Redes Neurais Artificiais: Revisão da Literatura

Victor São Paulo Ruela
Programa de Pós-Graduação em Engenharia Elétrica
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
Email: victorspruela@ufmg.br

***Resumo*— Este trabalho tem como objetivo apresentar uma revisão da literatura de redes neurais aritificiais, com enfoque nas evoluções desenvolvidas a partir dos principais trabalhos clássicos, os que estabeleceram os fundamentos desta grande área de pesquisa.**

I. Introdução

A Rede Neural Artificial (RNA) é uma classe de modelos muito popular em problemas de classificação, reconhecimento de padrões, regressão e predição [1]. Inspirado pelas características do cérebro humano, elas possuem como elementos básicos neurônios artificiais capazes de executar operações matemáticas, representando desta forma modelos de neurônios biológicos. Através de sua organização em diferentes estruturas de rede, tais modelos são capazes de se adaptar e representar funções matemáticas bastante complexas.

Diferentes representações estão presentes na literatura, as quais são classificadas de acordo com o seu nível de complexidade e requisitos coputacionais de implementação. Hipóteses básicas para regras de aprendizado de associações entre neurônios podem ser encontradas em trabalhos bastante antigos, como abordado no livro de William James em 1982 [2]. Entretanto, um grande marco desta área de pesquisa ocorreu na década de 40 após a introdução do modelo de McCulloch and Pitts (MCP) [3], o qual é a adotado atualmente nos principais modelos de RNAs.

O modelo MCP tem como saída a soma das ativações dos neurônios anteriores ponderados pelos pesos das conexões entre eles. Originalmente, uma função de ativação do tipo degrau é aplicada sobre sua saída, configurando modelo de soma-e-limiar originalmente descrito pelos autores. Este trabalho apresentou a configuração de diversas redes de neurônios MCP, com enfoque na implementação de funções lógicas. Vale a pena notar que os primeiros computadores digitais estavam surgindo nesta época, motivando esta aplicação. Entretanto, as estruturas apresentadas eram estáticas e não houve a sugestão de algum método de aprendizado para adaptá-las.

O aprendizado surgiu de forma mais concreta com o postulado de Hebb [4], originalmente publicado em 1949. De acordo com o autor, a eficiência de uma determinada sinapse que conecta dois neurônios é proporcional à co-ocorrência de ativação entre eles. Portanto, o princípio de aprendizado Hebbiano visa reforçar as conexões relevantes para as diferentes saída da rede, guiado pela correlação entre os neurônios. Considerando o neurônimo MPC, suas primeiras estruturas de rede e algoritmos de treinamento descritos na literatura são o *Adaline* [5], em 1960, e o Perceptron simples, em 1957 [6].

Após um período de euforia com a introdução destes últimos dois modelos, a área de pesquisa de RNAs sofreu um descrédito e frustação até o início dos anos 80. Isso decorreu do trabalho de Minsky e Papert [7], o qual generalizou as limitações destes modelos para problemas considerados fundamentais, como o do OU-exclusivo (XOR). O interesse só foi reativado após o re-descobrimento do algoritmo *back-propagation* para treinamento de redes de múltiplas camadas [8], as quais são capazes de superar as limitações até então existentes das redes de camada única. Destacam-se também a introdução dos mapas de Kohonen [9], redes recorrentes de Hopfield [10] e o modelo ART para aprendizado não-supervisionado [11]. Além disso, nesta época surgiram as primeiras conferências e periódicos dedicados à area de RNAs [12].

A partir destes princípios elementares, a área de RNAs evoluiu bastante nas últimas décadas. Após a introdução das primeiras regras de aprendizado, este tipo de modelo ganhou maior visibilidade e aplicabilidade para problemas reais, sendo possível encontrar uma enorme quantidade de aplicações publicadas [13]. Além disso, o aumento dos recursos computacionais disponíveis fomentou o desenvolvimento de novas técnicas para aprendizado e o aprimoramento das existentes, além de propostas de novas estruturas redes complexas capazes de lidar com problemas de grande dificuldade.

Portanto, o objetivo deste trabalho é apresentar a uma revisão da literatura contendo os principais trabalhos e entender a evolução dos diferentes modelos de redes neurais utilizadas atualmente. Partindo das referências clássicas, diferentes abordagens propostas serão analisadas de forma cronológica com o intuito de se entender a evolução desta área de pesquisa até o tempo presente. Este trabalho será divido da seguinte forma: a Seção II apresenta uma revisão da literatura com uma análise crítica dos principais trabalhos, os quais serão organizados usando como referência o livro de Simon Haykin [14] e as notas de aula. Na seção III serão apresentadas algumas das principais aplicações de RNAs publicadas. Finalmente, é feita uma conclusão deste trabalho.

