UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Saul Campos Berardo

Aprendizado Ativo com Agrupamentos em Espaço de Características

Belém
2015

Saul Campos Berardo

Aprendizado Ativo com Agrupamentos em Espaço de Características

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação (área de concentração: Sistemas de Computação) como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação. Instituto de Ciências Exatas e Naturais. Universidade Federal do Pará.

Orientador: Prof. Dr. Eloi Luiz Favero (PPGCC/UFPA)

Co-orientador: Prof. Dr. Nelson Cruz Sampaio Neto (PPGCC/UFPA)

Belém
2015

Dados Internacionais de Catalogação-na-Publicação (CIP)
Sistema de Bibliotecas da UFPA

Berardo, Saul Campos, 1986-
Aprendizado ativo com agrupamentos em espaço de características / Saul Campos Berardo. - 2015.

Orientador: Eloi Luiz Favero;
Coorientador: Nelson Cruz Sampaio.
Dissertação (Mestrado) - Universidade Federal do Pará, Instituto de Ciências Exatas e Naturais, Programa de Pós-Graduação em Ciência da Computação, Belém, 2015.

1. Aprendizado de computador. 2. Algoritmos de computador. I. Título.

CDD 23. ed. 006.31

Saul Campos Berardo

Aprendizado Ativo com Agrupamentos em Espaço de Características

Dissertação de Mestrado apresentada para obtenção do grau de Mestre em Ciência da Computação. Programa de Pós-Graduação em Ciência da Computação (área de concentração: Sistemas de Computação). Instituto de Ciências Exatas e Naturais. Universidade Federal do Pará.

Data da Aprovação: Belém-PA, __/__/____

Banca Examinadora:

Prof. Dr. Eloi Luiz Favero
Programa de Pós-Graduação em Ciência da Computação - UFPA - Orientador

Prof. Dr. Nelson Cruz Sampaio Neto
Programa de Pós-Graduação em Ciência da Computação - UFPA - Co-orientador

Prof. Dr. Roberto Célio Limão de Oliveira
Programa de Pós-Graduação em Engenharia Elétrica - UFPA - Membro Externo

Prof. Dr. Jefferson Magalhães de Morais
Programa de Pós-Graduação em Ciência da Computação - UFPA - Membro Interno

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS - ICEN
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Aprendizado Ativo com Agrupamentos em Espaço de Características

Autor: Saul Campos Berardo

DISSERTAÇÃO SUBMETIDA À AVALIAÇÃO DA BANCA EXAMINADORA APROVADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DA UNIVERSIDADE FEDERAL DO PARÁ E JULGADA ADEQUADA PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO COM ÊNFASE EM SISTEMAS INTELIGENTES.

APROVADA EM: ____/____/_____

BANCA EXAMINADORA:

Prof. Dr. Eloi Luiz Favero
(ORIENTADOR - UFPA)

Prof. Dr. Nelson Cruz Sampaio Neto
(CO-ORIENTADOR - UFPA)

Prof. Dr. Roberto Célio Limão de Oliveira
(MEMBRO EXTERNO - UFPA)

Prof. Dr. Jefferson Magalhães de Morais
(MEMBRO INTERNO - UFPA)

Visto

Prof. Dr. Jefferson Magalhães de Morais
(COORDENADOR DO PPGCC/ICEN - UFPA)

UFPA-ICEN-PPGCC
CAMPOS UNIVERSITÁRIO DO GUAMÁ
BELÉM-PARÁ-BRASIL
2015

“Nihil est in intellectu quod non
prius fuerit in sensu.”
(Nada está no intelecto que não
tenha estado antes nos sentidos)

Tomás de Aquino, De Veritate, q. 2

Agradecimentos

Agradeço ao meu orientador Eloi Luiz Favero pela orientação e por ter me apresentado aos primeiros algoritmos de Aprendizado de Máquina. Essas primeiras lições tiveram um papel muito importante no direcionamento de meus interesses de pesquisa. Agradeço ao meu co-orientador Nelson Cruz Sampaio Neto pela co-orientação, sem a qual este trabalho não teria alcançado o mesmo sucesso. Agradeço ao meu amigo de longa data Rafael Martins Feitosa por todas as conversas intermináveis, sem as quais este trabalho provavelmente não teria sido concebido. Agradeço ao CNPQ pelo suporte financeiro. Por fim, agradeço à minha família, cujo apoio foi essencial para a conclusão de mais esta etapa em minha vida.

Resumo

Há fortes evidências de que o córtex cerebral nos mamíferos é organizado de forma hierárquica e de que a percepção de conceitos complexos ocorre através da identificação e composição de conceitos mais simples. Esta organização cortical em camadas inspirou recentemente a criação de novos algoritmos de Aprendizado de Máquina, batizados coletivamente sob a alcunha de *Deep Learning*. Esses algoritmos foram responsáveis por grandes avanços na área nos últimos anos, permitindo a criação de sistemas de reconhecimento de fala e imagens, dentre outros, capazes de atingir em alguns *benchmarks* desempenho até mesmo superior a de seres humanos. Um problema grave das técnicas atuais é ainda a dependência de grandes quantidades de dados rotulados, que em muitos casos é de difícil obtenção. Além disso, deve-se considerar que seres humanos parecem ter pouca necessidade de um sinal supervisionado durante o processo de aprendizado. Portanto, é comum a crença de que para se desenvolver sistemas capazes de verdadeira inteligência artificial é preciso que sejam criadas novas técnicas de aprendizado não supervisionado, ou semi-supervisionado, minimizando a necessidade de dados rotulados. Neste trabalho, demonstra-se experimentalmente que algoritmos de aprendizado não supervisionado de características podem ser utilizados de forma hierárquica para converter dados de seus espaços originais para espaços de características, nos quais a comparação entre objetos é mais significativa e, portanto, algoritmos de agrupamentos, como o k-médias, são capazes de achar agrupamentos mais homogêneos. A partir dessa constatação, é proposta uma abordagem de aprendizado ativo (tipo de aprendizado semi-supervisionado na qual os rótulos são obtidos somente durante a execução do algoritmo através da interação com um agente externo), que obtém resultados no estado da arte no *benchmark* de reconhecimento de imagens MNIST.

