Universidade de São Paulo
Instituto de Matemática e Estatística
Departamento de Matemática Aplicada

Classificação binária com Segmentos de Retas baseada em Redes Neurais

Carine Naomi Imasato
carine.imasato@usp.br
Prof. Dr. Ronaldo Fumio Hashimoto
ronaldo@ime.usp.br

Resumo

Um dos campos em rápida ascensão nos tempos atuais é o campo da Aprendizagem de Máquina. Suas técnicas têm semostrado cada vez mais eficientes, e com o auxílio da tecnologia atual, este campo tem se aprimorado continuamente.

O problema explorado neste trabalho é o da classificação binária, em que a técnica proposta para resolvê-lo é uma combinação de Redes Neurais (estruturadas popularmente utilizadas na Aprendizagem de Máquina) com Classificadores Binários que usam Segmentos de Retas (Classificadores SLS).

Este trabalho procura desenvolver e explorar esta nova técnica, detalhando alguns procedimentos e explicando algumas decisões tomadas no decorrer de seu desenvolvimento, assim como testá-la e exibir resultados obtidos com os modelos resultantes deste algoritmo. E finalmente pontuar algumas alterações à implementação que possivelmente pode melhorar a performance do mesmo.

Resumo

Machine Learning is undoubtedly within the fields with the fastest growths in recent times. Its techniques and tools have been proving their worth in terms of efficiency which, paired with the latest achievements in technology, has proved that this field is and will continue to ascend steadily.

In this thesis, it is desired to solve a Binary Classification problem. In order to fulfill this objective, the proposed technique combines Artificial Neural Networks (which are already a popular choice in the Machine Learning realm) and Straight Line Segments Classifiers (SLS Classifiers).

The main objective of this thesis is to develop and exploit the proposed technique, providing with the details to certain procedures and the reasons for certain decisions taken along the development of this algorithm, as well as testing the models obtained through this method and displaying the results given a set of training data and another set of testing data. And finally, a few modifications that could possibly improve said algorithm's performance will be pointed out.

Dedicatória

Dedico este trabalho aos meus professores, em especial o Prof. Dr. Ronaldo F. Hashimoto, que me orientou e apoiou durante o seu desenvolvimento. Dedico-o também a familiares e amigos, que me sempre me apoiaram na jornada acadêmica.

Lista de Tabelas

Lista de Figuras

Conteúdo

1 Introdução

Recentemente, o interesse pela a área de Aprendizagem de Máquina tem aumentado devido a grande disponibilidade de dados (*big data*). Em particular, técnicas para o aprendizado supervisionado (classificação binária) são bastante estudadas devido à grande variedade de aplicações tais como: diagnóstico de tumor benigno ou maligno a partir de imagens médicas, recomendar ou não um determinado produto a um cliente a partir de seu perfil, classificar se uma determinada peça em uma linha produção é ideal para uso ou não, entre outros. Além disso, a área tem sido alvo de muitos incentivos nos últimos anos, pois o *hardware* atual finalmente é capaz de suportar a carga computacional exigida por essas técnicas; haja vista as grandes empresas produtoras de processadores têm promovido o uso de suas peças especificamente para este tipo de atividade.

Há vários métodos e diferentes técnicas usadas para classificar um conjunto de dados. Uma das formas mais intuitivas para classificar um objeto é encontrando a menor distância entre este e um conjunto de objetos já classificados. No caso particular deste trabalho, lidaremos com a classificação binária, utilizando-se de segmentos de reta para representar as características dos grupos já classificados. Essa técnica é chamada de *aprendizado por segmentos de reta*, ou simplesmente *SLS* (do Inglês, *Straight Line Segment*) [7, 8, 9, 10, 11, 12].

Neste trabalho, iremos propor uma extensão desse classificador implementando-o em uma Rede Neural de uma camada oculta com n nós de forma que cada nó da rede neural é um classificador SLS. Intuitivamente, a estrutura de uma Rede Neural seria capaz de otimizar o problema de classificação binária, neste trabalho será desenvolvida a proposta de uma nova técnica, seus resultados e pontos que podem ser aprimorados.

A implementação do modelo foi feito inteiramente na linguagem R, com o auxílio de pacotes como *stat* e *dplyr*, usados para tratar com os dados envolvidos nas operações e procedimentos do programa.

2 Motivação

Num problema de classificação binária em que os dados envolvidos são tais que $(\mathbf{x}, y) \in \mathbb{R}^d \times \{0, 1\}$, com d sendo um número natural "suficientemente grande", um classificador SLS deve otimizar pelo menos $2 \cdot 2 \cdot d \cdot N$ parâmetros (N é a quantidade de segmentos de reta presentes em cada classe), visto que há duas classes, cada qual com N segmentos de reta, os quais são definidos por duas extremidades $\mathbf{p}, \mathbf{q} \in \mathbb{R}^{d+1}$.

Considerando os valores citados acima, é possível assumir que este problema de otimização tem complexidade muito superior a um problema cujos dados estão em $\mathbb{R}^2$ ou $\mathbb{R}^3$, por exemplo. Tendo isso em mente, foi então concebida a ideia de organizar uma série de classificadores SLS cujos dados não têm dimensão superior a 3 (ou seja, pertencem a $\mathbb{R}^2$ ou $\mathbb{R}^3$) em cadeia, de forma semelhante a uma Rede Neural, estrutura com a qual é muito comum resolver problemas de otimização/aprendizado.

3 Objetivo

Como observado acima, o método proposto tem potencial para apresentar alguma melhora sobre o algoritmo de classificação binária por segmentos de reta tradicional.

Este projeto tem como objetivo principal o desenvolvimento e a implementação de um algoritmo que, dado um conjunto de amostras, seja capaz de encontrar a melhor Rede Neural em que cada nó é um Classificador SLS de acordo com o critério da minimização do erro quadrático.

A partir dos modelos obtidos pelo algoritmo proposto acima, deseja-se encontrar os limites de sua eficiência e precisão, com o auxílio de alguns conjuntos de treinamento e teste gerados a partir de funções conhecidas.

Se o algoritmo implementado não se mostrar suficientemente eficaz ou preciso como o esperado, será apresentada uma lista pontuando algumas modificações que podem ser feitas no código para melhorar o seu desempenho. Modificações estas que não foram implementadas pelo tempo limitado imposto para o desenvolvimento deste trabalho, mas que foram observadas durante o processo de implementação e portanto merecem um destaque na conclusão deste trabalho.

Com este trabalho espero conseguir ao menos intrigar aqueles que têm interesse em experimentar novos formatos e combinações de técnicas para Aprendizagem de Máquina, abrindo assim novas portas para possibilidades ainda não exploradas.

4 O Método

Na primeira parte deste capítulo deseja-se apresentar de forma genérica e superficial o funcionamento de algumas estruturas e algoritmos que são utilizados ou mencionados no decorrer deste trabalho, assim como contextualizar o leitor de forma suficiente para prosseguir com leitura deste trabalho.

A segunda metade é dedicada à apresentar o novo algoritmo, proposto para efetuar a classificação binária de objetos. Também serão apresentados maiores detalhes quanto aos procedimentos e escolhas tomadas ao longo de desenvolvimento deste trabalho.

4.1 Conceitos preliminares

4.1.1 O algoritmo do *k*-médias

O *k-médias* (*k-means*) é um algoritmo que particiona um conjunto de dados de forma a minimizar um critério de agrupamento (a soma de distâncias euclidianas é popularmente utilizada como este critério). Suponha que dado um conjunto $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ em que todo $\mathbf{x}_i \in \mathbb{R}^d$, deseja-se particioná-lo em M agrupamentos (*clusters*) $C_1, ..., C_M$, de forma que a soma do quadrado das distâncias Euclidianas entre cada ponto $\mathbf{x}_i$ e o centroide $\mathbf{m}_k$ da classe C_k que ele pertença sejam minimizadas [1, 2].

$$E(\mathbf{m}_1, ..., \mathbf{m}_M) = \sum_{i=1}^{N} \sum_{k=1}^{M} I(\mathbf{x}_i \in C_k)|\mathbf{x}_i - \mathbf{m}_k|^2 \tag{1}$$

em que,

$$I(b) = \begin{cases} 1, & \text{se } b \text{ é verdadeiro} \\ 0, & \text{caso contrário} \end{cases}$$

O método visa encontrar os centroides $\mathbf{m}_1, ..., \mathbf{m}_M$ que minimizam o erro denotado acima como $E(\mathbf{m}_1, ..., \mathbf{m}_M)$.

O algoritmo é inicializado tomando-se k pontos aleatórios, estes servirão de protótipos dos centroides dos k agrupamentos desejados. Com base nestes centroides, os dados do conjunto X são rotulados de acordo com o centroide mais próximo (segundo a função da Equação 1), formando assim as k partições dos dados. Uma vez que X foi particionado e os protótipos dos k agrupamentos foram formados, os centroides são então atualizados, tomando-se como novo valor a média dos pontos que compõem o agrupamento em questão. E novamente os pontos são rotulados, mas agora de acordo com os novos centroides. Este processo é então repetido até que, de um passo a outro, a configuração dos agrupamentos não tenha mudado, ou seja, supondo que no j-ésimo passo os k agrupamentos são os mesmos obtidos pelo $(j+1)$-ésimo passo, o algoritmo é interrompido e os centroides obtidos pela iteração mais recente definem o particionamento dos dados.

Por este algoritmo não ser o centro deste trabalho, optamos por utilizar a função *k-means* do pacote *stat* da linguagem R no lugar de implementá-la por completo.

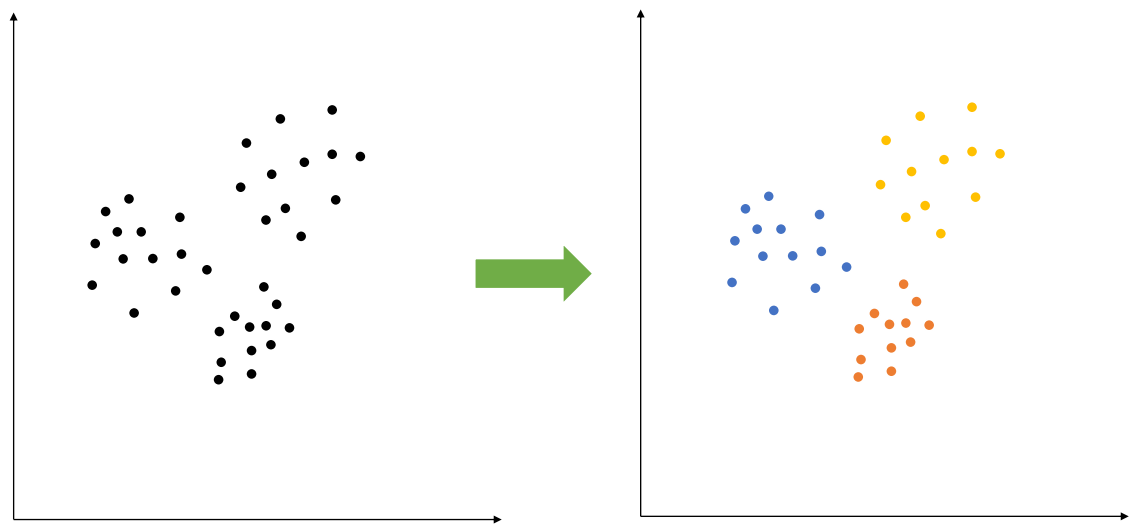

Figura 1: Exemplo de como o 3-médias pode atuar no conjunto de pontos ilustrados

4.1.2 Otimização com o Gradiente Descendente

O Gradiente Descendente (*Gradient Descent*), ou como o chamaremos neste trabalho, GD, é muito utilizado em diversos problemas de otimização cuja função objetivo é contínua e diferenciável. Trata-se também do algoritmo mais utilizado no campo de Aprendizagem de Máquina [3, 5].

A função objetivo do problema deve ser necessariamente contínua e diferenciável pois o algoritmo consiste no uso do gradiente desta como orientação para os ajustes dos parâmetros da função de forma iterativa, e dessa forma atingir um mínimo local, como ilustrado na Figura 2. Com o auxílio do gradiente, é possível mensurar o quanto um ajuste influencia o valor final, e logo, o quanto esse ajuste influencia no retorno da função objetivo.

Dada a direção (pelo gradiente da função objetivo), é necessário então determinar o tamanho do passo a ser dado nesta direção, a taxa de aprendizado. Esta taxa dita o quão rápido ou lento o algoritmo avança ao longo de suas iterações.

Considere, por exemplo, uma função contínua $E(p_1, ..., p_n)$ como a função objetivo. A taxa de aprendizado será denotada por α e o posicionamento inicial dos parâmetros é $E(\hat{p_1}, ..., \hat{p_n})$, deseja-se ajustá-los de forma a levar a função objetivo ao mínimo. O gradiente desta função é definido por:

$$\nabla E(p_1, ..., p_n) = \begin{bmatrix} \frac{\partial E}{\partial p_1} \\ ... \\ \frac{\partial E}{\partial p_n}. \end{bmatrix}$$

Até que a função objetivo (erro) atinja um valor mínimo desejado ou um número máximo de iterações seja atingido, os parâmetros são ajustados da seguinte forma:

$$p_k = \hat{p_k} - \alpha \frac{\partial E}{\partial p_k}$$

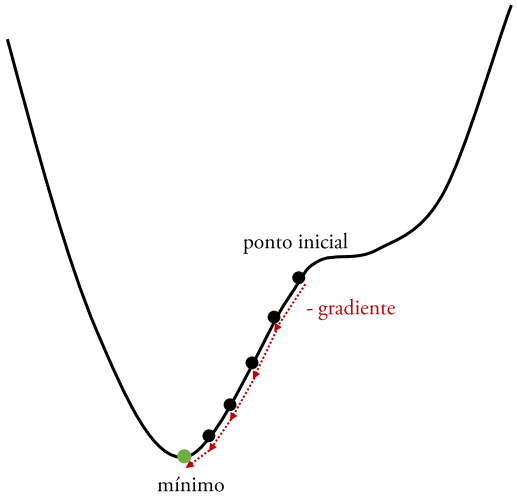

Figura 2: Ilustração simples do funcionamento do algoritmo do GD.