II. Revisão da Literatura

A. Aprendizado Hebbiano

B. Perceptron

Proposto inicialmente por Rosenblatt [6], este é um modelo geralmente utilizado para a solução de problemas de classificação. No seu trabalho original, o autor descreve formas de adaptação dos parâmetros, ou pesos, da rede com o objetivo de reduzir a discrepância entre as saídas esperadas e estimadas e aprender associações entre os neurônios, o que é a base da indução para diversos algoritmos atuais. Este trabalho é considerado um marco na literatura por diversos autores. Embora descrito como uma rede de duas camadas, originalmente seu treinamento só considerava uma camada. Por esse motivo, o Perceptron simples é comumente descrito na forma de somente um neurônio MCP. Sua regra de aprendizado é bem direta e consiste em alterar iterativamente os pesos da rede adicionado o erro total entre as saídas medidas estimadas ponderada pelo vetor de entradas [14].

Se considerarmos uma função de ativação contínua e diferenciável, os pesos da rede poderão ser inferidos de forma explícita, através do cálculo da pseudo-inversa, ou pelo algoritmo do gradiente descendente [15]. Exemplos de funções de ativação com esta característica frequentemente empregadas na literatura são a função logística, tangente hiperbólica e linear [1]. Vale a pena ressaltar que a covergência destas abordagens está condicionada aos dados utilizados para treinamento serem linearmente independentes [15].

Rosenblatt provou a convergência da regra de aprendizado original, porém a mesma só é garantida para problemas linearmente separáveis, o que constitui a principal limitação deste modelo. O trabalho de Minsky e Papert [7] evidenciou bastante esta limitação e através da aplicação do Perceptron a diversos problemas considerados fundamentais, levou ao descrédito deste modelo pela comunidade científica. Após este trabalho, Rosenblatt avaliou diferentes arquiteturas de rede tentando superar esta limitação, mas não conseguiu chegar ao desenvolvimento do aprendizado para múltiplas camadas. Por conta disso, o Perceptron foi pouco estudado pelos próximos de 20 anos [15].

O interesse pelo Perceptron retornou na década de 80 com a descrição do método de aprendizado conhecido como *back-propagation*, o qual é capaz de aprender os pesos de redes de múltiplas camadas de forma eficiente [8]. Aliado a isso, o Perceptron de múltiplas camadas é capaz de descrever superfícies de separação não-lineares, superando a pricipal limitação do trabalho de Rosenblatt. Uma descrição mais completa desta família de modelos é feita na próxima seção.

C. Adaline

O Adaline foi inicialmente desenvolvido por Widrow em 1960 [5], sendo principamente aplicado em problemas de regressão lineares. Assim como o Perceptron, originalmente este modelo considera somente um neurônio MCP em sua formulação, entretando sua função de ativação é a identidade. Seu treinamento é formulado como um problema de otimização com custo quadrático, onde originalmente foi utilizado o algoritmo do gradiente descendente para sua solução.

Para este agoritmo, em cada iteração é dado um passo na direção oposta ao gradiente da função objetivo, resultanto em uma convergência gradual para o mínimo do problema. Este gradiente pode ser calculado de forma análitica para a estrutura de rede do Adaline, o qual é no fim proporcional à diferença entre os valores estimados e reais [5], bastante similar ao Perceptron simples. É fácil notar que o treinamento também pode ser realizado de forma direta através do cálculo da pseudo-inversa dos dados de entrada, já que este é um problema de mínimos quadrados [14].

Uma extensão proposta deste modelo é conhecida como Madaline, o qual é caracterizada por uma rede composta por vários Adalines. Existem duas principais regras para seu treinamento, conhecidas por MRI [16] and MRII [17]. É interessante notar que a MRI surgiu bem antes do algoritmo *back-propagation*, podendo ser considerada uma estrutura primitiva de uma rede de múltiplas camadas. Os leitores são referidos à [18] para uma descrição mais completa destas regras e suas aplicações.

D. Perceptron de múltiplas camadas

O Perceptron de múltiplas camadas (MLP) é uma rede neural com uma ou mais camadas escondidas, ou seja, localizadas entre as entradas e saídas do modelo. Além disso, são caracterizadas por um alto grau de conectividade e por aplicar funções de ativação não-lineares e diferenciáveis ao modelo dos neurônios [14]. Estas camadas adicionais funcionam como detectores de características, aplicando transformações não-lineares sequenciais aos dados de forma que estes sejam mais facilmente separados nesse novo espaço. Portanto, a introdução das camadas escondidas permite modelar superfícies de decisão não-lineares, sendo consideradas aproximadores universais de funções se a função de ativação é contínua, limitada e não-constante [19].

O treinamento de redes de camada única vistas nas seções anteriores é bem direto pois podemos facilmente derivá-las analiticamente. Entretando, ao incluir camadas escondidas e alta conectividade, analisar teoricamente o comportamento do Perceptron torna-se mais difícil. Aliado a isso, o seu treinamento se torna mais complexo justamente por haver uma maior quantidade de estruturas de rede possíveis para representar os dados de entrada. O primeiro algoitmo eficiente de treinamento de tais foi formalizada por Rumelhart em 1985 [8], conhecido como *back-propagation*.

O *back-propagation* é uma técnica de aprendizado online (ou estocástica), na qual os pesos da rede são ajustados amostra-a-amostra. Ou seja, em cada época de treinamento, os dados de entrada são apresentados individualmente para a rede objetivando a minimização do erro das saídas estimadas e desejadas. O algoritmo pode ser dividido em duas etapas: na primeira os dados são apresentados à rede mantendo os pesos fixos, e calculada a sua respectiva saída; na segunda, o sinal de erro em relação à saída esperada é calculado e propagado no

sentido inverso da rede, onde ajustes sucessivos são realizados. A atualização dos pesos da rede é feita com base na técnica do gradiente descendente, cuja derivação completa será omitida deste trabalho por brevidade. O leitor pode encontrá-la em [8], [14].