Palavras-chave: Aprendizado de Máquina, Aprendizado Não Supervisionado de Características, Aprendizado Ativo, *Deep Learning*.

Abstract

There is strong evidence that the cerebral cortex in mammals is organized hierarchically and that perception of complex concepts occurs through the identification and composition of simpler ones. This layered cortical organization inspired recently the creation of new Machine Learning algorithms, collectively designated as Deep Learning. These new algorithms were responsible for big breakthroughs in the field in the last years, enabling the development of speech and image recognition systems, among others, that were able to achieve in some benchmarks even superhuman performances. A problem of these techniques is their dependence in large amounts of labeled data, which in many cases are hard to obtain. Besides that, we should consider that human beings do not seem to need so much supervised signals during the learning process. Therefore, it is a common believe that in order to develop systems with real artificial intelligence, we need to build new unsupervised or semi-supervised learning techniques, minimizing the need for labeled data. In this work, we experimentally demonstrate that unsupervised learning algorithms can be used hierarchically to convert the data from their original space to a feature space, in which the comparison between objects is more meaningful and therefore clustering algorithms, such as k-means, are able to find more homogenous clusters. From this finding, we propose an active learning approach (kind of semi-supervised learning in which labels are obtained just during algorithm execution through the interaction with an external agent), which achieves state-of-the-art results in the MNIST image recognition benchmark.

Keywords: Machine Learning, Unsupervised Feature Learning, Active Learning, Deep Learning.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Publicações

No decorrer do curso de mestrado foi elaborado o artigo *Active Learning with Clustering and Unsupervised Feature Learning*, publicado pela Springer International Publishing como capítulo do livro *Advances in Artificial Intelligence: 28th Canadian Conference on Artificial Intelligence, Canadian AI 2015, Halifax, Nova Scotia, Canada, June 2-5, 2015, Proceedings* na série *Lecture Notes in Artificial Intelligence*.

Lista de Símbolos e Abreviaturas

AM:	Aprendizado de Máquina
ConvNet:	*Convolutional Neural Network*
DAE:	*Denoising Auto-Eencoder*
DNN:	*Deep Neural Network*
GPG:	*Graphics Processor Unit*
GRF:	*Gaussian Random Field*
IA:	Inteligência Artificial
KNN:	*K-Nearest Neighbors*
MLE:	*Maximum Likelihood Estimation*
MLP:	*Multi Layer Perceptron*
PCA:	*Principal Component Analysis*
RBM:	*Restricted Boltzmann Machine*
SDAE:	*Stacked Denoising Auto-Encoder*
SVM:	*Support Vector Machine*

Capítulo 1

Introdução

A importância da experiência sensorial na construção de conhecimento foi reconhecida já por Aristóteles e posteriormente resumida por Tomás de Aquino através da citação latina em epígrafe (*nada está no intelecto que não tenha estado antes nos sentidos*). Embora esta ideia tenha permeado a tradição filosófica ocidental e tenha sido promovida a uns dos pilares do método científico, as primeiras explicações científicas dos processos biológicos e computacionais responsáveis pela percepção surgiram somente ao longo do século XX.

A descoberta do neurônio em 1870 por Golgi e o reconhecimento deste como principal unidade funcional do sistema nervoso em 1887 por Ramón y Cajal inspiraram posteriormente a criação por McCulloch & Pitts [1943] do primeiro modelo de neurônio artificial, que embora simples, era capaz de computar funções booleanas arbitrárias quando estruturado em redes. A descoberta do mecanismo de plasticidade sináptica hebbiano [Hebb 1949] influenciou a criação do Perceptron, o primeiro algoritmo de aprendizado de redes neurais artificiais [Rosenblatt 1962]. Alguns anos antes, ainda na década de 50, Selfridge [1959] propora a criação de um modelo de redes de neurônios com hierarquia de múltiplas camadas, na qual cada camada seria responsável pela identificação de padrões de complexidade crescente nas camadas inferiores, permitindo o reconhecimento de conceitos de alto nível, como imagens, por exemplo. Nos anos seguintes, o algoritmo backpropagation foi desenvolvido simultaneamente por vários pesquisadores [Werbos 1974, LeCun 1985, Rumelhart et al. 1986], possibilitando o treino de redes neurais artificiais com múltiplas camadas, como as propostas por Selfridge. O sucesso deste algoritmo foi, no entanto, limitado. Embora arquiteturas "profundas" possuíssem vantagens teóricas, na prática o algoritmo backpropagation não era capaz de treiná-las. Esta dificuldade decorre do fenômeno apelidado de *vanishing gradient*[1], que foi identificado

inicialmente por Hochreiter [1991]. Até recentemente, salvo poucas exceções — e.g. *Convolutional Neural Network* (ConvNet), que foram utilizadas com sucesso no reconhecimento de dígitos em cheques na década de 90 [LeCun et al. 1998] —, aplicações baseadas em redes neurais com mais de uma camada intermediária foram pouco utilizadas. Este cenário mudou radicalmente na última década com o advento de novas técnicas para treinamento de redes profundas [Bengio et al. 2007, Hinton et al. 2006, Ranzato et al. 2007]. Estas técnicas foram batizadas coletivamente sob a denominação de *deep learning*.

As primeiras abordagens de *deep learning* possuíam sobretudo dois elementos em comum: (i) inicialização dos parâmetros do modelo através do treinamento camada por camada (chamado de "pré-teino"); (ii) minimização de uma função de reconstrução dos dados de treino de forma não supervisionada (isto é, utilizando unicamente dados não rotulados). O procedimento de pré-treino funciona, pois a os parâmetros da rede são inicializados na vizinhança de um bom mínimo local, melhor do que se obtém com a inicialização aleatória dos parâmetros. Embora estes métodos tenham conseguido um grande sucesso, eles caíram rapidamente em desuso a partir da criação de novos métodos de treinamento, como a *otimização livre de hessiana* [Martens 2010], que possibilitaram a obtenção de resultados ainda melhores, sem a utilização de uma etapa adicional de treinamento não supervisionado.

