,0,0,3,7,1,1,1,1,1,1,1,red,0,0,0,0,7,0,0,1,1,0,gold,red  
 Vietnam,5,1,333,60,10,6,0,0,2,1,0,0,1,0,0,red,0,0,0,0,1,0,0,0,0,0,red,red  
 Western-Samoan,6,3,3,0,1,1,0,0,3,1,0,1,0,1,0,red,0,0,0,1,5,0,0,0,0,0,blue,red  
 Yugoslavia,3,1,256,22,6,6,0,3,4,1,0,1,1,1,0,red,0,0,0,0,1,0,0,0,0,0,blue,red  
 Zaire,4,2,905,28,10,5,0,0,4,1,1,0,1,0,0,1,green,1,0,0,0,0,0,0,1,1,0,green,green  
 Zambia,4,2,753,6,10,5,3,0,4,1,1,0,0,0,1,1,green,0,0,0,0,0,0,0,1,0,green,brown  
 Zimbabwe,4,2,391,8,10,5,0,7,5,1,1,0,1,1,0,green,0,0,0,0,1,0,1,1,0,green,green

## ANEXO C – Imagens das bandeiras