Embora bastante eficiente, sua convergência pode ser lenta se o algoritmo não for usado corretamente [20]. É recomendado realizar a normalização das entradas para equalizar a taxa de atualização dos pesos entre as camadas, além de remover variávies altamente correlacionadas [21]. Ainda de acordo com [21], é sugerido o uso de sigmoides simétricas, como a tangente hiperbólica, as quais geralmente possuem maior velocidade de convergência. Além disso, é importante que os valores desejados estejam dentro dos limites da sigmoide escolhida. Outro fator importante é a taxa de treinamento: [22] apresenta o estudo de algumas técnicas para adaptação da taxa de treinamento presentes na literatura, mostrando que o seu uso é bastante benéfico.

É interessante notar que os problemas descritos anteriormente são similiares aos encontrados para a otimização de funções não-lineares e não-convexas. Ou seja, é possível analisar o treinamento do MLP como um problema de otimização e aplicar diferentes algoritmos e heurísticas disponíveis da literatura. De fato, isso é explorado por diversos autores, os quais aplicam o método de Newton [23], gradiente conjugado [24], Gauss-Newton [25], Levenberg-Marquardt [26] e Quasi-Newton [27].

Métodos de segunda ordem possuem o atrativo da convergência acelarada, mas em contra-partida é necessária a estimativa da Hessiana, o que exige mais recursos computacionais e está sujeito a problemas numéricos adicionais, sendo limitado a redes pequenas e a usar usando aprendizado por batelada. Portanto, este são fatores que devem ser levados em conta na hora de escolher o otimizador para realizar o treinamento de uma rede. Conforme argumentado por [21], o uso de informações de segunda ordem nem sempre é necessário em alguns problemas, para os quais a técnica do gradiente estocástico bem ajustada é dificilmente superado para problemas de larga escala.

O estado da arte para treinamento de RNAs são o algoritmo Adadelta [28] e o Adam [29], impulsionado pelo crescimento da popularidade das redes de aprendizado profundo.

Mais recentementes, alguns autores começaram a propor o uso de algoritmos evolucionários (EAs) para o treinamento do MLP. Eles são atrativos pelo fato de convergirem para o ótimo global se um tempo de treinamento suficiente estiver disponível. Entretanto, o custo computacional poderá ser bastante alto, além do ajuste dos hiper-parâmetros ser trabalhoso. Em [30], os autores propõe o uso do algotimo Differential Evolution, o qual concluem que ele não apresenta desempenho superior ao *back-propagation*. Já [31] propôs uma técnica híbrida de algoritmos genéticos e backpropagation, a qual se mostrou menos suscetível a ficar presa em mínimos locais durante o treinamento. O leitor pode ser referir a [32] para um estudo mais completo de EAs e RNAs. Outro área interessante são as redes neurais evolutivas [33], para as quais além do treinamento dos pesos evolui-se também outras características da rede.

E. Redes de funções de base radial (RBF)

As redes RBF foram inicialmente introduzidas por [34] e são caracterizadas por um aprendizado que envolve duas etapas: (i) aplicar uma transformação aos padrões para um espaço onde a expecativa de serem linearmente separáveis é alta (ii) encontrar os pesos usando o estimador mínimos quadrados usado no Perceptron simples. Essa estrutura pode ser representada por um rede de três camadas, onde sua camada escondida é reponsável pela transformação não-linear das estradas para o novo espaço, geralmente para uma dimensão muito alta.

Essa transformação é justificada pelo teorema de Cover sobre a separabilidade de padrões [35], o qual diz que um problema de classificação complexo projetado não-linearmente para um espaço de alta dimensão é mais provável de ser separável do que em um espaço de baixa dimensão, desde que o espaço não seja densamente povoado. Boa parte da teoria, que é relacionada ao campo de interpolação multivariável, considera um kernel baseado na função Gaussiana, que é uma classe importante de RBFs. Teoricamente, as redes RBF podem ser consideradas um aproximador universal de funções contínuas se a RBF é selecionada apropriadamente [36], [37], [38]. Uma condição apropriada é dada pelo teorema da interpolação de Micchelli [39], ou quando uma determinada classe de RBFs é contínua e diferenciável [37], por exemplo.

No seu treinamento, além de tratamentos especiais na presença de ruídos nos dados [40], uma etapa importante é a estimativa dos parâmetros das unidades Gaussianas além dos pesos da rede. Isso pode ser feito distribuindo os centros uniformemente, por exemplo. De forma mais geral, pode-se selecionar aleatoriamente subconjutos dos padrões de entrada se este são representativos e grandes o suficiente para o apredizado [41]. Para aproximação de funções, uma heurística possível é colocar os centros nos extremos da derivada de segundo grau da função e também mais densamente em áreas cujo valor absoluto e maior [42], o que possui resultados melhores em relação à distribuição uniforme.