O enorme sucesso dos métodos atuais de *deep learning*, que permitiram até mesmo a construção de sistemas com performances super-humanas em algumas tarefas [He et al. 2015, Ciresan et al. 2012, Mnih et al. 2013], foram possíveis graças à disponibilidade atual de quantidades moderadas de dados rotulados e à criação de arquiteturas de processamento em paralelo de alto desempenho baseadas em GPUs e *clusters*. A utilização de dados não rotulados (existentes em abundância) tem desempenhado um papel secundário no treinamento de redes neurais profundas em aplicações comerciais. No entanto, ainda é forte a crença de que o futuro da IA (Inteligência Artificial) depende da utilização de dados não rotulados [LeCun 2015]. Esta premissa é influenciada sobretudo pela constatação evidente do processo pelo qual o ser humano identifica novos conceitos no mundo. Para um ser humano, comumente basta a identificação de um único exemplar de uma categoria para que se aprenda a identificá-la, enquanto que os algoritmos atuais de AM (Aprendizado de Máquina) ainda necessitam de grandes quantidades de dados rotulados para cada novo conceito. Baseando-se neste pressuposto, este trabalho descreve uma abordagem desenvolvida para utilização de dados não rotulados para o treinamento

[1] A magnitude do gradiente decresce exponencialmente à medida que os erros são propagados para as camadas inferiores, inviabilizando a alteração dos pesos nas camadas mais baixas.

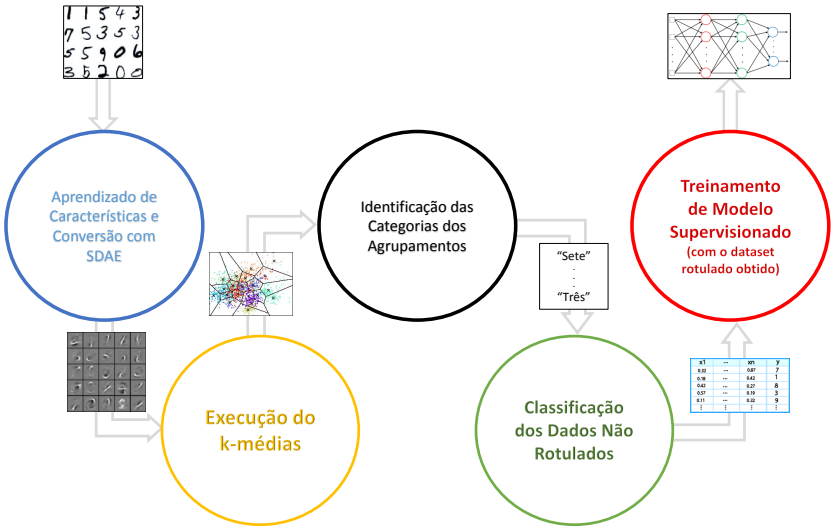

Figura 1.1: Etapas executadas para treinamento de classificador utilizando a abordagem proposta.

de modelos preditivos, cujas etapas estão ilustradas na Figura 1.1, descrita em maiores detalhes no Capítulo 4.

1.1 Descrição do Problema

Há fortes indícios de que o processamento da informação visual no córtex dos mamíferos ocorre de forma hierárquica. Wiesel & Hubel [1959] demonstraram que os neurônios da região V1 (a parte do córtex onde ocorre o primeiro estágio do processamento visual) são especializados no reconhecimento de padrões oriundos da retina correspondentes a segmentos de retas orientados e localizados. Olshausen [1996] propôs um algoritmo não supervisionado (Codificação Esparsa) capaz de extrair de imagens naturais (isto é, imagens "comuns") padrões similares aos detectados pela região V1. Há evidências fortes de que os neurônios da região V2 (a parte do córtex onde ocorre o segundo estágio do processamento visual), por sua vez, são especializados na detecção de combinações dos padrões identificados previamente na região V1, possibilitando a identificação de formas de maior complexidade [Ito & Komatsu 2004]. Padrões de ativação similares aos da região V2 foram obtidos pela aplicação do algoritmo de Codificação Esparsa em duas camadas [Lee et al. 2008]. A criação de um modelo de grande porte com nove camadas, utilizando uma enorme quantidade de imagens baixadas aleatoriamente da internet,

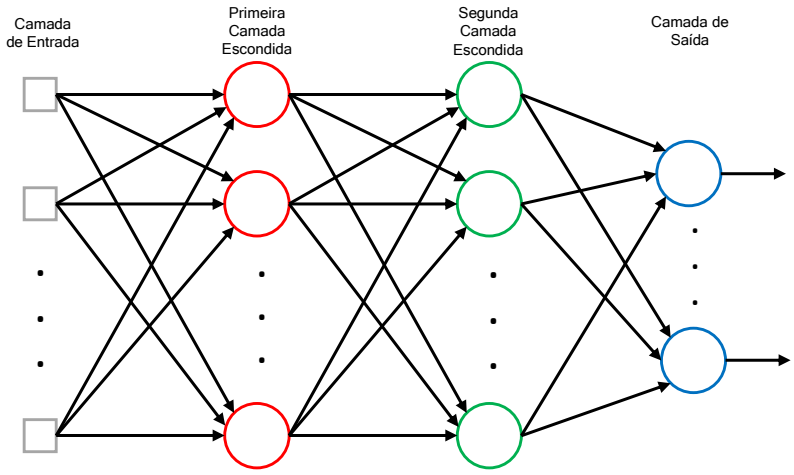

Figura 1.2: Exemplo de Rede Neural com múltiplas camadas. A imagem da camada de entrada é um ponto em um espaço vetorial, no qual cada coordenada representa a intensidade de um pixel. A primeira camada escondida realiza uma transformação não linear, projetando o ponto do espaço original para um novo espaço no qual cada coordenada do vetor indica a probabilidade da presença (ou ausência) de uma característica específica. Cada camada posterior repete este processo, projetando os dados de entrada em um novo espaço de características, no qual as coordenadas indicam a existência de características de mais alto nível.

permitiu a identificação de padrões de alto nível, tais quais rostos de gatos e humanos [Le 2013], sem a necessidade de utilização de qualquer rótulo. Embora o funcionamento do cérebro ainda seja um mistério, estes resultados servem como evidências de que os processos biológicos dos cérebros dos mamíferos executam um tipo de algoritmo não supervisionado especializado em extrair dos dados sensoriais características organizadas hierarquicamente, capazes de representar conceitos de complexidade crescente baseadas em conceitos mais simples.