4.2 Classificação Binária com Segmentos de Reta

Uma das formas mais tradicionais e intuitivas de classificação de dados é por meio da avaliação da distância de um dado (i.e. um ponto) e as classes dentre as quais desejamos classificá-lo. Realisticamente, é inviável mensurar a distância entre um ponto todos os dados já classificados para concluir a qual classe este pertence; disso surgiu a necessidade de encontrar bons *"representantes"* para conjuntos de dados, principalmente quando são extremamente volumosos. Estudos exploram as várias formas que estes *"representantes"* podem assumir e como isso pode influenciar na precisão de um classificador.

A Classificação Binária usando Segmentos de Reta (*Straight Line Segment Binary Classification*), ou classificação SLS, é um método que consiste em classificar dados entre duas classes (neste trabalho as chamaremos de *classe 0* e *classe 1*) usando N segmentos de reta para *"representar"* cada uma dessas classes. Cada segmento é definido pelo par de coordenadas de suas extremidades (os quais chamaremos de $\mathbf{p} \in \mathbb{R}^{d+1}$ e $\mathbf{q} \in \mathbb{R}^{d+1}$). Este método tem como objetivo encontrar a configuração dos N segmentos de reta de cada classe que minimiza o erro de classificação.

O posicionamento inicial dos segmentos de reta para uma classe c parte de um conjunto de dados já classificados como c. Com o auxílio do *k-médias* (em que $k = N$), são encontrados N agrupamentos, dos quais são tirados os segmentos de reta ao aplicar o *2-médias*. Em outras palavras, no primeiro momento é aplicado o *N-médias* para que o posicionamento geral de cada segmento seja obtido; desses agrupamentos são então tiradas as extremidades $\mathbf{p}$ e $\mathbf{q}$ por meio do *2-médias*.

Dados os dois pontos $\mathbf{p}, \mathbf{q} \in \mathbb{R}^{d+1}$, um *segmento de reta* $\overline{\mathbf{pq}} \subset \mathbb{R}^{d+1}$ é definido como:

$$\overline{\mathbf{pq}} = \{\mathbf{x} \in \mathbb{R}^{d+1} : \mathbf{x} = \lambda \cdot \mathbf{p} + (1 - \lambda) \cdot \mathbf{q}, \quad \lambda \in [0, 1]\}.$$

A *distância* (Euclidiana) entre dois pontos $\mathbf{p}, \mathbf{q} \in \mathbb{R}^{d+1}$ é definida como:

$$dist(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{d+1} (p_i - q_i)^2}.$$

Note que na Figura 3, a distância que é considerada para a análise (e para a classificação) é denotada por $pdist(x, segmento)$. Esta *distância* de um ponto $\mathbf{x} \in \mathbb{R}^d$ a um segmento de reta $\overline{\mathbf{pq}} \subset \mathbb{R}^{d+1}$ é definida como:

$$pdist(\mathbf{x}, \overline{\mathbf{pq}}) = \frac{dist(\mathbf{x}_e, \mathbf{p}) + dist(\mathbf{x}_e, \mathbf{q}) - dist(\mathbf{p}, \mathbf{q})}{2}, \quad \mathbf{x}_e = (\mathbf{x}, 0).$$

Dados dois conjuntos de segmentos de reta $\mathcal{L}_0$ e $\mathcal{L}_1$, rotulados como das classes 0 e 1, respectivamente, definimos a *função discriminante* $T_{\mathcal{L}_0,\mathcal{L}_1} : \mathbb{R}^d \to \mathbb{R}$ como:

$$T_{\mathcal{L}_0,\mathcal{L}_1}(\mathbf{x}) = \lim_{\epsilon \to 0} \left(\sum_{\overline{\mathbf{pq}} \in \mathcal{L}_1} pdist(\mathbf{x}, \overline{\mathbf{pq}}) - \sum_{\overline{\mathbf{pq}} \in \mathcal{L}_0} pdist(\mathbf{x}, \overline{\mathbf{pq}}) \right) \tag{2}$$

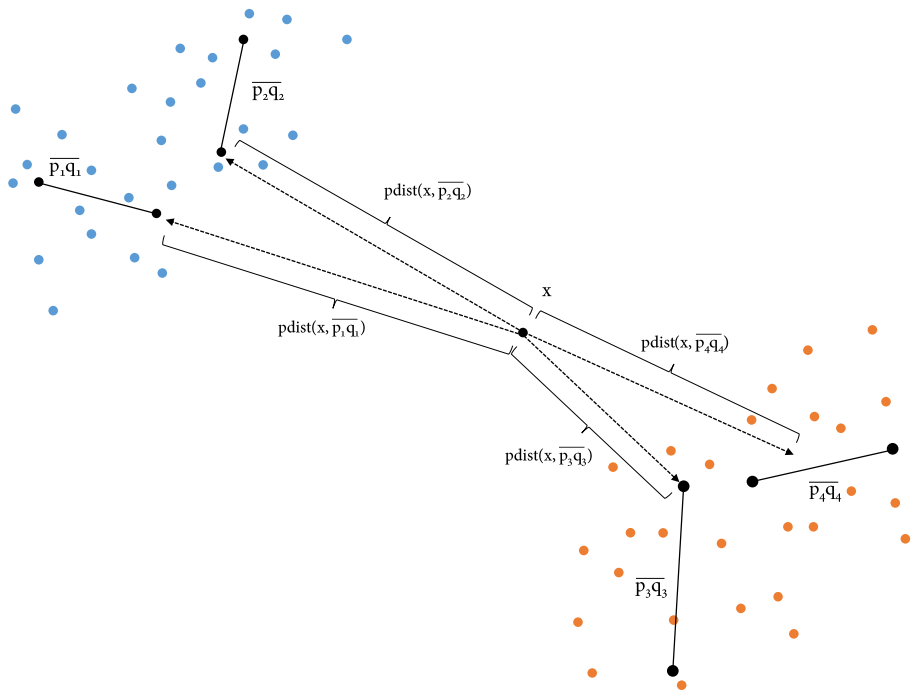

Figura 3: Exemplo de um classificador SLS. Note que neste caso cada classe possui dois segmentos de reta para representá-la.

A *função de ativação* $\phi : \mathbb{R} \rightarrow]0,1[$ é definida como:

$$\phi(z) = \frac{1}{1 + e^{-g \cdot z}}. \tag{3}$$

Um *classificador SLS* a função $f_{\mathcal{L}_0,\mathcal{L}_1} : R^d \rightarrow]0,1[$ definida simplesmente como a composição da função discriminante com a função de ativação, ou seja,

$$f_{\mathcal{L}_0,\mathcal{L}_1}(\mathbf{x}) = \phi(T_{\mathcal{L}_0,\mathcal{L}_1}(\mathbf{x})) \tag{4}$$

E finalmente, a *função de classificação* $y_{\mathcal{L}_0,\mathcal{L}_1} : \mathbb{R}^d \rightarrow \{0,1\}$ é definida como:

$$y_{\mathcal{L}_0,\mathcal{L}_1}(\mathbf{x}) = \begin{cases} 0 & \text{se } f_{\mathcal{L}_0,\mathcal{L}_1}(\mathbf{x}) < 0.5; \\ 1 & \text{caso contrário.} \end{cases} \tag{5}$$

4.3 Redes Neurais

Baseadas em estruturas neurais biológicas, as Redes Neurais (*Artificial Neural Networks*, em inglês) são estruturas popularmente utilizadas na Aprendizagem de Máquina, principalmente nos campos de reconhecimento de imagem e de fala [5].

Uma Rede Neural é composta de neurônios, que por sua vez são organizados em camadas. Cada camada pode conter um ou múltiplos neurônios, e sua posição na rede neural dita a qual classe ela pertence: camada de entrada, camada oculta ou camada de saída. A primeira e a última camada são classificadas como camada de entrada e camada de saída, respectivamente. Toda e qualquer camada entre as camadas de entrada e de saída é considerada como camada oculta.

Um exemplo simples de rede neural pode ser encontrado na Figura 4. Neste caso, a camada de entrada é composta de três neurônios, a camada de saída de um neurônio, e sua única camada oculta tem quatro neurônios.

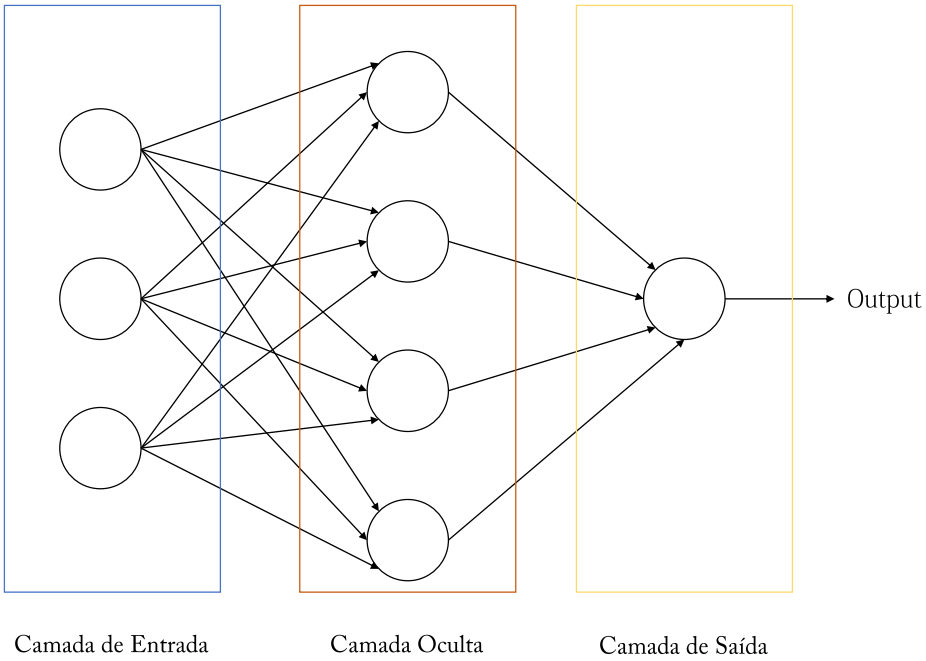

Figura 4: Exemplo de uma Rede Neural simples.

Um modelo básico de um neurônio é descrito pela aplicação de uma função de ativação Φ sobre uma combinação linear F dos dados de entrada, $x_1, x_2, ..., x_d$, com seus pesos correspondentes, $w_1, w_2, ..., w_d$, e uma constante *bias*, b. Obtendo assim a saída y tal que $y = \Phi(F)$, em que $F = b + \sum_{i=1}^{d} w_i x_i$. Esta saída é então passada junto com as demais saídas desta camada como dados de entrada para a próxima camada; esta cadeia é perpetuada até que a saída final é gerada pela camada de saída.

O modelo de neurônio mais conhecido é chamado de *neurônio sigmoide*, cuja função de ativação Φ é a *função sigmoide*, ou seja $\Phi(F) = \frac{1}{1+e^{-\eta F}}$, e portanto a saída y é calculada da seguinte forma:

$$y = \frac{1}{1 + e^{-\eta \sum_{i=1}^{d} w_i x_i - b}}$$

A popularidade do neurônio sigmoide se dá pela facilidade de se obter a derivada parcial desta função, uma vez que a técnica utilizada por muitos para o ajuste dos pesos e do *bias* é o GD. Estes ajustes são feitos pelos processos de *backpropagation* e *feed-forward*.

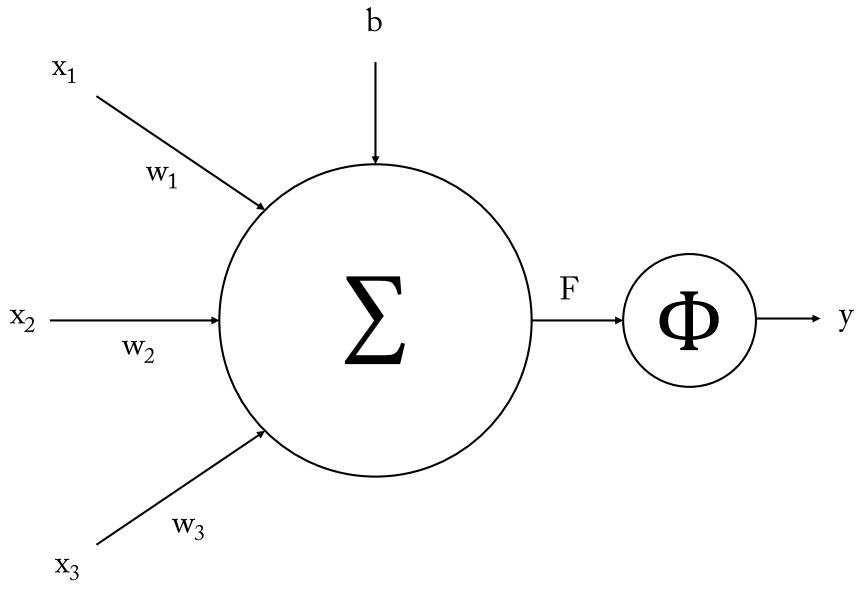

Figura 5: Exemplo de um modelo simples de um neurônio.

O termo *backpropagation* vem de *"backward propagation of errors"*, ou seja, o método propaga o erro obtido na saída do sistema para as etapas anteriores.

A fim de obter uma Rede Neural capaz de identificar caracteres escritos à mão, por exemplo, é de suma importância que um conjunto de dados de treinamento e um conjunto de dados de teste sejam providos; no exemplo dado, estes dados poderiam ser imagens de caracteres escritos à mão já rotulados com o caractere que elas representam. Os dados de treino são então utilizados para ajustar os parâmetros (os pesos e as constantes *bias*) do sistema, uma vez que o seu resultado esperado já é conhecido. Uma vez que o sistema está completamente ajustado, os dados de teste, que por sua vez são diferentes dos dados de treinamento, são alimentados ao sistema já ajustado e verifica se estes objetos foram devidamente rotulados [3, 4, 5, 6].