Outra abordagem bastante utilizada para a definição dos centros é o uso de técnicas de agrupamento (*clustering*) dos dados presentes na literatura [43], [44]. Uma escolha popular são o algoritmo não-supervisionado *k-means* [45] e redes neurais baseadas em memórias associativas [46]. Também podem ser utilizadas técnicas de clustering supervisionadas [47], as quais podem ser mais eficientes para redes RBF [41]. Após determinados os centros, define-se as matrizes de convariância das RBFs como a convariância dos dados de cada *cluster* [41].

Com os centros e covariâncias definidos, o aprendizado dos pesos é feito pela minimização do erro quadrático médio. Para problemas simples, pode-se utilizar o algoritmo de mínimos quadrados ou gradiente descendente, similar ao aprendizado do Perceptron simples. O treinamento via gradiente descendente é equivalente ao do MLP [48], onde pode-se também utilizar

as mesmas abordagens por otimização irrestrita [43]. Uma descrição de diferentes abordagens para treinamento de RBFs por otimizaçao podem ser encontradas em [41], como o uso de *back-propagation* seletivo [49] e programação linear [50].

No caso dos mínimos quadrados, problemas complexos que requerem um número elevado de RBFs na camada escondida estão sujeitos a problemas numéricos na estimativa da pseudo-inversa. Para lidar com isso, uma abordagem comum é o uso de técnicas de ortogonalização, como as decomposições SVD e QR [51]. Em algumas condições, é possível também resolver esse problema usando a transformada de Fourier da rede RBF, a qual requer menos esforço computacional necessário [52]. Outra alternativa eficiente é aplicar a ortogonalização de Gram-Schimidt (GSO) à RBF [53], a qual possui menor requisito de armazenamento e pode sem implementado de forma paralela.

Outra forma eficiente para aprendizado da rede consiste no uso da técnica de mínimos quadrados ortogonal (OLS) [54], [55], [56], [57]. Nesta abordagem, não só os pesos podem ser determinados, mas também a quantidade e centros das RBFs. o GSO é inicialmente aplicado para contruir um conjunto de vetores ortogonais no espaço criado pela unidade escondida, e na sequência um novo centro RBF é selecionado de acordo com a sua minimização do erro quadrático médio de treinamento. Esse algotimo pode ser executado de forma sequencial ou reversa, ou seja, podemos iniciar com uma rede vazia ou com o máximo de unidades escondidas, por exemplo. Existem também versões recursivas deste algoritmo na literatura, conhecidas por ROLS [58], as quais são aplicadas a problemas envolvendo sistemas de múltiplas entradas e saídas, por exemplo.

F. Generalização

Uma das suas principais características do modelo RNA é sua habilidade de generalização, que é sua capacidade em estimar corretamente dados ainda não vistos. Em outras palavras, algoritmos de aprendizado de RNA possuem como objetivo encontrar um modelo que consigar caputar as principais características do conjunto de dados de treinamento, mas que também seja capaz de prever com precisão um conjunto de teste ainda não visto.

Este conceito é introduzido pela primeira vez por Geman et. al em 1992 [59], o qual é o bastante conhecido dilema viés-variância. Através deste trabalho, os autores introduzem o conceito de que existe uma competição entre duas fontes de erro que impede o modelo de generalizar além dos dados de treinamento, o viés e a variância. Além disso, eles mostra que existe um compromisso entre estas grandezas, de forma que quando um modelo com viés máximo possuirá mínima variância, e vice-versa.

O erro de viés está relacionado a suposições incorretas sobre o modelo, podendo ser defindo como a diferença entre o valor previsto médio e o esperado. Um alto viés leva ao fenômeno de *under-fitting*, onde o modelo possuem uma estrtura simples e não é capaz de representar os principais padrões presentes nos dados. Já o erro de variância indica a sensibilidade do modelo a flutuações no conjunto de treinamento. Uma variância elevada leva ao fenômeno de *over-fitting*, onde o modelo é bastante complexo que se ajusta perfeitamente aos dados de treinamento. Um exemplo deste comportamente pode ser visto na Figura 1. É importante notar que estes dois erros podem contribuir para um desempnho ruim, ou seja, mesmo um modelo não-enviésado com variância alta pode resultar em um erro de treinamento maior que o esperado [59].

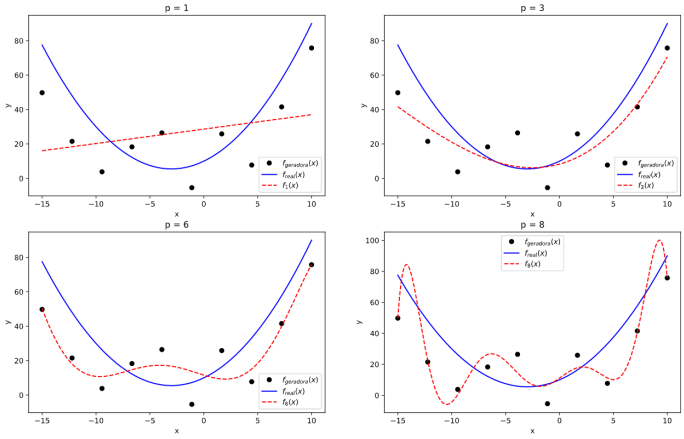

Fig. 1. Exemplo de *under-fitting* ($p = 1$) e *over-fitting* ($p > 3$) para problema de regressão polinomial sobre dados uma função quadrática.