Matematicamente, o processo de extração de características por uma rede neural também pode ser entendido como uma transformação não linear entre espaços vetoriais. A Figura 1.2 mostra esquematicamente as transformações sucessivas aplicadas a uma imagem por cada camada. Cada camada projeta os vetores da camada inferior em um novo espaço, no qual as coordenadas passam a ter um significado diferente: pixels, na camada de entrada; características de nível crescente, nas camadas superiores.

A representação de alto nível de um vetor de entrada favorece a comparação entre dois objetos utilizando um métrica de dissimilaridade. É fácil observar que em uma imagem o mero deslocamento da posição de um objeto faz com que a imagem resultante deixe de compartilhar grande parte dos pixels com a imagem original, contudo as caracterís-

ticas de alto nível que compoem o objeto continuam. Mecanismos para comparação da similaridade entre dois objetos são essenciais em quase todas as técnicas de aprendizado de máquina. Algoritmos de agrupamento, por exemplo, utilizam métricas de distância que medem o quão diferentes dois objetos são. Diante destas observações, os seguintes questionamentos são feitos:

- Um algoritmo não supervisionado de aprendizado de características poderia ser utilizado para se converter dados brutos de entrada (e.g. pixels em imagens) em representações de mais alto nível (e.g. partes de objetos), nas quais as métricas de distâncias utilizadas por algoritmos de agrupamento sejam mais significativas e consequentemente a qualidade dos agrupamentos encontrados seja maior?
- Caso sejamos capazes de encontrar agrupamentos relativamente homogêneos, seria possível determinar a categoria predominante em cada agrupamento utilizando os rótulos de apenas algumas amostras (e.g. pedindo a um anotador humano para classificar o objeto) e assim obter uma grande quantidade de dados rotulados (considerando que todos os exemplares pertencentes a um mesmo agrupamento pertencem à mesma categoria)?
- A grande quantidade de dados rotulados obtidos desta maneira (utilizando-se uma quantidade pequena de rótulos) poderia ser utilizada como dados de treino de uma rede neural, e assim se obter um classificador com acurácia superior à que se poderia obter com o treinamento de um modelo supervisionado utilizando a mesma quantidade de exemplares rotulados?

1.2 Objetivos

O objetivo geral deste trabalho é determinar experimentalmente a viabilidade da estratégia aprendizado ativo proposta, baseada no uso de técnicas não supervisionadas de extração de características e no uso de algoritmos de agrupamentos. Os objetivos específicos consistem em determinar experimentalmente se:

- a utilização de técnicas não supervisionadas de extração de características é capaz de melhorar a qualidade dos agrupamentos encontrados por algoritmos de agrupamento;
- as categorias mais prevalentes de agrupamentos podem ser determinadas a partir do rótulo de algumas poucas amostras;
- os dados rotulados com as classes das categorias mais prevalentes de cada agrupamento podem ser utilizados como dados de treino de um modelo preditivo, que

obtenha desempenho ao menos similar ao de outras técnicas de aprendizado semi-supervisionado no estado da arte, ao se utilizar as mesmas quantidades de exemplares rotulados.

As ações executadas para atingir os objetivos foram as seguintes:

- Comparar a relação existente entre a qualidade dos agrupamentos encontrados por algoritmos de agrupamento quando se utiliza os dados brutos com a qualidade ao se utilizar técnicas de extração de características não supervisionadas e PCA (*Principal Component Analysis*).
- Determinar o tamanho do erro cometido ao se classificar todos os exemplares de um agrupamento com a classe de uma pequena amostra representativa do agrupamento ao invés de se classificar os pontos com a classe mais frequente do agrupamento.
- Propor e avaliar experimentalmente um modelo probabilístico para determinação da probabilidade da classificação em modelos baseados em distância (e.g KNN - *K-Nearest Neighbours*).
- Determinar a influência do método de inicialização do k-médias e da sua função de distorção na pureza dos agrupamentos encontrados.
- Determinar se os dados rotulados obtidos pela classificação com os rótulos das amostras representativas podem ser utilizados como dados de treinamento de uma rede neural e assim se obter um classificador com acurácia superior ao que seria obtido ao se utilizar somente os dados rotulados como dados de treino.
- Comparar a acurácia da abordagem proposta com a de outras abordagens de aprendizado semi-supervisionado ao se utilizar as mesmas quantidades de dados rotulados.

1.3 Contribuições

A principal contribuição deste trabalho foi a criação de uma abordagem de aprendizado semi-supervisionado (mais especificamente, aprendizado ativo), cujos resultados no *benchmark* MNIST [LeCun et al. 1998] são comparáveis aos obtidos por outras técnicas no estado da arte ao se utilizar as mesmas quantidades de exemplares rotulados. Outras contribuições secundárias foram:

- Demonstração experimental da melhora da qualidade de agrupamentos encontrados pelo algoritmo k-médias através da utilização de aprendizado não supervisionado de características.

- Demonstração experimental de que o erro obtido ao se classificar os exemplares dos agrupamentos encontrados pelo k-médias com a classe mais frequente do agrupamento é similar ao erro obtido ao se classificar os exemplares com a classe do ponto mais próximo de seu centroide.
- Criação dos modelos probabilísticos para determinação da probabilidade de classificação em algoritmos baseados em distância descritos na seção 4.3.