No momento do treinamento do sistema, é calculado o erro entre a saída obtida e o valor que o sistema idealmente geraria para aquela entrada em específico, o qual chamaremos de valor esperado. Essa função de erro também é chamada de função de custo, a qual indicaremos por $C(\vec{w}, \vec{b})$, e seus parâmetros são os pesos $\vec{w} = (w_1^1, w_2^1, ..., w_m^n)$ e as constantes

bias $\vec{b} = (b^1, b^2, ..., b^n)$

A função custo $C(\vec{w}, \vec{b})$ cumpre o papel de indicar o quão próximos ou distantes estão os parâmetros do ponto ideal (mínimo) para que as saídas do sistema atinjam os valores esperados. Os ajustes dos parâmetros são feitos com auxílio do GD, dessa forma:

$$\tilde{w}_j^i = w_j^i - \eta \frac{\partial C(\vec{w}, \vec{b})}{\partial w_j^i}$$

$$\tilde{b}^i = b^i - \eta \frac{\partial C(\vec{w}, \vec{b})}{\partial b^i}$$

em que w_j^i representa o valor atual do peso no j-ésimo peso no i-ésimo neurônio, b^i representa o *bias* no i-ésimo neurônio do sistema e η é a taxa de aprendizado utilizada no GD. A este processo de ajustamento dos parâmetros dá-se o nome de *backpropagation*. Note que estes ajustes podem potencialmente causar mudanças na saída de alguns neurônios, no entanto nesta etapa as atualizações são feitas todas sem considerar a atualização de valores de saída.

Assim que os parâmetros são ajustados, os valores de saída são então atualizados com o *feed-forward*, o qual é responsável por recalcular as saídas de todos os neurônios e finalmente obter valor da função $C(\vec{w}, \vec{b})$ com os parâmetros devidamente ajustados. Caso o erro obtido seja suficientemente pequeno, o algoritmo é terminado e os parâmetros $\vec{w}$ e $\vec{b}$ obtidos são os que configuram a Rede Neural que melhore resolve o problema de interesse. Caso contrário, o algoritmo volta a ajustar os parâmetros e verificar o erro gerado pelos mesmos até que a função $C(\vec{w}, \vec{b})$ retorne um valor desejável.

4.4 O algoritmo da Rede Neural de Classificadores SLS

Neste trabalho é proposto um novo método que explora a estrutura de Redes Neurais aplicada a classificadores SLS, o qual chamaremos de *Rede Neural de Classificadores SLS*, ou simplesmente *Rede SLS*. O algoritmo a ser descrito em seguida é dito como *Classificador* dentro dos diferentes algoritmos em Aprendizagem de Máquina [3] uma vez que sua finalidade é encontrar a qual categoria pre definida um objeto pertence.

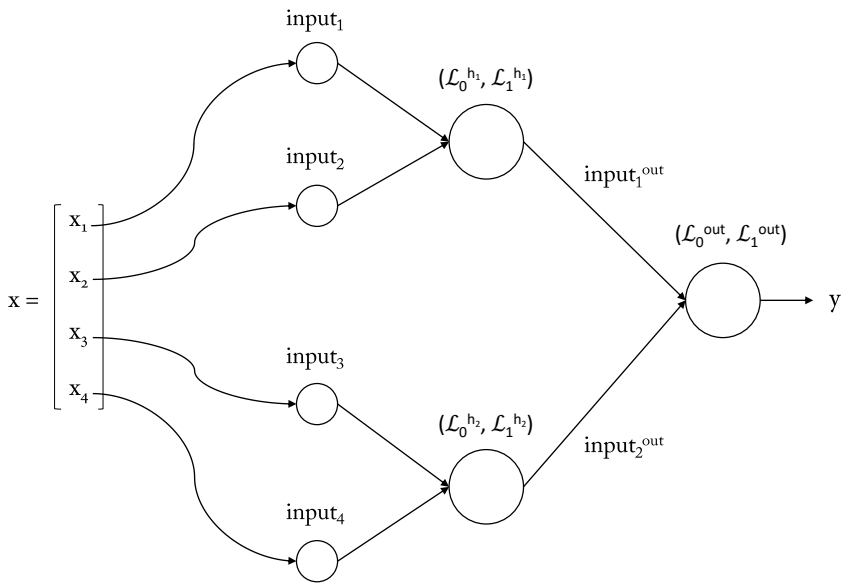

Figura 6: Esboço da rede SLS que será utilizada neste trabalho.

Uma Rede SLS utiliza-se da estrutura de uma Rede Neural em que cada neurônio é um Classificador SLS, como definido na Equação 4, além disso sua última camada tem somente um neurônio. Um esboço de um neurônio SLS pode ser encontrado na Figura 6. Note que o par de conjuntos de segmentos de reta $(\mathcal{L}_0, \mathcal{L}_1)$ define completamente um neurônio SLS.

O exemplo ilustrado pela Figura 6 será utilizado junto ao fato de que quantidade de segmentos de reta N por classe será fixada em $N = 1$ ao longo do desenvolvimento deste trabalho a fim de simplificar os exemplos utilizados e as equações que serão exibidas. Note que nesse exemplo o vetor de entrada $\mathbf{x} = (x_1, x_2, x_3, x_4)^T$ foi dividido em 2 partes, $(x_1, x_2)^T$ e $(x_3, x_4)^T$, de forma que cada parte é entrada de um dos neurônios da primeira camada. Dessa maneira, problemas maiores (por exemplo, de tamanho d) são divididos em problemas menores (nesse caso, em problemas de tamanho $d/2$). Dessa forma, o uso de redes neurais SLS em problemas de Aprendizagem de Máquina é motivado pelo paradigma de divisão e conquista.

Formalmente, uma rede SLS pode ser definida como um grafo dirigido $R = (V, E)$

em que o conjunto de vértices é composto por neurônios $N_i = (\mathcal{L}_0^i, \mathcal{L}_1^i)$, para $i = 1, \ldots, n$, e os arcos indicam como os neurônios de uma camada se conectam a neurônios da camada seguinte (veja um exemplo na Fig. 6).

A classificação de um ponto $\mathbf{x} \in \mathbb{R}^d$ (ou seja, o ato de rotular um objeto como membro da classe 0 ou da classe 1), ponto este que é entrada da rede SLS $R = (V, E)$, é obtida pela aplicação da função de classificação, dada pela Eq. (5), no resultado do único neurônio da camada de saída de $R = (V, E)$ e será denotada por $y_R(\mathbf{x})$.

4.4.1 Inicialização do algoritmo

No primeiro momento é necessário definir o posicionamento inicial dos segmentos de reta em cada um dos nós (neurônios) da rede. De forma análoga ao algoritmo de *classificação SLS*, a inicialização dos segmentos de reta é feita com o auxílio do k-médias.

Tomando o exemplo dado anteriormente, ilustrado pela Figura 6, os dados de entrada são providos como pares ordenados à primeira (e única) camada oculta e a função de ativação $\phi(T_{\mathcal{L}_0,\mathcal{L}_1}(\mathbf{x}))$ é aplicada às mesmas. Para melhor compreensão do procedimento, neste primeiro momento a constante g será fixada em $g = 1.0$.

Suponha que um dos dados de entrada é $\alpha \in \mathbb{R}^4$ tal que $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$. Estas coordenadas são divididas em dois pares $\alpha^{h_1} = (\alpha_1, \alpha_2)$ e $\alpha^{h_2} = (\alpha_3, \alpha_4)$, que são alimentados aos nós ocultos $hidden_1$ e $hidden_2$, respectivamente. Este procedimento é repetido para todos os demais dados de entrada, culminando numa coleção de dados de entrada para o nó $hidden_1$, $input^{h_1}$, e outra para o nó $hidden_2$, $input^{h_2}$. Os conjuntos de entrada são submetidos ao k-médias em seus respectivos nós, em que $k = 2 \cdot N = 2 \cdot 1 = 2$ para cada uma das classes, o que significa que para classe haverá dois centroides, os quais são tomados como as coordenadas iniciais das extremidades $\mathbf{p}$ e $\mathbf{q}$ do segmento de reta.

É importante notar que neste caso, os dados de entrada são bidimensionais, mas os segmentos de reta são alocados no espaço tridimensional. Para que este feito seja possível, os dados de *input* são completados com uma terceira coordenada (fixada em 0), e as

extremidades dos segmentos de reta são inicializados com a última coordenada em 1.

Para a inicialização dos segmentos de reta presentes no nó de saída *out* é necessário primeiramente obter as saídas dos nós da camada oculta, uma vez que os mesmos formarão as entradas para a camada de saída. Diz-se então que $\text{input}^{out} = (\text{input}_1^{out}, \text{input}_2^{out}, 0)$, em que $\text{input}_1^{out} = \phi(T_{\mathcal{L}_0^{h_1}, \mathcal{L}_1^{h_1}}(\text{input}^{h_1}))$ e $\text{input}_2^{out} = \phi(T_{\mathcal{L}_0^{h_2}, \mathcal{L}_1^{h_2}}(\text{input}^{h_2}))$. Perceba que, novamente os dados informados na entrada desta camada são bidimensionais, porém os segmentos de reta são posicionados no espaço tridimensional, o que requer que as entradas sejam completadas em sua terceira coordenada. De forma semelhante aos nós ocultos, as extremidades dos segmentos são resultados do 2-médias aplicado aos dados de cada classe. Uma vez que os segmentos de reta da camada de saída estão inicializados, o erro inicial do sistema é calculado e utilizado como base para as próximas iterações da otimização.

O procedimento descrito acima está transcrito no pseudo-código a seguir para melhor compreensão das etapas.

```
N = 1
g = 1
input_h1 = (input[,1] , input[,2] , 0)
input_h2 = (input[,3] , input[,4] , 0)

%% Inicializacao dos nos ocultos %%
hidden = (hidden_1, hidden_2)
input = (input_h1, input_h2)
saida_h = (saida_h1, saida_h2)

Para i de 1 a 2:
    SLS da classe 0 de hidden[i] = kmeans(input[i] em que classe == 0)
    SLS da classe 1 de hidden[i] = kmeans(input[i] em que classe == 1)

    saida_h[i] = phi(input[i], SLS hidden[i])

input_o = (saida_h1, saida_h2, 0)
SLS da classe 0 de output = kmeans(input_o em que classe == 0)
SLS da classe 1 de output = kmeans(input_o em que classe == 1)
saida_final = phi(input_o, SLS output)

err_inicial = MSE(val_esperado, saida_final)
```

4.4.2 Verificação do Erro

Para avaliar o desempenho da rede SLS optamos por usar o erro quadrático médio (MSE, *Mean Squared Error*).

Dado um conjunto de amostras $S_N = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{0,1\} : i \in \{1,2,\ldots,N\}\}$, o MSE avalia o erro de treinamento de uma rede SLS $R = (V, E)$ por:

$$MSE_{S_N}(R) = \frac{1}{N}\sum_{i=1}^{N}[y_i - y_R(\mathbf{x}_i)]^2 \tag{6}$$

No entanto, é importante notar que é possível que o MSE indique um erro alto, mas que o classificador está devidamente rotulando os objetos. Para avaliar este critério foi definido o erro BIN, que fornece uma taxa de precisão da classificação, no entanto é crucial informar que este erro foi criado com o intuito de observar o comportamento da rede ao longo de seu treinamento, e não foi utilizado no treinamento propriamente dito.

$$BINE_{S_N}(R) = \frac{1}{N}\sum_{i=1}^{N}\Gamma(y_i, y_R(\mathbf{x}_i)) \tag{7}$$

em que

$$\Gamma(a,b) = \begin{cases} 1, & \text{se a = b;} \\ 0, & \text{caso contrário.} \end{cases} \tag{8}$$

4.4.3 Treinamento usando Gradiente Descendente

Dado que as funções envolvidas no sistema são todas conhecidas, as derivadas parciais da função $MSE_{S_N}(R)$ em função das coordenadas das extremidades $\mathbf{p}$ e $\mathbf{q}$ de todas os segmentos de reta presentes no sistema foram calculadas com o auxílio do *Sympy*. As derivadas parciais calculadas podem ser encontradas em detalhe no Apêndice deste trabalho.

Uma vez calculados o gradiente

$$\vec{\nabla}(MSE_{S_N}(R)) = \begin{bmatrix} \frac{\partial MSE_{S_N}(R)}{\partial p^{h_1}_{class_0}} \\ \frac{\partial MSE_{S_N}(R)}{\partial q^{h_1}_{class_0}} \\ \frac{\partial MSE_{S_N}(R)}{\partial p^{h_1}_{class_1}} \\ \frac{\partial MSE_{S_N}(R)}{\partial q^{h_1}_{class_1}} \\ \cdots \\ \frac{\partial MSE_{S_N}(R)}{\partial p^{h_2}_{class_0}} \\ \cdots \\ \frac{\partial MSE_{S_N}(R)}{\partial p^{out}_{class_0}} \\ \cdots \\ \frac{\partial MSE_{S_N}(R)}{\partial q^{out}_{class_1}} \end{bmatrix},$$

e posicionamento corrente dos segmentos de reta

$$\hat{R} = \begin{bmatrix} \hat{p}^{h_1}_{class_0} \\ \hat{q}^{h_1}_{class_0} \\ \hat{p}^{h_1}_{class_1} \\ \hat{q}^{h_1}_{class_1} \\ \cdots \\ \hat{p}^{h_2}_{class_0} \\ \cdots \\ \hat{p}^{out}_{class_0} \\ \cdots \\ \hat{q}^{out}_{class_1} \end{bmatrix},$$

o ajuste dos segmentos de reta é feito usando o GD:

$$R = \begin{bmatrix} p^{h_1}_{class_0} \\ q^{h_1}_{class_0} \\ p^{h_1}_{class_1} \\ q^{h_1}_{class_1} \\ \dots \\ p^{h_2}_{class_0} \\ \dots \\ p^{out}_{class_0} \\ \dots \\ q^{out}_{class_1} \end{bmatrix} = \begin{bmatrix} \hat{p}^{h_1}_{class_0} \\ \hat{q}^{h_1}_{class_0} \\ \hat{p}^{h_1}_{class_1} \\ \hat{q}^{h_1}_{class_1} \\ \dots \\ \hat{p}^{h_2}_{class_0} \\ \dots \\ \hat{p}^{out}_{class_0} \\ \dots \\ \hat{q}^{out}_{class_1} \end{bmatrix} - \eta \cdot \begin{bmatrix} \frac{\partial MSE_{S_N}(R)}{\partial p^{h_1}_{class_0}} \\ \frac{\partial MSE_{S_N}(R)}{\partial q^{h_1}_{class_0}} \\ \frac{\partial MSE_{S_N}(R)}{\partial p^{h_1}_{class_1}} \\ \frac{\partial MSE_{S_N}(R)}{\partial q^{h_1}_{class_1}} \\ \dots \\ \frac{\partial MSE_{S_N}(R)}{\partial p^{h_2}_{class_0}} \\ \dots \\ \frac{\partial MSE_{S_N}(R)}{\partial p^{out}_{class_0}} \\ \dots \\ \frac{\partial MSE_{S_N}(R)}{\partial q^{out}_{class_1}} \end{bmatrix}. \tag{9}$$

Simulando o funcionamento do algoritmo, no momento da inicialização pode-se afirmar que o erro obtido é o menor erro encontrado até o momento, e portanto a rede SLS R_{ini} obtida na inicialização é considerada como o melhor resultado corrente.