Existem algumas formas de se balancear o viés e variância, as são divididas no caso da conjuntos amostras finitas e asintóticamente infinitas. Por serem mais comuns em aplicações práticas, as últimas serão omitidas deste trabalho e o autor pode se referir a [59] para mais detalhes. Para conjuntos de dados finitos, destacam-se a validação cruzada [60] e o uso de regularização pela inclusão de penalidades à função objetivo do treinamento [61].

O conceito de generalização deu origem a técnicas de aprendizado que incluem em sua formulação formas de se balancear o viés e a variância automaticamente. Entre eles, destacam-se as máquinas de vetores suporte (SVM), máquinas de aprendizado extrem (ELM) e o aprendizado multiobjetivo.

G. Máquinas de aprendizado extremo

Inicialmente proposto por [62], as máquinas de aprendizado extremo (ELM) são redes neurais *feed-forward* com uma única camada escondida, as quais possuem o atrativo de poucos parâmetros a serem ajustados, generalização maior ou similar e redução do tempo de treinamento das redes em relação aos métodos convencionais. Seu treinamento é baseado na teoria de minimização de risco empírico, necessitando de somente uma iteração para este processo, evitando múltiplas iterações e algoritmos de otimização local [63].

ELMs são capazes de definir adaptivamente o número neurônios da rede e aleatoriamente escolher os pesos das entradas e viéses da camada escondida [64]. Isso faz com o que a rede possa ser considerada como um sistema linear, o qual pode ser calculado de forma analítica através de uma operação de inversão da matriz de pesos da camada de saídas [64].

Essa característica permite uma drástica redução do esforço computacional do treinamento, geralmente de 10 vezes ou mais [65].

Apesar do apelo no ganho de tempo de treinamento, essa abordagem permite menor adaptabilidade ao conjunto de dados, além de problemas numéricos para o cálculo da inversão por mínimos quadrados e problemas de robustez se os dados forem ruidosos [63]. Além disso, a aleatoriedade na escolha de pesos e viéses pode levar a uma quantidade maior de neurônios na camada escondida, bem como tornar o sistema linear não solucionável e reduzir sua acurácia [66].

Embora tenha sido proposto bem recentemente (2004), têm recebido uma atenção considerável na literatura a nível téorico e de aplicações [67]. Do ponto de vista da teoria, o foco dos autores é divido entre duas frentes: (1) aprimorar sua capacidade universal de aproximação com camadas escondidas aleatórios e (2) propor novas técnicas de aprendizado mais rápidas e robustas. De acordo com uma revisão feita em 2011 por Huang et. al [67], destacam-se o ELM baseado em kernel [68], ELM com aprendizado online [69], [70], ELM incremental [71], *ensemble* de ELM [72], [73], [74] e ELM com poda [75].

H. Máquinas de vetores suporte

Introduzido por Vapnik em 1992 [76] ás máquinas de vetores suporte (SVM) são considerados um dos algoritmos mais robustos e poderosos para aprendizado supervisionado até os dias atuais. No trabalho de Vapnik, o autor apresenta de forma bem completa os fundamentos teóricos deste modelo, apresentando justificativas para sua ótima capacidade de generalização, bem como os limites para a sua validade. Seu princípio de treinamento está na maximização da margem entre os padrões de treino e a superfície de decisão, que é uma representação convexa do compromisso entre viés e variância.

Estas propriedades são uma consequência do uso dos chamados vetores de suporte no seu treinamento, que são os subconjuntos de dados mais próximos da superfície de decisão, ou seja, os mais difícies de classificar e relevantes para sua otimalidade. O problema então consiste em encontrar o hiperplano que maximiza a margem de separação entre os padrões, que é equivalente a minimizar a norma Euclidiana do seu vetor de pesos [14]. Isso pode ser visto graficamente na Figura 2.

O problema de encontrar o vetor de pesos que define o hiper-plano foi convenientemente formulado com base na teoria de otimização convexa. Isso é muito importante pois esta classe de problemas tem sua otimalidade garantida e bem definida, não estando sujeita aos problemas observados no algoritmo *back-propagation*, por exemplo. A função objetivo é descrita como a norma Euclideana dos pesos somada acrescida um segundo termo que limita a quantidade de erros de treinamento, já que os dados podem não ser perfeitamente serparáveis por um hiper-plano. Este problema é então transformado utilizando a técnica de multiplicadores de Lagrange, a qual faz com que seja descrito em função somente dos dados de treinamento.

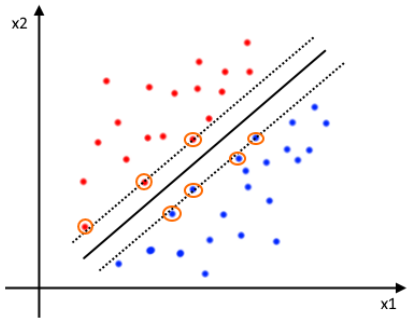

Fig. 2. Elementos principais da otimização pela maximização da margem. Para um problema de classificação binário, é encontrado o hiper-plano separador (linha sólida) que maximiza a distância dos vetores de suporte (pontos marcados em laranja). As linhas pretas pontilhadas representam os hiper-planos de suporte.