1.4 Estrutura da Dissertação

Esta dissertação é dividida em seis capítulos. Após esta introdução, no Capítulo 2 são explicados os elementos teóricos utilizados neste trabalho. No Capítulo 3 são descritos os principais trabalhos de aprendizado semi-supervisionado desenvolvidos em pesquisas anteriores (que de alguma forma "concorrem" com a abordagem proposta). No Capítulo 4 a abordagem proposta de aprendizado ativo é descrita. No Capítulo 5 os resultados obtidos são apresentados em comparação aos resultados de outras abordagens de aprendizado semi-supervisionado no estado da arte. No Capítulo 6 são apresentadas as conclusões e sugestões de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Distingue-se tradicionalmente na filosofia duas formas de raciocínio: dedutivo e indutivo. Enquanto o raciocínio dedutivo é tradicionalmente associado às abordagens simbólicas, como o *Logic Theorist* de Newell & Simon [1956], que predominaram nos primórdios da IA até meados da década de 80; o raciocínio indutivo é associado às técnicas de IA atualmente consideradas como Aprendizado de Máquina (AM), que vem recebendo crescente atenção na última década [Russell & Norvig 2009].

Embora a IA simbólica tenham sido bem-sucedida em algumas áreas (como na criação de sistemas especialistas), tarefas consideradas "triviais" para seres humanos se mostraram de muito difícil execução. Enquanto, no final da década de 90, um sistema de IA simbólica era capaz de derrotar o campeão mundial xadrez [Newborn 1996], a "simples" tarefa de reconhecer imagens de rostos humanos ainda parecia longe de uma solução.

O termo "aprendizado" em AM é definido de forma abrangente. Ele inclui "qualquer programa capaz de melhorar o desempenho na realização de alguma tarefa por meio da experiência" [Tom Mitchel 1997]. A depender do objetivo da tarefa e do tipo de dados de treino disponíveis, a tarefa de aprendizado pode ser classificada como:

- **Aprendizado Supervisionado**: o objetivo é induzir a partir de um conjunto de dados de treino, formado por pares de exemplos de treino e respectivo valores associados, uma função que mapeia um exemplar arbitrário ao valor associado. Os valores de saída associados podem ser categorias, nas tarefas de classificação, ou valores numéricos, nas tarefas de regressão. Ex.: a partir de um conjunto de imagens de animais rotuladas com os nomes dos respectivos animais, infere-se uma função (classificador) que identifica o animal presente em uma imagem nova.
- **Aprendizado Não Supervisionado**: o objetivo é descobrir alguma estrutura presente nos dados. Esta estrutura pode consistir em sub-regiões de alta densidade

dentro do espaço onde os dados residem (em tarefas de agrupamento e redução de dimensionalidade) ou padrões frequentes de associação (em tarefas de associação). O conjunto de dados de treino é composto por exemplos sem nenhum valor de saída associado. Ex.: a partir de um conjunto de imagens de animais sem nenhuma classificação prévia, deseje-se agrupar as imagens de animais da mesma espécie.

- **Aprendizado Semi-Supervisionado**: o objetivo é igual ao do aprendizado supervisionado, no entanto, além de se utilizar o conjunto de dados rotulados (composto pelos pares de exemplos e valores de saída), também se utiliza um conjunto de dados não rotulados (sem os valores de saída). Espera-se, assim, que a estrutura presente nos dados não rotulados contribua para a indução de uma função. Ex.: a partir de uma pequena quantidade de imagens de animais rotuladas e um conjunto grande de imagens aleatórias baixadas na internet, infere-se uma função (classificador) que identifica o animal presente em uma imagem nova.
- **Aprendizado por Reforço**: o objetivo é escolher a sequência de decisões que maximiza uma função de recompensa cumulativa. Neste tipo de tarefa, o programa interage com o ambiente e decide que ações tomar em cada instante do tempo. A depender da sequência de ações escolhidas, a recompensa obtida pode ser maior ou menor. Não se utiliza um conjunto de dados previamente coletado, como nos tipos de tarefas anteriores. A experiência é adquirida através da iteração com o ambiente e através da função de recompensa, que informa ao algoritmo de aprendizado o resultado de suas tomadas de decisões. Ex.: deseja-se criar um programa para jogar *Space Invaders*. O programa recebe como entrada a imagem da tela do jogo, atualizada periodicamente, e a pontuação obtida após cada ação (função de recompensa). O objetivo de aprendizado é escolher a sequência de comandos no *joystick* para maximizar a pontuação obtida após várias partidas.

A abordagem proposta neste trabalho, embora seja uma abordagem de aprendizado semi-supervisionado, é composta também de etapas de aprendizado não supervisionado e de aprendizado supervisionado. Neste capítulo são descritos os conceitos fundamentais destas três áreas necessários para a compressão da abordagem proposta. Conceitos elementares de AM constantes da literatura tradicional da área podem ser consultados em [Bishop 2007, Hastie et al. 2009].

2.1 Aprendizado Supervisionado

Algoritmos de aprendizado supervisionado têm o objetivo de, a partir de um conjunto de dados de treino, inferir uma função que associa a cada exemplo o respectivo valor de saída. Mais formalmente, o algoritmo recebe como entrada um conjunto de dados de treino $D = \left\{ (\mathbf{x^{(1)}}, y^{(1)}),\ \ldots\ , (\mathbf{x^{(n)}}, y^{(n)}) \right\}$, formado por pares de *exemplos de treino* $\mathbf{x}$ e respectivos *valores associados* y, e gera como saída uma função $h(\mathbf{x})$ (*hipótese*), tal que o erro cometido ao se aproximar y por $h(\mathbf{x})$ é minimizado por algum critério, a depender do modelo específico. Embora $\mathbf{x}$ possa ser um valor escalar, em geral $\mathbf{x}$ é um vetor de valores reais $\mathbf{x} = (x_1,\ \ldots\ x_d)$, onde d é o número de dimensões e cada elemento x de $\mathbf{x}$ é uma *característica* extraída dos dados. O valor associado y pode ser do tipo escalar ou vetorial (neste trabalho nos limitaremos ao caso no qual y é escalar). Adotar-se-á a convenção de representar os valores vetoriais por letras em negrito, enquanto os valores escalares serão representados por letras em fonte normal.