Supomos que ao ajustar os segmentos de reta com o GD (descrito pela Equação 9) o MSE obtido é maior que o erro obtido no momento da inicialização. Isso pode ser uma indicação de que a taxa de aprendizado η está maior que o ideal. O resultado obtido é então descartado e o ajuste é refeito, no entanto com $\eta = 0.5\hat{\eta}$.

Supondo que ajustando o valor de η foi possível atingir um erro menor que o erro inicial (considerado até então o menor erro). Assim a rede ajustada R é tomada como o melhor modelo e passa a ser a referência para as próximas iterações. O valor de η também sofre um ajuste a partir do momento que um novo mínimo é encontrado, $\eta = 1.5\hat{\eta}$, uma vez que a diminuição na função custo indica que o algoritmo está avançando na direção correta e, como em qualquer algoritmo de otimização, deseja-se encontrar o mínimo na menor quantidade de iterações possível.

O algoritmo prossegue com novas iterações a menos que o MSE obtido seja menor que uma constante suficientemente pequena (ERRMIN) ou até que o número de iterações

atinja um limite máximo (MAXITER).

No entanto, um problema que foi frequentemente observado em simulações se deu pelo fato dos dados de entrada da camada de saída mudar sempre que os ajustes na camada oculta eram feitos, e em alguns casos essa mudança ocorria de forma drástica e o ajuste feito nos segmentos de reta na camada de saída não era suficiente para mantê-los dentro do escopo dos dados. Este problema é muito bem ilustrado pela Figura 7, em que é claro verificar que os segmentos de reta estão muito distantes dos dados.

Uma solução simples proposta para contornar este problema foi o uso do 2-médias em cima dos novos dados de entrada de cada classe na camada de saída. Isso então gera uma nova rede SLS R cujo erro é possivelmente diferente do erro obtido anteriormente. É então verificado, depois deste reposicionamento dos segmentos de reta, se o MSE obtido diminuiu ou não para determinar se este modelo será tomado como o melhor. Em seguida, para verificar se é possível obter um erro ainda menor, é feito o ajuste dos segmentos de reta reposicionados (**somente** na camada de saída) usando o GD, conforme descrito pela Equação 9), porém somente com os parâmetros presentes na camada de saída. Novamente, se o MSE obtido for menor que o mínimo encontrado até o momento, o modelo ajustado depois do reposicionamento é tomado como o melhor e referência para as futuras iterações. Um exemplo de resultado obtido com o uso desta solução pode ser visto na Figura 8.

Ambas as figuras são de resultados da otimização na camada de saída e provêm do mesmo contexto, sendo a única diferença o uso do reposicionamento com o 2-médias caso o erro obtido no ajuste seja maior que o menor erro encontrado até então. É importante observar que o MSE obtido no caso ilustrado pela Figura 7 foi de aproximadamente 0,25, e seu erro BIN foi de 0,5. Já no caso representado pela Figura 8 obteve MSE de aproximadamente 0,19 e erro BIN de 0,2625.

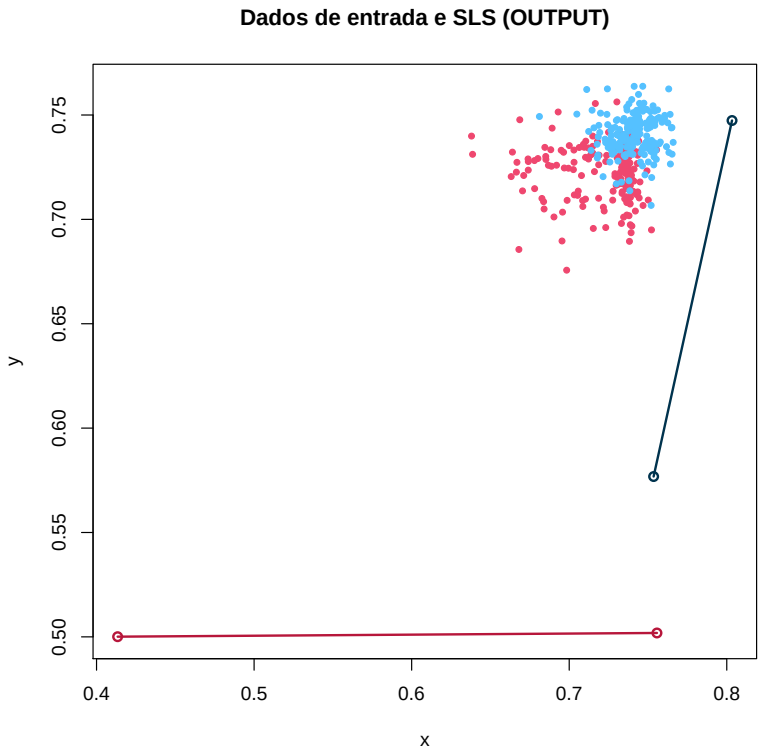

Figura 7: Caso em que os dados modificaram muito depois de um ajuste, mas o ajuste dos SLS neste nó foi insuficiente.

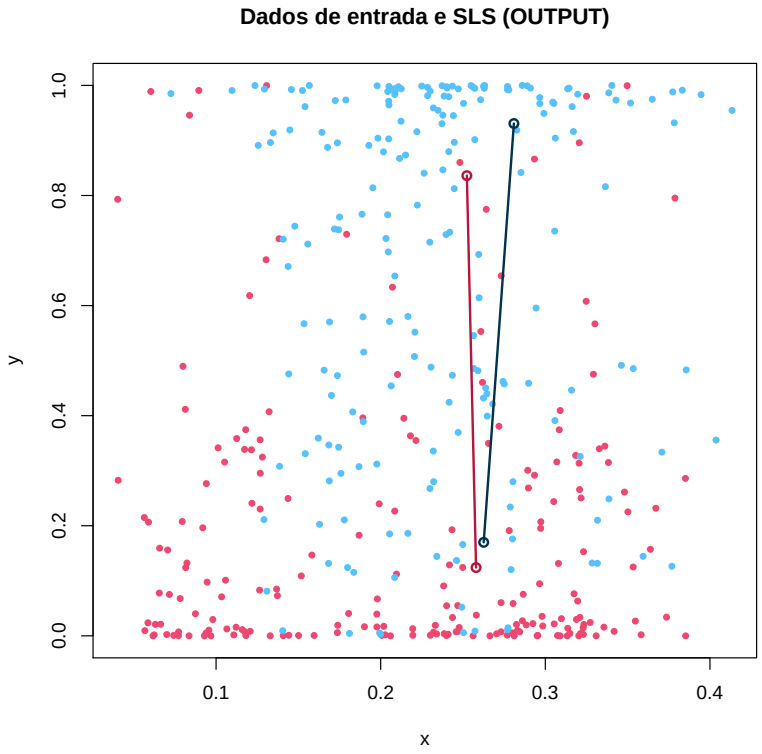

Figura 8: Caso em que a correção usando o k-médias foi utilizada.

```
%% Nos da rede SLS depois da inicializacao %%
best_R = R = (hidden_1, hidden_2, output)
best_ERR = MSE(R)
eta = 0.1
ERRMIN = 0.0001
MAXITER = 100
iter = 1
enquanto (best_ERR > ERRMIN) OU (iter < MAXITER)
    R = backpropagation(best_R)
    Para i de 1 a 2:
        saida_h[i] = phi(input[i], SLS hidden[i])
    input_o = (saida_h1, saida_h2, 0)
    saida_final = phi(input_o, SLS output)
    se MSE(val_esperado, saida_final) <= best_ERR
        best_R = R
        best_ERR = MSE(val_esperado, saida_final)
        eta = 1.5*eta
    senao
        repos SLS da classe 0 = 2-medias(input_o em que classe == 0)
        repos SLS da classe 1 = 2-medias(input_o em que classe == 1)

        saida_final_repos = phi(input_o, SLS output reposicionado)

        R = ajuste_na_camada_de_saida(R_repos)
        saida_final_repos_ajust =
            phi(input_o, SLS output reposicionado e ajustado)
        se MSE(val_esperado, saida_final_repos) <
                MSE(val_esperado, saida_final_repos_ajust)
            R = R_repos
            erro_corr= MSE(val_esperado, saida_final_repos)
        senao
            R = R_repos_ajust
            erro_corr= MSE(val_esperado, saida_final_repos_ajust)
```

```
        se err_corr <= best_ERR
            best_R = R
            best_ERR = err_corr
            eta = 1.5*eta
        senao
            eta = 0.5*eta
    iter++
```

4.4.4 Valor da variável g

Um problema demasiadamente recorrente nas simulações do algoritmo era que o MSE calculado tinha o mínimo muio próximo de 0,25. Observando o comportamento da rede SLS em cada etapa do algoritmo, foi possível concluir que isto se dava pelo fato das saídas da camada oculta serem muito próximas de 0,5 (note que a distância de 0,5 para 0 é a mesma que para 1, e este valor na potência 2 resulta em 0,25, próximo do valor MSE obtido em muitas simulações).

Ao inspecionar as equações com mais cautela, foi possível observar que isso ocorria pois o termo $-g \cdot T_{\mathcal{L}_0,\mathcal{L}_1}(\mathbf{x})$ da Equação 3 resultava em valores muito próximos de 0, e portanto

$$\phi(T_{\mathcal{L}_0,\mathcal{L}_1}(\mathbf{x})) \approx \frac{1}{1+e^0} = \frac{1}{2}$$

Este problema, por sua vez, decorria do fato dos valores retornados pela função $T_{\mathcal{L}_0,\mathcal{L}_1}(\mathbf{x})$ serem demasiadamente pequenos em alguns momentos. Dessa forma, foi decidido que o valor de g seria ajustado em função dos valores de $T_{\mathcal{L}_0,\mathcal{L}_1}(\mathbf{x})$, especificamente:

$$g = \frac{1}{\text{porção mais densa do histograma de } |T_{\mathcal{L}_0,\mathcal{L}_1}(\mathbf{x})|}.$$

É importante notar que g deve ser positivo, uma vez que o sinal do termo $-g \cdot T_{\mathcal{L}_0,\mathcal{L}_1}(\mathbf{x})$, e

logo, de $T_{\mathcal{L}_0,\mathcal{L}_1}(\mathbf{x})$, determina se $\phi(T_{\mathcal{L}_0,\mathcal{L}_1}(\mathbf{x}))$ estará mais próximo de 0 ou de 1.

5 Resultados

5.1 Simulações

Para a realização de simulações é necessário ter uma quantidade significativa de dados. Para tanto, dados artificiais foram gerados a partir de distribuições de probabilidades definidas pelas somas de M normais bidimensionais, conforme descrito em [7]:

$$p(\mathbf{x}, y = C) = \sum_{i=1}^{M} Normal(\mathbf{x}, \mu_i^C, \Sigma_i^C), \tag{10}$$

em que C é o conjunto de classes, $C = 0, 1$, $\mu_i^C \in \mathbb{R}^2$ descreve o centro da curva Normal, P_i é um escalar tal que $\sum_{i=1}^{M} P_i^C = 1$ e Σ é a matriz de covariância (2×2), cujo formato é descrito por:

$$\begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix},$$

em que ρ é a correlação entre as variáveis x_1 e x_2 e σ é o desvio padrão.

Foram então elaboradas quatro distribuições: *Simples*, *distX*, *distS* e *distF*, cada qual definida conforme as descrições abaixo.