O segundo termo pode ser ajustado pelo usuário por meio de uma constante, a qual irá controlar o nível de regularização aplicada ao problema, o que resulta em sua característica de generalização. Um valor alto poderá resultar em *overfitting* se os dados forem ruidosos, o que sugere cuidado na sua escolha. Sua definição pode ser feita experimentalmente usando técnicas de validação cruzada [60], por exemplo.

Através da discussão anterior, é fácil notar que o SVM é aplicável somente a problemas linearmente separáveis se são for realizada nenhuma transformação não-linear aos dados. Portanto, um conceito importante utilizado por este algoritmo são os *kernels* [77]. A partir da escolha de um kernel apropriado [78], [79], o problema poderá ser resolvido por um modelo linear. Note que isto é bem similiar ao princípio de funcionamento das redes RBF. Em relação ao MLP, ele possui a vantagem de controlar a complexidade independente de sua dimensionalidade, ou seja, é possível ter uma quantidade muito grade de neurônios na camada escondida, conforme o teorema de Cover [35], e ainda assim obter uma boa generalização [80]. Um diagrama mostrando o algotimo SVM completo pode ser visto na Figura 3.

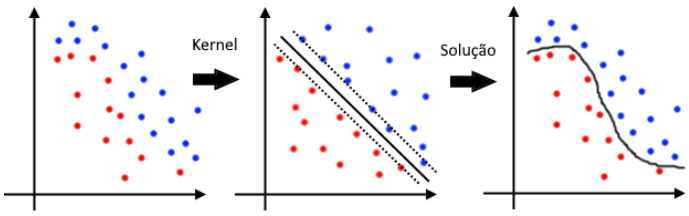

Fig. 3. Diagrama do algoritmo SVM

Embora muito poderoso, esse algoritmo é caracterizado por um elevado custo computacional de sua implementação [81]. Como sua complexidade pode ser proporcional ao quadrado da quantidade de amostras de treinamento, sua aplicação em problemas de grande porte pode se tornar proibitiva. Outra limitação está relacionada à solução do problema de otimização, uma vez que solvers disponíveis possuem suas próprias limitação em relação ao número de variáveis de

decisão suportadas. O problema de otimização é ainda mais dificultado pela esparsidade da solução do SVM, pois o vetor de pesos a ser encontrado possuirá poucos elementos não-nulos [14], resultando em problemas númericos que impedirão sua correta solução. Diversas melhorias foram propostas para melhorar esta situação, as quais são classificadas entre seleção de dados [82], decomposição [83], geometria [84], implementação paralela [85] e heurísticas [86].

Outra limitação deste algoritmo está relacionada a dados não-balanceados. Técnicas propostas para lidar com essa situação realizam o balanceamento dos dados antes do treinamento, ou modificam a estrutura do modelo para torná-los menos sensitivos. As técnicas mais comuns são o *under* e *over-sampling*, e o SMOTE [87]. Esta última é mais indicada para SVMs [88]. É importante ressaltar também que o SVM foi originalmente formulado problemas de classificação binária, de forma que este precisou de ser posteriormente extendido para ser aplicação à essa classe de problemas [89], [90].

Em [91], os autores apresentam um revisão bem completa da literatura de SVMs para classificação. Embora utilizado com sucesso em diversas aplicaçõos reais, o SVM ainda possui pouca adoção para problemas com grandes volumes de dados, bem como baixo desempenho para dados não-balanceados. Como tendências de trabalhos futuros, os autores citam além de melhorias nestes problemas clássicos, avanções em treinamento on-line, seleção automática de kernel e parâmetros com menor custo, além de aplicações para aprendizado semi-supervisionado.

I. Aprendizado multiobjetivo

J. SOM

III. Aplicações

IV. Conclusões

Referências

[1] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[2] William James. *Psychology, briefer course*, volume 14. Harvard University Press, 1984.

[3] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[4] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[5] Bernard Widrow and Marcian E Hoff. Adaptive switching circuits. Technical report, Stanford Univ Ca Stanford Electronics Labs, 1960.

[6] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[7] Marvin Minsky and Seymour Papert. An introduction to computational geometry. *Cambridge tiass., HIT*, 1969.

[8] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[9] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

[10] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[11] Gail A Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer vision, graphics, and image processing*, 37(1):54–115, 1987.

[12] Bohdan Macukow. Neural networks–state of art, brief history, basic models and architecture. In *IFIP international conference on computer information systems and industrial management*, pages 3–14. Springer, 2016.

[13] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.

[14] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice-Hall, Inc., 2007.

[15] John Hertz, Anders Krogh, Richard G Palmer, and Heinz Horner. Introduction to the theory of neural computation. *PhT*, 44(12):70, 1991.

[16] Bernard Widrow. Generalization and information storage in network of adaline'neurons'. *Self-organizing systems-1962*, pages 435–462, 1962.