Caso a função $h(\mathbf{x})$ assuma valores numéricos, a tarefa é um problema de regressão. Caso assuma valores dentro um conjunto discreto e não ordenado, a tarefa é um problema de classificação. Exemplos de regressão e classificação são: regressão linear e regressão logística, repectivamente; que serão descritos brevemente antes do tratamento de redes neurais.

2.1.1 Regressão Linear

Regressão linear é um modelo clássico da estatística, que também pode ser compreendido como um modelo de AM. No exemplo mais simples de regressão linear, o conjunto de dados de treino é formado por pontos no $\mathbb{R}^2$. O objetivo do algoritmo é determinar a linha que melhor se ajusta aos pontos, como ilustrado na Figura 2.1. Considerando que a equação de uma linha é dada por

$$h(x) = w_0 + w_1 x \tag{2.1}$$

onde os parâmetros w_0 e w_1 são respectivamente a interceptação com o eixo y e o coeficiente angular, cada escolha possível destes parâmetros determina uma reta diferente. Para se achar a reta que melhor se adequa aos dados de treino, pode-se associar a cada escolha dos parâmetros uma *função custo* $J(w_0, w_1)$, que mensura o desvio quadrático médio do valor real de cada y ao valor $f(x)$ calculado pela reta escolhida. Esta função é dada por

$$J(w_0, w_1) = \frac{1}{2n} \sum_{i=0}^{n} (y^{(i)} - h(x^{(i)}))^2 \tag{2.2}$$

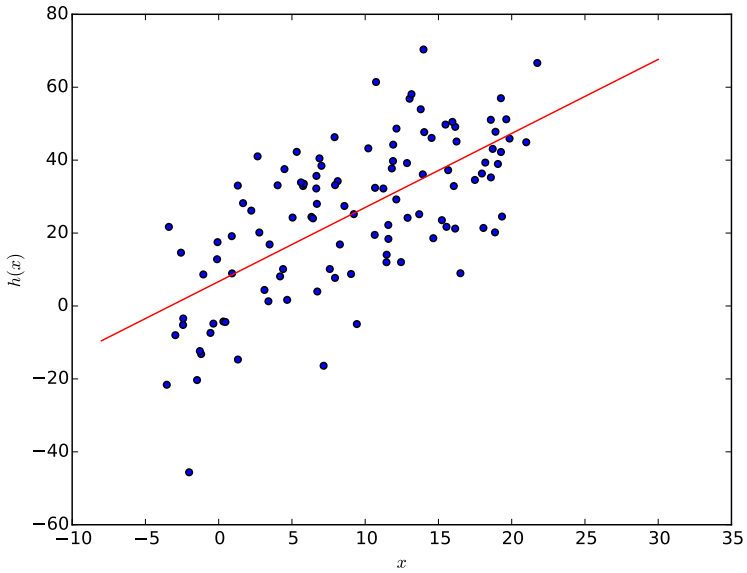

Figura 2.1: Exemplo de problema de regressão linear. A cada escolha possível dos parâmetros w_0 e w_1 está associada uma reta. O problema consiste em achar a reta que melhor se ajusta aos pontos do conjunto de treino.

onde n é o número de pontos no conjunto de dados de treino e o fator 2 no denominador é adicionado por convenção, para simplificar as contas em uma etapa posterior, já que sua inserção não influencia na determinação dos parâmetros que minimizam a função. A função custo é ilustrada na Figura 2.2. Embora existam fórmulas fechadas para se achar os valores dos parâmetros que a minimizam, o algoritmo de descida de gradiente será descrito, pois este algoritmo desempenha um papel importante no treinamento de redes neurais.

Na maior parte dos problemas de interesse prático a variável dependente y é dada em função de um vetor de características $\mathbf{x} = (x_1, \ \ldots \ x_d)$, ao invés de um valor escalar, e consequentemente não se deseja encontrar uma reta, mas sim um hiper-plano, dado por

$$h(\mathbf{x}) = w_0 + w_1 x_1 + \ \ldots \ + w_d x_d \tag{2.3}$$

Adotando-se a convenção de que $b = w_0$ e $\mathbf{w} = (w_1, \ \ldots \ , w_d)$, a Equação 2.3 pode ser escrita em forma vetorial como

$$h(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i = \mathbf{w}^T \mathbf{x} + b \tag{2.4}$$

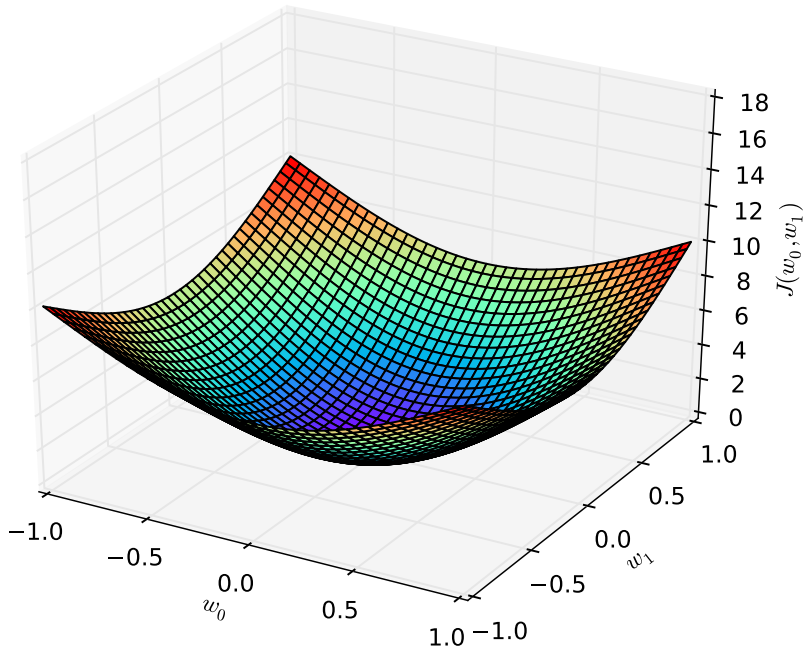

Figura 2.2: Exemplo de função custo em um problema de regressão linear. O ponto que minimiza a função (parâmetros w_0 e w_1) pode ser determinado com o algoritmo de descida de gradiente.