5.1.1 Distribuição Simples

$$P_1^0 = P_1^1 = 1$$

$$\mu_1^0 = (0,6; 0,6) \qquad \mu_1^1 = (0,4; 0,4)$$

$$\Sigma_1^0 = \begin{bmatrix} 0,01 & -0,009 \\ -0,009 & 0,01 \end{bmatrix} \qquad \Sigma_1^1 = \begin{bmatrix} 0,01 & 0,009 \\ 0,009 & 0,01 \end{bmatrix}$$

5.1.2 Distribuição distX

$$P_1^0 = P_2^0 = 0,5 \qquad P_1^1 = P_2^1 = 0,5$$

$$\mu_1^0 = (0,25; 0,25) \qquad \mu_1^1 = (0,25; 0,75)$$

$$\mu_2^0 = (0,75; 0,75) \qquad \mu_2^1 = (0,75; 0,25)$$

$$\Sigma_i^C = \begin{bmatrix} 0,04 & 0 \\ 0 & 0,04 \end{bmatrix} : C = 0; 1 \text{ e } i = 1; 2$$

5.1.3 Distribuição distS

$$P_1^0 = P_1^1 = P_2^0 = P_2^1 = 0,5$$

$$\mu_1^0 = (0,4; 0,4) \qquad \mu_2^0 = (0,4; 0,8)$$

$$\mu_1^1 = (0,6; 0,2) \qquad \mu_2^1 = (0,6; 0,6)$$

$$\Sigma_i^C = \begin{bmatrix} 0,02 & 0 \\ 0 & 0,01 \end{bmatrix} : C = 0; 1 \text{ e } i = 1; 2$$

5.1.4 Distribuição distF

$$P_1^0 = P_0^1 = 0,574 \qquad P_2^0 = P_3^0 = P_2^1 = P_3^1 = 0,213$$

$$\mu_1^0 = (0,125; 0,5) \qquad \mu_1^1 = (0,875; 0,5)$$

$$\mu_2^0 = (0,5; 0,375) \qquad \mu_2^1 = (0,5; 0,125)$$

$$\mu_3^0 = (0,5; 0,875) \qquad \mu_3^1 = (0,5; 0,625)$$

$$\Sigma_1^C = \begin{bmatrix} 0,01 & 0 \\ 0 & 0,04 \end{bmatrix}$$

$$\Sigma_2^C = \begin{bmatrix} 0,012 & 0 \\ 0 & 0,01 \end{bmatrix}$$

$$\Sigma_3^C = \begin{bmatrix} 0,012 & 0 \\ 0 & 0,01 \end{bmatrix} \qquad C = 0;1$$

Exemplos de dados gerados pelas distribuições Simples, distX, distS e distF podem ser vistos na Figura 9.

Como este algoritmo é completamente dependente dos dados fornecidos como entrada, é evidente que os resultados variam dentre os diferentes tipo de entradas geradas pelas distribuições listadas acima. Os resultados que serão apresentados nesta seção vão exibir o posicionamento dos segmentos de retas nas camadas oculta e de saída, assim como os dados de entrada para os dois nós ocultos e único nó de saída.

A evolução do MSE e do erro BIN obtidos pelo algoritmo em suas 10 primeiras iterações também será exibido em tabelas. Estas informam o MSE obtido depois do *backpropagation* se o mesmo apresentou uma melhora comparado ao menor erro até o momento. Caso contrário, o MSE obtido depois do reposicionamento feito com 2-médias e o MSE obtido depois do *backpropagation* aplicado somente à camada de saída dado o reposicionamento são exibidos. E finalmente, a taxa de aprendizado η utilizada na iteração também consta na tabela.

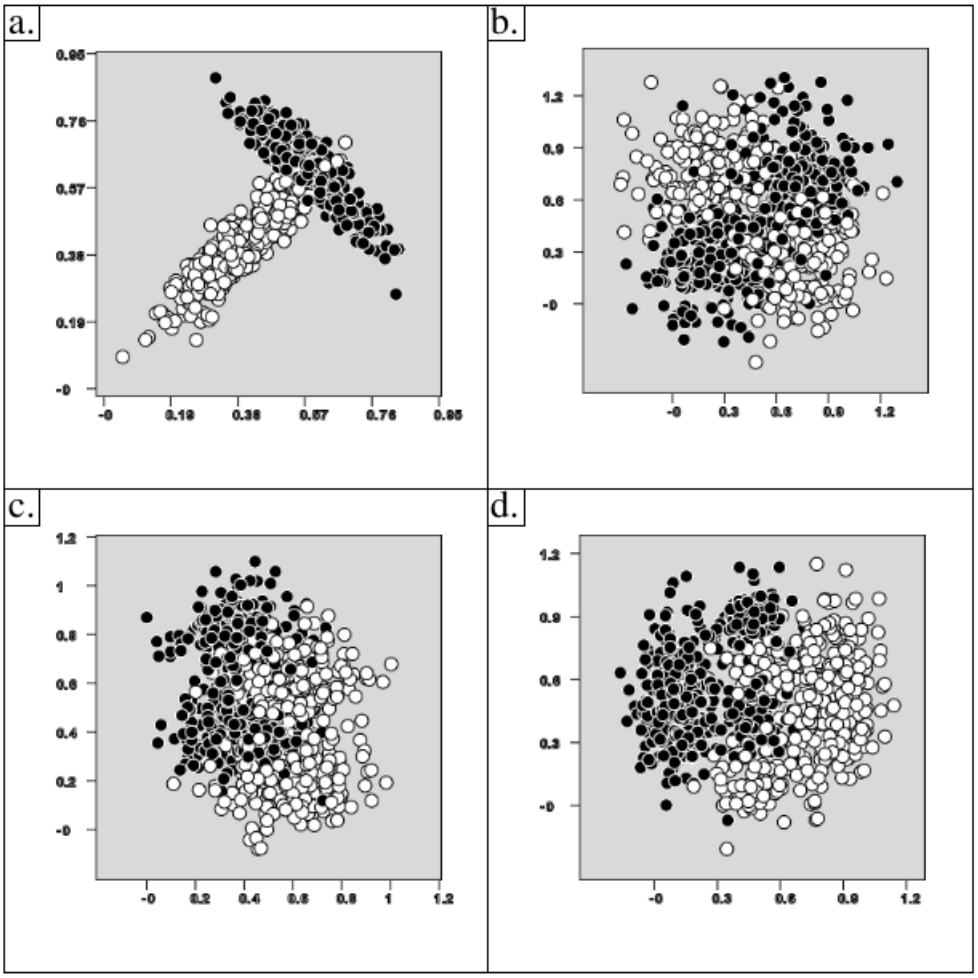

Figura 9: Dados gerado pelas ditribuições (a) Simples, (b) distX, (c) distS, e (d) distF.

5.1.5 Resultados com a distribuição distF

Utilizando-se da configuração descrita pelas Figuras 10, 11 e 12 para classificar o conjunto de dados de teste (não utilizado para o treinamento) para esta distribuição, o MSE e o erro BIN obtidos foram próximos de 0.07227 e 0.09733, respectivamente.

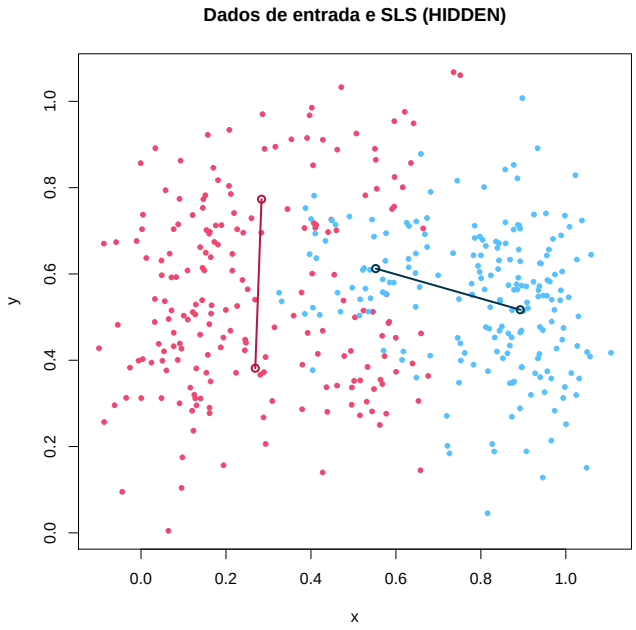

Figura 10: Disposição dos dados gerados pela distribuição distF e segmentos de reta do neurônio oculto hidden_1.

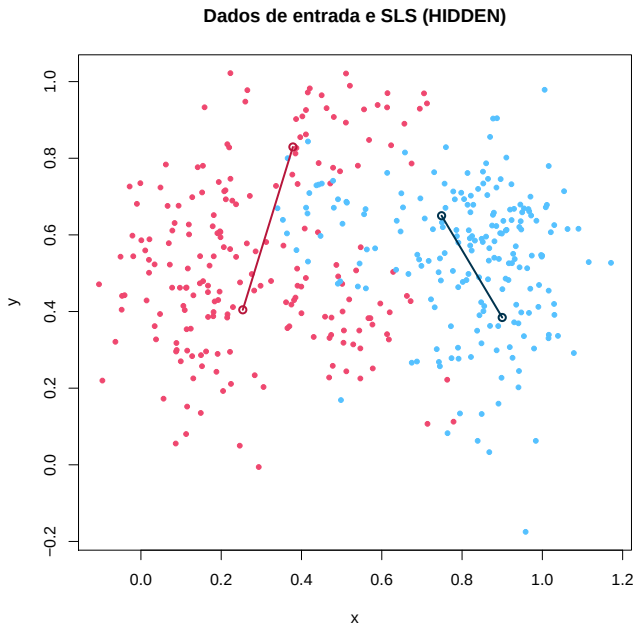

Figura 11: Disposição dos dados gerados pela distribuição distF e segmentos de reta do neurônio oculto hidden_2.

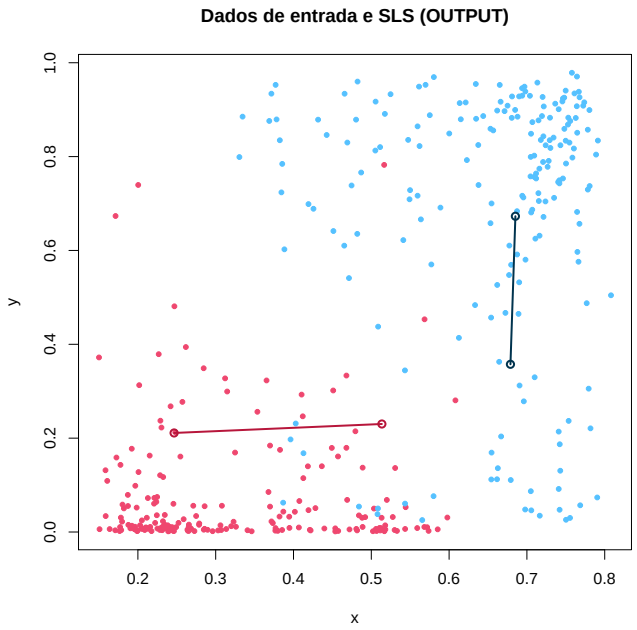

Figura 12: Disposição dos dados resultantes das operações feitas sobre os valores gerados pela distribuição distF e segmentos de reta do neurônio de saída.

Iter	MSE(b)	MSE(b+r)	MSE(b+r+b)	BIN	η
1	0.09461	-	-	0.105	1e-04
2	0.06646	-	-	0.0775	1e-04
3	-	0.16435	0.10646	0.075	0.00011
4	-	0.18226	0.21841	0.0775	5.5e-05
5	-	0.16547	0.13499	0.075	2.75e-05
6	-	0.16384	0.13513	0.0775	1.375e-05
7	-	0.1647	0.14238	0.08	6.875e-06
8	-	0.17622	0.21851	0.08	3.4375e-06
9	-	0.17635	0.21858	0.08	1.71875e-06
10	-	0.16546	0.149	0.08	8.59375e-07

Tabela 1: Progresso do algoritmo com os dados de treinamento gerados pela distribuição distF.

5.1.6 Resultados com a distribuição Simples

Utilizando-se da configuração descrita pelas Figuras 13, 14 e 15 para classificar o conjunto de dados de teste (não utilizado para o treinamento) para esta distribuição, o MSE e o erro BIN obtidos foram próximos de 0.03639 e 0.04996, respectivamente.

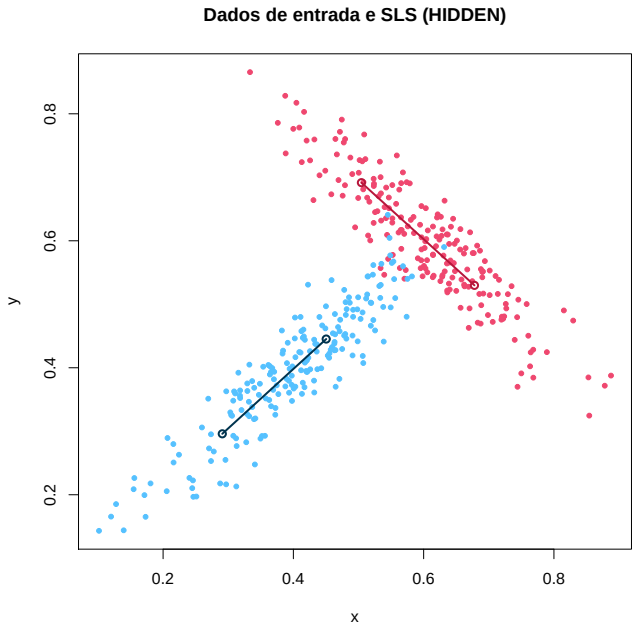

Figura 13: Disposição dos dados gerados pela distribuição Simples e segmentos de reta do neurônio oculto hidden_1.

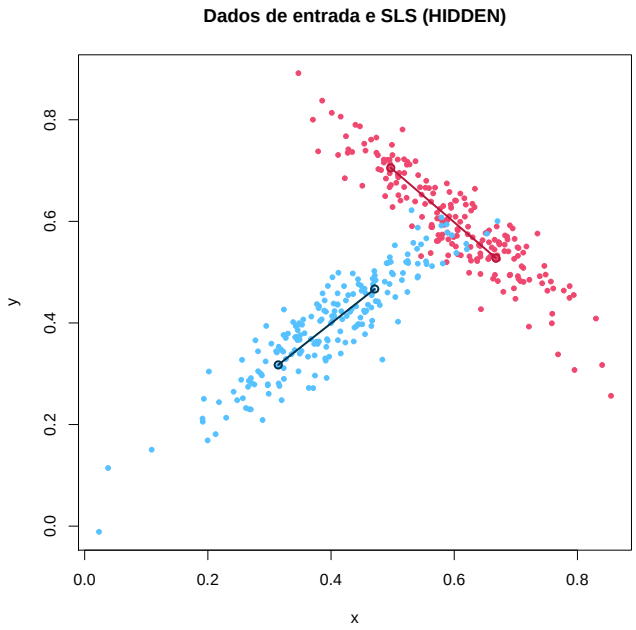

Figura 14: Disposição dos dados gerados pela distribuição Simples e segmentos de reta do neurônio oculto hidden_2.