[17] Capt Rodney Winter and B Widrow. Madaline rule ii: A training algorithm for neural networks. In *Secondf Annual International Conference on Neural Networks*, pages 1–401, 1988.

[18] Bernard Widrow and Michael A Lehr. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, 1990.

[19] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[20] Yann LeCun. Efficient learning and second-order methods. *A tutorial at NIPS*, 93:61, 1993.

[21] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[22] George D. Magoulas, Michael N. Vrahatis, and George S Androulakis. Improving the convergence of the backpropagation algorithm using learning rate adaptation methods. *Neural Computation*, 11(7):1769–1796, 1999.

[23] Sue Becker, Yann Le Cun, et al. Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the 1988 connectionist models summer school*, pages 29–37, 1988.

[24] Erik M Johansson, Farid U Dowla, and Dennis M Goodman. Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method. *International Journal of Neural Systems*, 2(04):291–301, 1991.

[25] Roberto Battiti. First-and second-order methods for learning: between steepest descent and newton's method. *Neural computation*, 4(2):141–166, 1992.

[26] Martin T Hagan and Mohammad B Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE transactions on Neural Networks*, 5(6):989–993, 1994.

[27] B Robitaille, B Marcos, M Veillette, and G Payre. Modified quasi-newton methods for training neural networks. *Computers & chemical engineering*, 20(9):1133–1140, 1996.

[28] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Jarmo Ilonen, Joni-Kristian Kamarainen, and Jouni Lampinen. Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*, 17(1):93–105, 2003.

[31] Shifei Ding, Chunyang Su, and Junzhao Yu. An optimizing bp neural network algorithm based on genetic algorithm. *Artificial intelligence review*, 36(2):153–162, 2011.

[32] Seyedali Mirjalili. Evolutionary algorithms and neural networks. *Studies in Computational Intelligence*, 2019.

[33] Xin Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[34] D. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Syst.*, 2, 1988.

[35] Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, (3):326–334, 1965.

[36] Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[37] Jooyoung Park and Irwin W Sandberg. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257, 1991.

[38] Yi Liao, Shu-Cherng Fang, and Henry LW Nuttle. Relaxed conditions for radial-basis function networks to be universal approximators. *Neural Networks*, 16(7):1019–1028, 2003.

[39] Charles A Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive approximation*, 2(1):11–22, 1986.

[40] David S Broomhead and David Lowe. Radial basis functions, multivariable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[41] Yue Wu, Hui Wang, Biaobiao Zhang, and K-L Du. Using radial basis function networks for function approximation and classification. *ISRN Applied Mathematics*, 2012, 2012.

[42] V David Sánchez A. Second derivative dependent placement of rbf centers. *Neurocomputing*, 7(3):311–317, 1995.

[43] Ke-Lin Du and Madisetti NS Swamy. *Neural networks in a softcomputing framework*. Springer Science & Business Media, 2006.

[44] K-L Du. Clustering: A neural network approach. *Neural networks*, 23(1):89–107, 2010.

[45] John Moody and Christian J Darken. Fast learning in networks of locally-tuned processing units. *Neural computation*, 1(2):281–294, 1989.

[46] Teuvo Kohonen. *Self-organization and associative memory*, volume 8. Springer Science & Business Media, 2012.

[47] C-L Chen, W-C Chen, and F-Y Chang. Hybrid learning algorithm for gaussian potential function networks. In *IEE Proceedings D (Control Theory and Applications)*, volume 140, pages 442–448. IET, 1993.

[48] Dietrich Wettschereck and Thomas Dietterich. Improving the performance of radial basis function networks by learning center locations. In *Advances in neural information processing systems*, pages 1133–1140, 1992.

[49] Mohammad-Taghi Vakil-Baghmisheh and Nikola Pavešić. Training rbf networks with selective backpropagation. *Neurocomputing*, 62:39–64, 2004.

[50] Asim Roy, Sandeep Govil, and Raymond Miranda. An algorithm to generate radial basis function (rbf)-like nets for classification problems. *Neural networks*, 8(2):179–201, 1995.

[51] Gene H Golub and Charles F Van Loan. An analysis of the total least squares problem. *SIAM journal on numerical analysis*, 17(6):883–893, 1980.

[52] Yoshinori Abe and Y Figuni. Fast computation of rbf coefficients for regularly sampled inputs. *Electronics Letters*, 39(6):543–544, 2003.

[53] Wladyslaw Kaminski and Pawel Strumillo. Kernel orthonormalization in radial basis function neural networks. *IEEE Transactions on Neural Networks*, 8(5):1177–1183, 1997.

[54] Sheng Chen, Colin FN Cowan, and Peter M Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on neural networks*, 2(2):302–309, 1991.

[55] S Chen, PM Grant, and CFN Cowan. Orthogonal least-squares algorithm for training multioutput radial basis function networks. In *IEE Proceedings F (Radar and Signal Processing)*, volume 139, pages 378–384. IET, 1992.

[56] S Chen and J Wigger. Fast orthogonal least squares algorithm for efficient subset model selection. *IEEE Transactions on Signal Processing*, 43(7):1713–1715, 1995.