2.1.2 Descida de Gradiente

Imagine um astronauta perdido em algum ponto aleatório dentro de uma cratéra completamente escura na lua. Ele precisa achar o fundo da cratéra, onde se encontra sua espaçonave, para poder retornar à terra. Em cada instante o astronauta explora com o pé o terreno ao seu redor e determina a direção de maior declive e, em seguida, salta nesta direção. Caso a crátera seja concava (o que é o caso em uma função quadrática), após cada salto, o astronauta chegaria mais próximo de sua espaçonave. Este procedimento de busca é análogo ao executado pelo algoritmo de descida de gradiente. Para se achar o ponto de mínimo de uma função, a partir de um ponto arbitrário, vários passos sucessivos são tomados nas direções de maior declive, que são determinados pelo grandiente da função em cada ponto.

No problema de regressão linear descrito, deseja-se obter os parâmetros w_0 e w_1 que minimizam a função custo $J(w_0, w_1)$. Escolhe-se inicialmente valores arbitrários para os

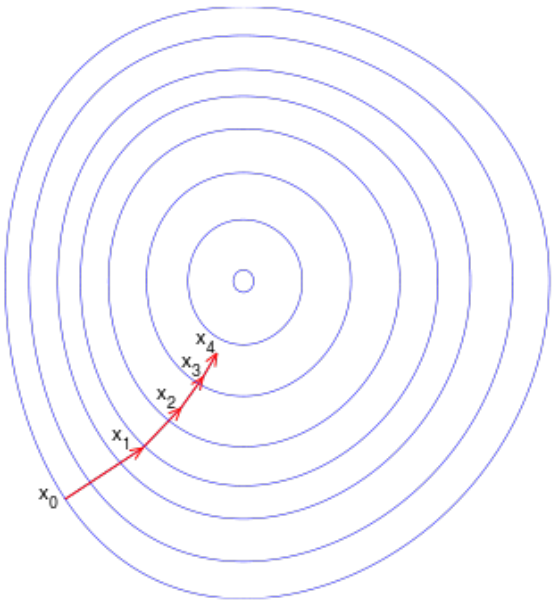

Figura 2.3: Exemplo do algoritmo de descida de gradiente. Em cada iteração o algoritmo determina a direção de maior declive (indicada pelas curvas de nível) e dá um "salto"em direção ao ponto de mínimo.

parâmetros, e, em cada passo, novos parâmetros w_j são determinados por

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1) \tag{2.5}$$

onde α é a taxa de aprendizado, que no exemplo do astronauta poderia ser compreendida como a “força” com que cada salto é realizado. O sinal negativo de α indica que o salto é direcionado para “baixo” (já que o objetivo é encontrar o ponto mínimo, diferentemente do que ocorreria no algoritmo de subida de gradiente, onde se deseja encontrar o ponto de máximo). A derivada parcial da função custo em relação a cada parâmetro é obtida facilmente através da aplicação da regra da cadeia, que nos dá

$$\frac{\partial}{\partial w_j} J(w_0, w_1) = \sum_{i=0}^{n} (h(x^{(i)}) - y^{(i)}) x_j^{(i)} \tag{2.6}$$

O algoritmo de descida de gradiente é ilustrado na Figura 2.3. Em funções convexas, que possuem um único mínimo global, o algoritmo é sempre capaz de achar este mínimo. Em funções não convexas, que podem possuir vários mínimos locais, o algoritmo não garante que o mínimo global seja atingido. Nos problemas de regressão linear, a função custo é sempre convexa (já que é quadrática), enquanto em redes neurais, por outro lado,

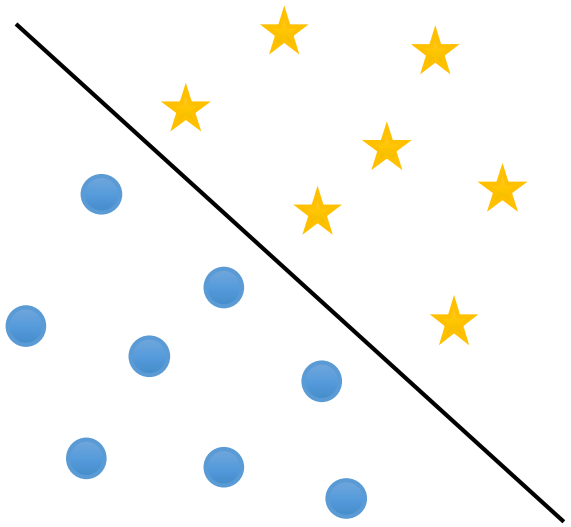

Figura 2.4: Exemplo de regressão logística. Assim como em problemas de regressão linear, deseja-se determinar os parâmetros de um hiperplano, no entanto neste caso a reta divide o espaço em duas regiões, isolando pontos de diferentes classes.

a função custo quase nunca é, e consequentemente possui muitos mínimos locais.