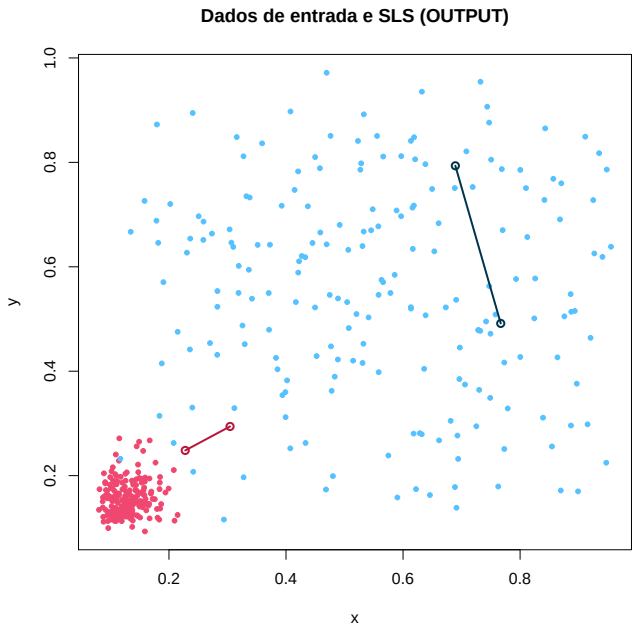

Figura 15: Disposição dos dados resultantes das operações feitas sobre os valores gerados pela distribuição Simples e segmentos de reta do neurônio de saída.

Iter	MSE(b)	MSE(b+r)	MSE(b+r+b)	BIN	η
1	0.0412	-	-	0.015	1e-04
2	-	0.37172	0.18357	0.485	1e-04
3	-	0.18832	0.22331	0.045	5e-05
4	0.03347	-	-	0.045	2.5e-05
5	-	0.16217	0.11371	0.0225	2.75e-05
6	-	0.16384	0.13513	0.0775	1.375e-05
7	-	0.15566	0.10675	0.0075	6.875e-06
8	-	0.16767	0.14857	0.015	3.4375e-06
9	-	0.17017	0.23143	0.015	1.71875e-06
10	-	0.1689	0.23134	0.015	8.59375e-07

Tabela 2: Progresso do algoritmo com os dados de treinamento gerados pela distribuição Simples.

5.1.7 Resultados com a distribuição distS

Utilizando-se da configuração descrita pelas Figuras 16, 17 e 18 para classificar o cojunto de dados de teste (não utilizado para o treinamento) para esta distribuição, o MSE e o erro BIN obtidos foram próximos de 0.18548 e 0.26129, respectivamente.

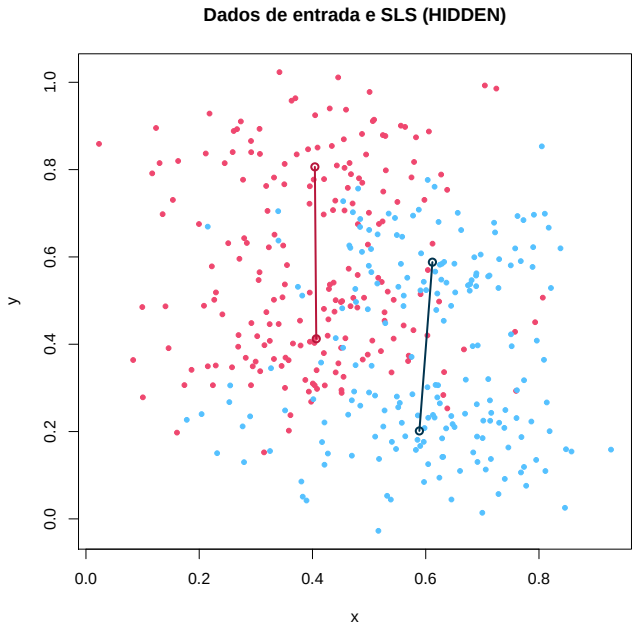

Figura 16: Disposição dos dados gerados pela distribuição distS e segmentos de reta do neurônio oculto hidden_1.

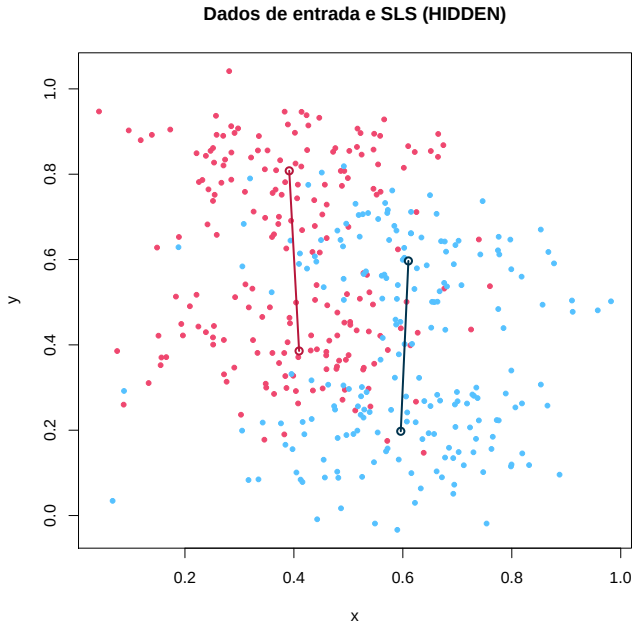

Figura 17: Disposição dos dados gerados pela distribuição distS e segmentos de reta do neurônio oculto hidden_2.

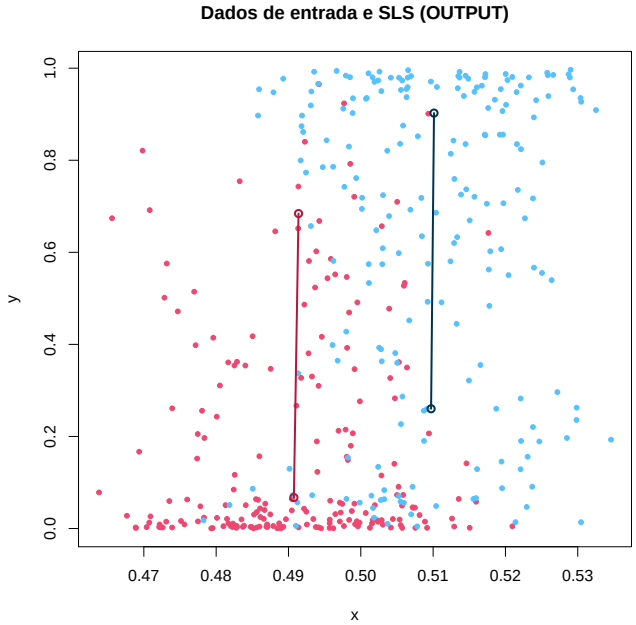

Figura 18: Disposição dos dados resultantes das operações feitas sobre os valores gerados pela distribuição distS e segmentos de reta do neurônio de saída.

Iter	MSE(b)	MSE(b+r)	MSE(b+r+b)	BIN	η
1	0.18445	-	-	0.265	1e-04
2	0.18037	-	-	0.24	1e-04
3	-	0.24901	0.23852	0.5	0.00011
4	-	0.24899	0.23968	0.5	5.5e-05
5	-	0.24895	0.23935	0.445	2.75e-05
6	-	0.24895	0.23896	0.36	1.375e-05
7	-	0.24898	0.23954	0.3075	6.875e-06
8	-	0.24902	0.24	0.2875	3.4375e-06
9	-	0.24894	0.23928	0.2675	1.71875e-06
10	-	0.18037	0.23968	0.26	8.59375e-07

Tabela 3: Progresso do algoritmo com os dados de treinamento gerados pela distribuição distS.

5.1.8 Resultados com a distribuição distX

Utilizando-se da configuração descrita pelas Figuras 19, 20 e 21 para classificar o conjunto de dados de teste (não utilizado para o treinamento) para esta distribuição, o MSE e o erro BIN obtidos foram próximos de 0.18558 e 0.25455, respectivamente.

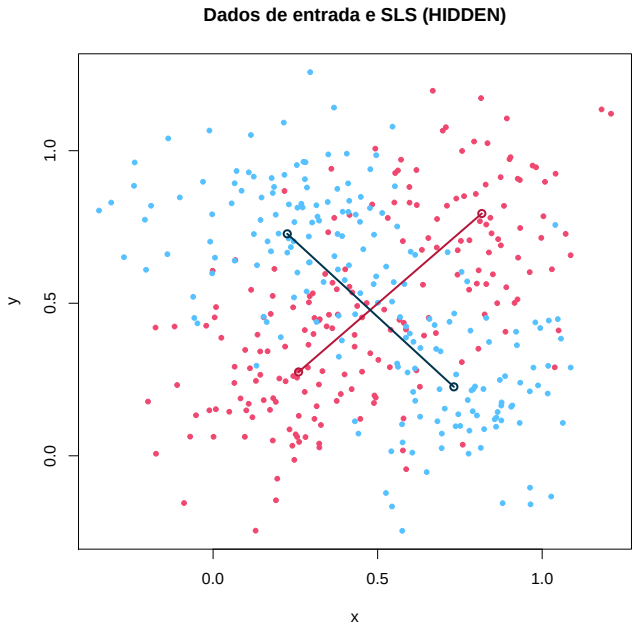

Figura 19: Disposição dos dados gerados pela distribuição distX e segmentos de reta do neurônio oculto hidden_1.

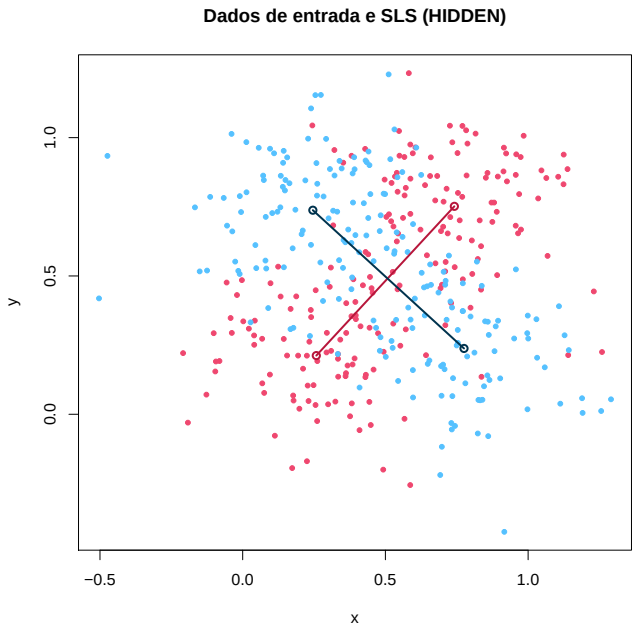

Figura 20: Disposição dos dados gerados pela distribuição distX e segmentos de reta do neurônio oculto hidden_2.

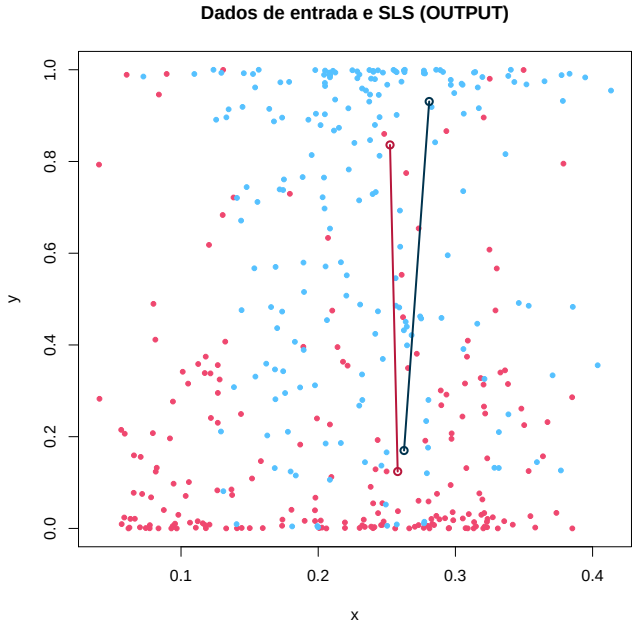

Figura 21: Disposição dos dados resultantes das operações feitas sobre os valores gerados pela distribuição distX e segmentos de reta do neurônio de saída.

Iter	MSE(b)	MSE(b+r)	MSE(b+r+b)	BIN	η
1	0.28053	-	-	0.5	1e-04
2	-	0.24999	0.49928	0.5	1e-04
3	-	0.25137	0.26311	0.5	5e-05
4	-	0.26311	0.2109	0.5	5.5e-05
5	-	0.31166	0.45385	0.5	6.05e-05
6	-	0.31158	0.3112	0.5	3.025e-05
7	-	0.49541	0.34886	0.4875	1.5125e-05
8	-	0.48817	0.23385	0.375	7.5625e-06
9	0.186876	-	-	0.2625	3.78125e-06
10	-	0.28598	0.33055	0.5	4.159375e-06

Tabela 4: Progresso do algoritmo com os dados de treinamento gerados pela distribuição distX.

5.2 Desafios encontrados na trajetória desse trabalho

Além das dificuldades relacionadas à decisão do valor da variável g e a brusca mudança dos dados de entrada no nó de saída depois de certos ajustes (já citados e detalhados em seções anteriores), outras dificuldades foram observadas no processo de implementação e idealização do projeto, dos quais os mais importantes serão detalhados nesta seção.

5.2.1 Dados de entrada

Muitas vezes um dos maiores desafios na esfera de Aprendizagem de Máquina é encontrar uma coletânea de dados para treinamento e teste de um sistema que sejam apropriados para o algoritmo.

No caso deste trabalho, estes dados foram fabricados a partir das distribuições *distF*, *Simples*, *distS* e *distX* definidas anteriormente e organizados de forma arbitrária em pares ordenados, o que no primeiro momento aparenta ser razoável, mas isto pode ter comprometido com o desempenho do algoritmo.