[57] Xia Hong and SA Billings. Givens rotation based fast backward elimination algorithm for rbf neural network pruning. *IEE Proceedings-Control Theory and Applications*, 144(5):381–384, 1997.

[58] DL Yu, JB Gomm, and D Williams. A recursive orthogonal least squares algorithm for training rbf networks. *Neural Processing Letters*, 5(3):167–176, 1997.

[59] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

[60] Sylvain Arlot, Alain Celisse, et al. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

[61] Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization theory and neural networks architectures. *Neural computation*, 7(2):219–269, 1995.

[62] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)*, volume 2, pages 985–990. IEEE, 2004.

[63] Shifei Ding, Han Zhao, Yanan Zhang, Xinzheng Xu, and Ru Nie. Extreme learning machine: algorithm, theory and applications. *Artificial Intelligence Review*, 44(1):103–115, 2015.

[64] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[65] Wan-Yu Deng, Qing-Hua Zheng, Lin Chen, and Xue-Bin Xu. Research on extreme learning of neural networks. *Chinese Journal of Computers*, 33(2):279–287, 2010.

[66] Yuguang Wang, Feilong Cao, and Yubo Yuan. A study on effectiveness of extreme learning machine. *Neurocomputing*, 74(16):2483–2490, 2011.

[67] Guang-Bin Huang, Dian Hui Wang, and Yuan Lan. Extreme learning machines: a survey. *International journal of machine learning and cybernetics*, 2(2):107–122, 2011.

[68] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2):513–529, 2011.

[69] Nan-Ying Liang, Guang-Bin Huang, Paramasivan Saratchandran, and Narasimhan Sundararajan. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on neural networks*, 17(6):1411–1423, 2006.

[70] Senyue Zhang, Wenan Tan, and Yibo Li. A survey of online sequential extreme learning machine. In *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 45–50. IEEE, 2018.

[71] Guang-Bin Huang and Lei Chen. Enhanced random search based incremental extreme learning machine. *Neurocomputing*, 71(16-18):3460–3468, 2008.

[72] Zhan-Li Sun, Tsan-Ming Choi, Kin-Fan Au, and Yong Yu. Sales forecasting using extreme learning machine with applications in fashion retailing. *Decision Support Systems*, 46(1):411–419, 2008.

[73] Mark Van Heeswijk, Yoan Miche, Tiina Lindh-Knuutila, Peter AJ Hilbers, Timo Honkela, Erkki Oja, and Amaury Lendasse. Adaptive ensemble models of extreme learning machines for time series prediction. In *International Conference on Artificial Neural Networks*, pages 305–314. Springer, 2009.

[74] Yuan Lan, Yeng Chai Soh, and Guang-Bin Huang. Ensemble of online sequential extreme learning machine. *Neurocomputing*, 72(13-15):3391–3395, 2009.

[75] Hai-Jun Rong, Yew-Soon Ong, Ah-Hwee Tan, and Zexuan Zhu. A fast pruned-extreme learning machine for classification problem. *Neurocomputing*, 72(1-3):359–366, 2008.

[76] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[77] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, illustrated edition edition, 2004.

[78] James Mercer. Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209(441-458):415–446, 1909.

[79] Richard Courant and David Hilbert. *Methods of Mathematical Physics*, volume 1. Wiley, New York, 1989.

[80] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Adaptive Computation and Machine Learning series, 2018.

[81] Léon Bottou and Chih-Jen Lin. Support vector machine solvers. *Large scale kernel machines*, 3(1):301–320, 2007.

[82] Jigang Wang, Predrag Neskovic, and Leon N Cooper. Training data selection for support vector machines. In *International Conference on Natural Computation*, pages 554–564. Springer, 2005.

[83] Jian-xiong Dong, Adam Krzyzak, and Ching Y Suen. Fast svm training algorithm with decomposition on very large data sets. *IEEE transactions on pattern analysis and machine intelligence*, 27(4):603–618, 2005.

[84] Zhi-Qiang Zeng, Hua-Rong Xu, Yan-Qi Xie, and Ji Gao. A geometric approach to train svm on very large data sets. In *2008 3rd International Conference on Intelligent System and Knowledge Engineering*, volume 1, pages 991–996. IEEE, 2008.

[85] Hans Graf, Eric Cosatto, Leon Bottou, Igor Dourdanovic, and Vladimir Vapnik. Parallel support vector machines: The cascade svm. *Advances in neural information processing systems*, 17:521–528, 2004.

[86] Dennis DeCoste and Kiri Wagstaff. Alpha seeding for support vector machines. In *Proceedings of the sixth ACM SIGKDD international*

conference on Knowledge discovery and data mining, pages 345–349, 2000.

[87] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[88] Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *European conference on principles of data mining and knowledge discovery*, pages 107–119. Springer, 2003.

[89] Jason Weston and Chris Watkins. Multi-class support vector machines. Technical report, Citeseer, 1998.

[90] Yi Liu and Yuan F Zheng. One-against-all multi-class svm classification using reliability measures. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 849–854. IEEE, 2005.

[91] Jair Cervantes, Farid Garcia-Lamont, Lisbeth Rodríguez-Mazahua, and Asdrubal Lopez. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408:189–215, 2020.