Ao julgar pelos resultados obtidos nas camadas ocultas, há de se admitir que o método demonstrou certa eficiência em identificar as classes e suas principais características, principalmente nos nós ocultos, onde as entradas são os próprios dados gerados.

5.2.2 Função T do classificador SLS

A função descrita na Equação 2 consta na sua definição original [7, 8, 9, 10, 11, 12] como:

$$T_{\mathcal{L}_0,\mathcal{L}_1}(\mathbf{x}) = \lim_{\epsilon \to 0} \left(\sum_{\overline{\mathbf{pq}} \in \mathcal{L}_1} \frac{1}{pdist(\mathbf{x}, \overline{\mathbf{pq}})} - \sum_{\overline{\mathbf{pq}} \in \mathcal{L}_0} \frac{1}{pdist(\mathbf{x}, \overline{\mathbf{pq}})} \right). \tag{11}$$

No entanto, por se tratar de uma cadeia de classificadores SLS, houve uma preocupação com relação às derivadas parciais do MSE em função às extremidades de segmentos de reta de camadas ocultas. A fim de simplificar as contas a serem feitas no GD foi proposta

a Equação 2, deixando de ser uma soma do inverso da função $pdist(\mathbf{x}, \overline{\mathbf{pq}})$ para ser a soma dessas mesmas funções.

Nota-se que em termos práticos, a função não teve o seu papel ou o seu significado comprometido. Quando o objeto se encontra mais próximo da classe 0, o retorno da Equação 2 tem sinal positivo, assim como na Equação 11. E quando o objeto a ser classificado está mais próximo da classe 1, o retorno da Equação 2 é negativo, o que também pode ser verificado na Equação 11.

5.2.3 Aleatoriedade no k-médias

Como foi citado anteriormente, pelo fato do k-médias não ser o objeto principal deste trabalho e ser usado simplesmente como uma ferramenta de apoio, foi decidido que o trabalho se utilizaria da implementação já existente da função *kmeans* do pacote *stat* do R.

No entanto, sem o próprio cuidado o uso desta função acarreta em resultados inconsistentes entre diferentes execuções, o que torna a verificação e correção do algoritmo difícil. Isso se dá pelo fato do algoritmo ser sensível à escolha de pontos iniciais para a otimização, o qual é feito aleatoriamente.

Para manter as simulações consistentes, foi necessário fixar o *seed* do gerador de valores aleatórios, dessa forma a inicialização dok-médias é sempre feita da mesma forma, logo, as partições e seus centroides também são mantidos.

6 Possíveis aprimoramentos no algoritmo

Nesta seção algumas das ideias discutidas nas reuniões com o orientador serão expostas, porém as mesmas não foram implementadas, seja por falta de tempo ou viabilidade.

6.1 Otimização da variável g pelo GD

O valor da variável g influencia diretamente na função de ativação ϕ, e logo nos erros obtidos pelo MSE. Atualmente o seu valor é ajustado de acordo com os valores obtidos na função T, porém é possível que o desempenho do algoritmo melhore se g também for treinado como auxílio do GD.

Para tanto, será necessário calcular as derivadas parciais do MSE em respeito aos g's do sistema.

6.2 Valor da constante N

Um dos pontos discutidos no contexto de classificadores SLS é a questão da quantidade de segmentos de reta N ideal para cada classe, de forma a obter a melhor desempenho possível do classificador. No entanto, esta mudança influenciaria diretamente na complexidade do algoritmo, visto que N está diretamente ligado ao tamanho e componentes do gradiente $\vec{\nabla}(MSE)$.

Apesar da implementação ideal parecer fora de alcance, um experimento que pode ser conduzido futuramente é repetir os testes realizados com $N \neq 1$, observando o posicionamento final dos segmentos e como a quantidade N pode influenciar na precisão do método.

6.3 Inicialização com k-médias

Em sua implementação atual o algoritmo tem o *seed* do gerador aleatório fixado numa constante arbitrária, o que torna os retornos da função *kmeans* consistentes, fator

desejável principalmente na fase de testes e verificação do algoritmo e o desempenho do classificador resultante.

No entanto, em termos práticos, quanto mais próximo ponto inicial dos parâmetros estiver do mínimo global, mais rápida será a convergência do erro para o mínimo. Considerando este ponto, é possível fazer a chamada da função *kmeans* um número arbitrário de vezes sem que o *seed* esteja fixado, e assim escolher a inicialização que retorna o menor MSE. Apesar de não haver meios de garantir que o ponto inicial adotado é ou está próximo do mínimo global, este procedimento pode trazer mais estabilidade à inicialização sem ter que limitar o gerador de valores aleatórios.

7 Conclusão

Há muitas formas de se implementar um classificador binário. Nesse trabalho, propomos um novo classificador baseado em distância a segmentos de reta dispostos em uma rede neural. Resultados preliminares utilizando dados simulados a partir de distribuições de probabilidade conhecidas indicam a viabilidade do método.

Apesar dos resultados obtidos não serem tão bons quanto o esperado, os mesmos têm potencial de serem melhorados com estudo e a implementação de novos métodos. Dessa forma, concluo esse trabalho, com a satisfação de ter implementado e explorado algo que ainda não era conhecido, com a esperança de que algum leitor há de se intrigar com essa experiência e motivar-se a explorar novas possibilidades.

1 Apêndice A: Derivadas parciais calculadas

É importante ressaltar que estas derivadas parciais foram calculadas para a rede SLS descrita por 6. A seguir, as derivadas parciais serão apresentadas por classe, observe que $\frac{\partial err}{\partial extrem^{node}_{class_i}}$ denota a derivada parcial de err em função da coordenada i na extremidade $extrem$ do nó $node$ desta rede SLS.

1.1 Classe 0 do nó de saída

$$\frac{\partial err}{\partial p^{out}_{0_x}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})} \left[-\frac{\left(p^{out}_{0_x} - q^{out}_{0_x}\right)}{2dist(p^{out}_0, q^{out}_0)} + \frac{\left(p^{out}_{0_x} - input^{out}_x\right)}{2dist(input^{out}, p^{out}_0)}\right]$$

$$\frac{\partial err}{\partial p^{out}_{0_y}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})} \left[-\frac{\left(p^{out}_{0_y} - q^{out}_{0_y}\right)}{2dist(p^{out}_0, q^{out}_0)} + \frac{\left(p^{out}_{0_y} - input^{out}_y\right)}{2dist(input^{out}, p^{out}_0)}\right]$$

$$\frac{\partial err}{\partial p^{out}_{0_z}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})} \left[-\frac{\left(p^{out}_{0_z} - q^{out}_{0_z}\right)}{2dist(p^{out}_0, q^{out}_0)} + \frac{\left(p^{out}_{0_z} - input^{out}_z\right)}{2dist(input^{out}, p^{out}_0)}\right]$$

$$\frac{\partial err}{\partial q_{0_x}^{out}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left[\frac{\left(p_{0_x}^{out} - q_{0_x}^{out}\right)}{2dist(p_0^{out}, q_0^{out})} + \frac{\left(q_{0_x}^{out} - input_x^{out}\right)}{2dist(input^{out}, q_0^{out})}\right]$$

$$\frac{\partial err}{\partial q_{0_y}^{out}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left[\frac{\left(p_{0_y}^{out} - q_{0_y}^{out}\right)}{2dist(p_0^{out}, q_0^{out})} + \frac{\left(q_{0_y}^{out} - input_y^{out}\right)}{2dist(input^{out}, q_0^{out})}\right]$$

$$\frac{\partial err}{\partial q_{0_z}^{out}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left[\frac{\left(p_{0_z}^{out} - q_{0_z}^{out}\right)}{2dist(p_0^{out}, q_0^{out})} + \frac{\left(q_{0_z}^{out} - input_z^{out}\right)}{2dist(input^{out}, q_0^{out})}\right]$$

1.2 Classe 1 do nó de saída

$$\frac{\partial err}{\partial p_{1_x}^{out}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left[\frac{\left(p_{1_x}^{out} - q_{1_x}^{out}\right)}{2dist(p_1^{out}, q_1^{out})} - \frac{\left(p_{1_x}^{out} - input_x^{out}\right)}{2dist(input^{out}, p_1^{out})}\right]$$

$$\frac{\partial err}{\partial p_{1_y}^{out}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left[\frac{\left(p_{1_y}^{out} - q_{1_y}^{out}\right)}{2dist(p_1^{out}, q_1^{out})} - \frac{\left(p_{1_y}^{out} - input_y^{out}\right)}{2dist(input^{out}, p_1^{out})}\right]$$

$$\frac{\partial err}{\partial p_{1_z}^{out}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left[\frac{\left(p_{1_z}^{out} - q_{1_z}^{out}\right)}{2dist(p_1^{out}, q_1^{out})} - \frac{\left(p_{1_z}^{out} - input_z^{out}\right)}{2dist(input^{out}, p_1^{out})}\right]$$

$$\frac{\partial err}{\partial q_{1_x}^{out}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left[-\frac{\left(p_{1_x}^{out} - q_{1_x}^{out}\right)}{2dist(p_1^{out}, q_1^{out})} - \frac{\left(q_{1_x}^{out} - input_x^{out}\right)}{2dist(input^{out}, q_1^{out})}\right]$$

$$\frac{\partial err}{\partial q_{1_y}^{out}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left[-\frac{\left(p_{1_y}^{out} - q_{1_y}^{out}\right)}{2dist(p_1^{out}, q_1^{out})} - \frac{\left(q_{1_y}^{out} - input_y^{out}\right)}{2dist(input^{out}, q_1^{out})}\right]$$

$$\frac{\partial err}{\partial q_{1_z}^{out}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left[-\frac{\left(p_{1_z}^{out} - q_{1_z}^{out}\right)}{2dist(p_1^{out}, q_1^{out})} - \frac{\left(q_{1_z}^{out} - input_z^{out}\right)}{2dist(input^{out}, q_1^{out})}\right]$$

1.3 Classe 0 do primeiro nó da camada oculta

$$\frac{\partial err}{\partial p_{0_x}^{h_1}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_x^{out}\right)^2 e^{-g_{h1}T(input^{h_1},\mathcal{L}^{h_1})} g_{h1}\left[-\frac{p_{0_x}^{h_1}-q_{0_x}^{h_1}}{2dist(p_0^{h_1},q_0^{h_1})} + \frac{p_{0_x}^{h_1}-input_x^{h_1}}{2dist(input^{h_1},p_0^{h_1})}\right]$$
$$\left[\frac{\left(-p_{0_x}^{out}+input_x^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_x}^{out}+input_x^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_x}^{out}+input_x^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_x}^{out}+input_x^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial p_{0_y}^{h_1}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_x^{out}\right)^2 e^{-g_{h1}T(input^{h_1},\mathcal{L}^{h_1})} g_{h1}\left[-\frac{p_{0_y}^{h_1}-q_{0_y}^{h_1}}{2dist(p_0^{h_1},q_0^{h_1})} + \frac{p_{0_y}^{h_1}-input_y^{h_1}}{2dist(input^{h_1},p_0^{h_1})}\right]$$
$$\left[\frac{\left(-p_{0_x}^{out}+input_x^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_x}^{out}+input_x^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_x}^{out}+input_x^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_x}^{out}+input_x^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial p_{0_z}^{h_1}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_x^{out}\right)^2 e^{-g_{h1}T(input^{h_1},\mathcal{L}^{h_1})} g_{h1}\left[-\frac{p_{0_z}^{h_1}-q_{0_z}^{h_1}}{2dist(p_0^{h_1},q_0^{h_1})} + \frac{p_{0_z}^{h_1}-input_z^{h_1}}{2dist(input^{h_1},p_0^{h_1})}\right]$$
$$\left[\frac{\left(-p_{0_x}^{out}+input_x^{out}\right)}{2dist(input^{out},p_0^{out})} - \frac{\left(-p_{1_x}^{out}+input_x^{out}\right)}{2dist(input^{out},p_1^{out})} + \frac{\left(-q_{0_x}^{out}+input_x^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-q_{1_x}^{out}+input_x^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial q_{0_x}^{h_1}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_x^{out}\right)^2 e^{-g_{h1}T(input^{h_1},\mathcal{L}^{h_1})} g_{h1}\left[\frac{p_{0_x}^{h_1}-q_{0_x}^{h_1}}{2dist(p_0^{h_1},q_0^{h_1})} + \frac{q_{0_x}^{h_1}-input_x^{h_1}}{2dist(input^{h_1},q_0^{h_1})}\right]$$
$$\left[\frac{\left(-p_{0_x}^{out}+input_x^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_x}^{out}+input_x^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_x}^{out}+input_x^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_x}^{out}+input_x^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial q_{0_y}^{h_1}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right]e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_x^{out}\right)^2 e^{-g_{h1}T(input^{h_1},\mathcal{L}^{h_1})} g_{h1}\left[\frac{p_{0_y}^{h_1} - q_{0_y}^{h_1}}{2dist(p_0^{h_1}, q_0^{h_1})} + \frac{q_{0_y}^{h_1} - input_y^{h_1}}{2dist(input^{h_1}, q_0^{h_1})}\right]$$
$$\left[\frac{\left(-p_{0_x}^{out} + input_x^{out}\right)}{2dist(input^{out}, p_0^{out})} + \frac{\left(-q_{0_x}^{out} + input_x^{out}\right)}{2dist(input^{out}, q_0^{out})} - \frac{\left(-p_{1_x}^{out} + input_x^{out}\right)}{2dist(input^{out}, p_1^{out})} - \frac{\left(-q_{1_x}^{out} + input_x^{out}\right)}{2dist(input^{out}, q_1^{out})}\right]$$

$$\frac{\partial err}{\partial q_{0_z}^{h_1}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right]e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_x^{out}\right)^2 e^{-g_{h1}T(input^{h_1},\mathcal{L}^{h_1})} g_{h1}\left[\frac{p_{0_z}^{h_1} - q_{0_z}^{h_1}}{2dist(p_0^{h_1}, q_0^{h_1})} + \frac{q_{0_z}^{h_1} - input_z^{h_1}}{2dist(input^{h_1}, q_0^{h_1})}\right]$$
$$\left[\frac{\left(-p_{0_x}^{out} + input_x^{out}\right)}{2dist(input^{out}, p_0^{out})} + \frac{\left(-q_{0_x}^{out} + input_x^{out}\right)}{2dist(input^{out}, q_0^{out})} - \frac{\left(-p_{1_x}^{out} + input_x^{out}\right)}{2dist(input^{out}, p_1^{out})} - \frac{\left(-q_{1_x}^{out} + input_x^{out}\right)}{2dist(input^{out}, q_1^{out})}\right]$$

1.4 Classe 1 do primeiro nó da camada oculta

$$\frac{\partial err}{\partial p_{1_x}^{h_1}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right]e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_x^{out}\right)^2 e^{-g_{h1}T(input^{h_1},\mathcal{L}^{h_1})} g_{h1}\left[\frac{p_{1_x}^{h_1}-q_{1_x}^{h_1}}{2dist(p_1^{h_1},q_1^{h_1})} - \frac{p_{1_x}^{h_1}-input_x^{h_1}}{2dist(input^{h_1},p_1^{h_1})}\right]$$
$$\left[\frac{\left(-p_{0_x}^{out}+input_x^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_x}^{out}+input_x^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_x}^{out}+input_x^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_x}^{out}+input_x^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial p_{1_y}^{h_1}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right]e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_x^{out}\right)^2 e^{-g_{h1}T(input^{h_1},\mathcal{L}^{h_1})} g_{h1}\left[\frac{p_{1_y}^{h_1}-q_{1_y}^{h_1}}{2dist(p_1^{h_1},q_1^{h_1})} - \frac{p_{1_y}^{h_1}-input_y^{h_1}}{2dist(input^{h_1},p_1^{h_1})}\right]$$
$$\left[\frac{\left(-p_{0_x}^{out}+input_x^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_x}^{out}+input_x^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_x}^{out}+input_x^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_x}^{out}+input_x^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial p_{1_z}^{h_1}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right]e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_x^{out}\right)^2 e^{-g_{h1}T(input^{h_1},\mathcal{L}^{h_1})} g_{h1}\left[\frac{p_{1_z}^{h_1}-q_{1_z}^{h_1}}{2dist(p_1^{h_1},q_1^{h_1})} - \frac{p_{1_z}^{h_1}-input_z^{h_1}}{2dist(input^{h_1},p_1^{h_1})}\right]$$
$$\left[\frac{\left(-p_{0_x}^{out}+input_x^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_x}^{out}+input_x^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_x}^{out}+input_x^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_x}^{out}+input_x^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial q_{1_x}^{h_1}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right]e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_x^{out}\right)^2 e^{-g_{h1}T(input^{h_1},\mathcal{L}^{h_1})} g_{h1}\left[-\frac{p_{1_x}^{h_1}-q_{1_x}^{h_1}}{2dist(p_1^{h_1},q_1^{h_1})} - \frac{q_{1_x}^{h_1}-input_x^{h_1}}{2dist(input^{h_1},q_1^{h_1})}\right]$$
$$\left[\frac{\left(-p_{0_x}^{out}+input_x^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_x}^{out}+input_x^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_x}^{out}+input_x^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_x}^{out}+input_x^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial q_{1_y}^{h_1}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_x^{out}\right)^2 e^{-g_{h1}T(input^{h_1},\mathcal{L}^{h_1})} g_{h1}\left[-\frac{p_{1_y}^{h_1} - q_{1_y}^{h_1}}{2dist(p_1^{h_1}, q_1^{h_1})} - \frac{q_{1_y}^{h_1} - input_y^{h_1}}{2dist(input^{h_1}, q_1^{h_1})}\right]$$
$$\left[\frac{\left(-p_{0_x}^{out} + input_x^{out}\right)}{2dist(input^{out}, p_0^{out})} + \frac{\left(-q_{0_x}^{out} + input_x^{out}\right)}{2dist(input^{out}, q_0^{out})} - \frac{\left(-p_{1_x}^{out} + input_x^{out}\right)}{2dist(input^{out}, p_1^{out})} - \frac{\left(-q_{1_x}^{out} + input_x^{out}\right)}{2dist(input^{out}, q_1^{out})}\right]$$

$$\frac{\partial err}{\partial q_{1_z}^{h_1}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_x^{out}\right)^2 e^{-g_{h1}T(input^{h_1},\mathcal{L}^{h_1})} g_{h1}\left[-\frac{p_{1_z}^{h_1} - q_{1_z}^{h_1}}{2dist(p_1^{h_1}, q_1^{h_1})} - \frac{q_{1_z}^{h_1} - input_z^{h_1}}{2dist(input^{h_1}, q_1^{h_1})}\right]$$
$$\left[\frac{\left(-p_{0_x}^{out} + input_x^{out}\right)}{2dist(input^{out}, p_0^{out})} + \frac{\left(-q_{0_x}^{out} + input_x^{out}\right)}{2dist(input^{out}, q_0^{out})} - \frac{\left(-p_{1_x}^{out} + input_x^{out}\right)}{2dist(input^{out}, p_1^{out})} - \frac{\left(-q_{1_x}^{out} + input_x^{out}\right)}{2dist(input^{out}, q_1^{out})}\right]$$

1.5 Classe 0 do segundo nó da camada oculta

$$\frac{\partial err}{\partial p_{0_x}^{h_2}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right]e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_y^{out}\right)^2 e^{-g_{h2}T(input^{h_2},\mathcal{L}^{h_2})} g_{h2}\left[-\frac{p_{0_x}^{h_2}-q_{0_x}^{h_2}}{2dist(p_0^{h_2},q_0^{h_2})} + \frac{p_{0_x}^{h_2}-input_x^{h_2}}{2dist(input^{h_2},p_0^{h_2})}\right]$$
$$\left[\frac{\left(-p_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial p_{0_y}^{h_2}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right]e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_y^{out}\right)^2 e^{-g_{h2}T(input^{h_2},\mathcal{L}^{h_2})} g_{h2}\left[-\frac{p_{0_y}^{h_2}-q_{0_y}^{h_2}}{2dist(p_0^{h_2},q_0^{h_2})} + \frac{p_{0_y}^{h_2}-input_y^{h_2}}{2dist(input^{h_2},p_0^{h_2})}\right]$$
$$\left[\frac{\left(-p_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial p_{0_z}^{h_2}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right]e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_y^{out}\right)^2 e^{-g_{h2}T(input^{h_2},\mathcal{L}^{h_2})} g_{h2}\left[-\frac{p_{0_z}^{h_2}-q_{0_z}^{h_2}}{2dist(p_0^{h_2},q_0^{h_2})} + \frac{p_{0_z}^{h_2}-input_z^{h_2}}{2dist(input^{h_2},p_0^{h_2})}\right]$$
$$\left[\frac{\left(-p_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial q_{0_x}^{h_2}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_y^{out}\right)^2 e^{-g_{h2}T(input^{h_2},\mathcal{L}^{h_2})} g_{h2}\left[-\frac{-p_{0_x}^{h_2}+q_{0_x}^{h_2}}{2dist(p_0^{h_2},q_0^{h_2})} + \frac{q_{0_x}^{h_2}-input_x^{h_2}}{2dist(input^{h_2},q_0^{h_2})}\right]$$
$$\left[\frac{\left(-p_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial q_{0_y}^{h_2}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_y^{out}\right)^2 e^{-g_{h2}T(input^{h_2},\mathcal{L}^{h_2})} g_{h2}\left[\frac{p_{0_y}^{h_2}-q_{0_y}^{h_2}}{2dist(p_0^{h_2},q_0^{h_2})} + \frac{q_{0_y}^{h_2}-input_y^{h_2}}{2dist(input^{h_2},q_0^{h_2})}\right]$$
$$\left[\frac{\left(-p_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial q_{0_z}^{h_2}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right] e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_y^{out}\right)^2 e^{-g_{h2}T(input^{h_2},\mathcal{L}^{h_2})} g_{h2}\left[\frac{p_{0_z}^{h_2}-q_{0_z}^{h_2}}{2dist(p_0^{h_2},q_0^{h_2})} + \frac{q_{0_z}^{h_2}-input_z^{h_2}}{2dist(input^{h_2},q_0^{h_2})}\right]$$
$$\left[\frac{\left(-p_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

1.6 Classe 1 do segundo nó da camada oculta

$$\frac{\partial err}{\partial p_{1_x}^{h_2}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right]e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_y^{out}\right)^2 e^{-g_{h2}T(input^{h_2},\mathcal{L}^{h_2})} g_{h2}\left[\frac{p_{1_x}^{h_2}-q_{1_x}^{h_2}}{2dist(p_1^{h_2},q_1^{h_2})} - \frac{p_{1_x}^{h_2}-input_x^{h_2}}{2dist(input^{h_2},p_1^{h_2})}\right]$$
$$\left[\frac{\left(-p_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial p_{1_y}^{h_2}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right]e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_y^{out}\right)^2 e^{-g_{h2}T(input^{h_2},\mathcal{L}^{h_2})} g_{h2}\left[\frac{p_{1_y}^{h_2}-q_{1_y}^{h_2}}{2dist(p_1^{h_2},q_1^{h_2})} - \frac{p_{1_y}^{h_2}-input_y^{h_2}}{2dist(input^{h_2},p_1^{h_2})}\right]$$
$$\left[\frac{\left(-p_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial p_{1_z}^{h_2}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right]e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_y^{out}\right)^2 e^{-g_{h2}T(input^{h_2},\mathcal{L}^{h_2})} g_{h2}\left[\frac{p_{1_z}^{h_2}-q_{1_z}^{h_2}}{2dist(p_1^{h_2},q_1^{h_2})} - \frac{p_{1_z}^{h_2}-input_z^{h_2}}{2dist(input^{h_2},p_1^{h_2})}\right]$$
$$\left[\frac{\left(-p_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial q_{1_x}^{h_2}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right]e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_y^{out}\right)^2 e^{-g_{h2}T(input^{h_2},\mathcal{L}^{h_2})}g_{h2}\left[-\frac{p_{1_x}^{h_2}-q_{1_x}^{h_2}}{2dist(p_1^{h_2},q_1^{h_2})} - \frac{q_{1_x}^{h_2}-input_x^{h_2}}{2dist(input^{h_2},q_1^{h_2})}\right]$$
$$\left[\frac{\left(-p_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial q_{1_y}^{h_2}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right]e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_y^{out}\right)^2 e^{-g_{h2}T(input^{h_2},\mathcal{L}^{h_2})}g_{h2}\left[-\frac{p_{1_y}^{h_2}-q_{1_y}^{h_2}}{2dist(p_1^{h_2},q_1^{h_2})} - \frac{q_{1_y}^{h_2}-input_y^{h_2}}{2dist(input^{h_2},q_1^{h_2})}\right]$$
$$\left[\frac{\left(-p_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

$$\frac{\partial err}{\partial q_{1_z}^{h_2}} = \frac{-2g_{out}}{\left(1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}\right)^2}\left[y - \frac{1}{1+e^{-g_{out}T(input^{out},\mathcal{L}^{out})}}\right]e^{-g_{out}T(input^{out},\mathcal{L}^{out})}$$
$$\left(input_y^{out}\right)^2 e^{-g_{h2}T(input^{h_2},\mathcal{L}^{h_2})}g_{h2}\left[-\frac{p_{1_z}^{h_2}-q_{1_z}^{h_2}}{2dist(p_1^{h_2},q_1^{h_2})} - \frac{q_{1_z}^{h_2}-input_z^{h_2}}{2dist(input^{h_2},q_1^{h_2})}\right]$$
$$\left[\frac{\left(-p_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_0^{out})} + \frac{\left(-q_{0_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_0^{out})} - \frac{\left(-p_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},p_1^{out})} - \frac{\left(-q_{1_y}^{out}+input_y^{out}\right)}{2dist(input^{out},q_1^{out})}\right]$$

Referências

[1] Sanjay; Alsabti, Khaled; Ranka and Vineet Singh. An efficient k-means clustering algorithm. *Electrical Engineering and Computer Science*, (43), 1997.

[2] Jakob J. Verbeek Aristidis Likas, Nikos Vlassis. The global k-means clustering algorithm. *Pattern Recognition*, 36:451–461, March 2003.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[4] S.S. Haykin. *Neural Networks: A Comprehensive Foundation*. International edition. Prentice Hall, 1999.

[5] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. http://neuralnetworksanddeeplearning.com.

[6] S. Raschka and V. Mirjalili. *Python Machine Learning*. Packt Publishing, 2017.

[7] J. H. B. Ribeiro. *Aprendizado Computacional Baseado em Distância a Segmentos de Reta*. Doutorado, IME-USP, 2009.

[8] J.H.B. Ribeiro and R.F. Hashimoto. A new training algorithm for pattern recognition technique based on straight line segments. In *Computer Graphics and Image Processing, 2008. SIBGRAPI '08. XXI Brazilian Symposium on*, pages 19 –26, 12-15 2008.

[9] J.H.B. Ribeiro and R.F. Hashimoto. *Pattern Recognition, Recent Advances*, chapter Pattern Recognition Based on Straight Line Segments, pages 167–188. I-Tech, 2010. Book Chapter.

[10] Joao H. Burckas Ribeiro and Ronaldo F. Hashimoto. A New Machine Learning Technique Based on Straight Line Segments. In *ICMLA '06: Proceedings of the 5th International Conference on Machine Learning and Applications*, pages 10–16, Washington, DC, USA, 2006. IEEE Computer Society.

[11] R.A. Medina Rodriguez and R.F. Hashimoto. Combining dialectical optimization and gradient descent methods for improving the accuracy of straight line segment classifiers. In *Sibgrapi 2011. 24 Conference on Graphics, Patterns and Images.*, pages 321–328. IEEE, august 2011.

[12] Rosario Alejandra Medina Rodriguez. Algoritmos evolutivos aplicados ao classificador baseado em segmentos de reta. Master's thesis, Instituto de Matemática e Estatística - Universidade de São Paulo, 2012.