Para impedir que o algoritmo fique preso muito cedo em um mínimo local "ruim", algumas técnicas são comumente utilizadas. Uma delas é o SGD (*stochastic gradient descent*), na qual ao invés de se utilizar todos os pontos do cojunto de treino em cada passo do algoritmo, utiliza-se apenas um único exemplar no cálculo do gradiente. Desta forma, a probabilidade de o gradiente não apontar exatamente para a região de máximo declive é alta. Após vários passos, no entanto, o algoritmo tende a ir na direção das regiões onde a função é mínima. O comportamente aleatório introduzido desta maneira ajuda o algoritmo a escapar de regiões de mínimos locais ruins. É possível também se utilizar um conjunto de algumas dezenas (ou centenas) de exemplares do conjunto de treino, ao invés de apenas um único ponto. Neste caso o algortimo é conhecido como BGD (*batch gradient descent*) e resulta em um comportamente menos caótico (esta é modalidade mais utilizada na prática).

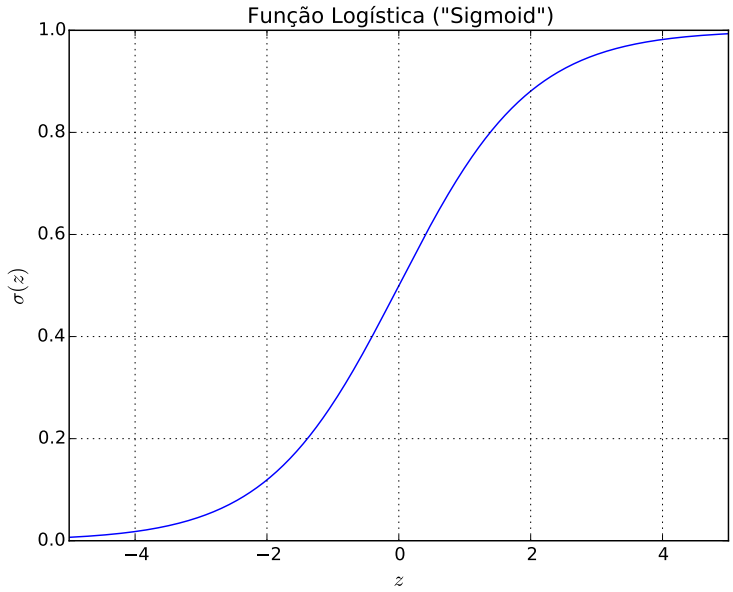

Figura 2.5: Variação da função logística no intervalo de -5 a 5. A função, que tem um formato "sigmoidal", se aproxima de zero para valores pequenos de z. A medida que z aumenta, a função tende a um.

2.1.3 Regressão Logística

No exemplo mais simples de regressão logística, assim como na regressão linear, deseja-se determinar os parâmetros w_0 e w_1 de uma reta, mas neste caso a reta desejada é aquela que divide o espaço em duas regiões, isolando pontos pertencentes a duas categorias diferentes, como ilustrado na Figura 2.4.

Como no exemplo de regressão linear, a cada escolha possível dos parâmetros da reta é associado o valor de uma função custo, que determina o quão bem a reta definida pelos parâmetros separa as duas categorias de pontos. Utilizando a técnica de estimação por máxima verosimilhança (do inglês, MLE — *Maximum Likelihood Estimation*), pode-se chegar a fução custo adequada, dado por

$$J(\mathbf{w}) = -\frac{1}{n}\left[\sum_{i=1}^{n} y^{(i)} \log f(\mathbf{x^{(i)}}) + (1+y^{(i)}) \log\left(1 - f(\mathbf{x^{(i)}})\right)\right] \tag{2.7}$$

Embora esta função tenha uma forma muito diversa da função custo da Equação 2.2, utilizada na regressão linear, o seu papel é análogo, de medir o quão bem os parâmetros definem a reta (ou o hiper-plano) desejado.

Na regressão logística, pode-se também utilizar o algoritmo de descida de gradiente para se minizar a função custo, atualizando os parâmetros em cada passo com a regra da

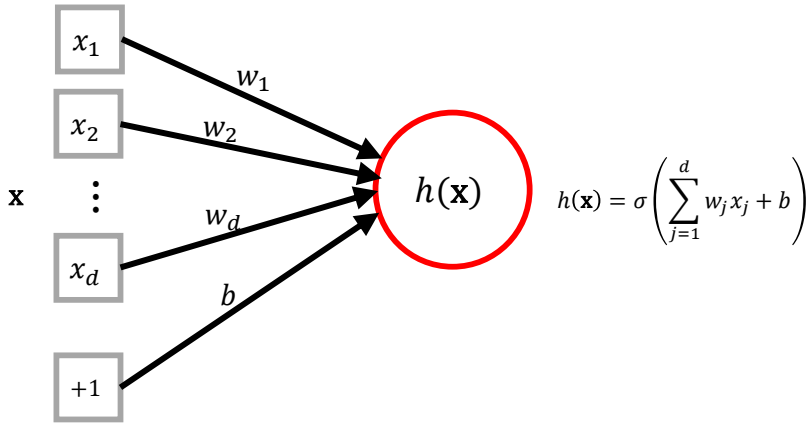

Figura 2.6: Modelo do funcionamento de um neurônio. Os valores de entrada x_i são multiplicados pelos parâmetros w_i e em sequida somados. Caso esta soma assuma um valor elevado, o neurônio é "ativado" (gera um valor próximo de um). Caso contrário, o neurônio permanece "inativo" (gera um valor próximo de zero).

Equação 2.5. A expressão da derivada parcial da função custo em relação aos parâmetros é dada por

$$\frac{\partial}{\partial w_j} J(\mathbf{w}) = \sum_{i=0}^{n} (h(\mathbf{x}^{(\mathbf{i})}) - y^{(i)}) \mathbf{x}_{\mathbf{j}}^{(\mathbf{i})} \tag{2.8}$$

Esta expressão é quase idêntica a derivada da função custo da função linear, no entanto, neste caso a hipótese $h(\mathbf{x})$ assume uma forma distinta da utilizada na Equação 2.6. Enquanto que na regressão linear a hipótese h consiste simplesmente em uma função linear, na regressão logística ela é dada por

$$h(\mathbf{x}) = \sigma(z) = \frac{1}{1 + \exp(-z)} \tag{2.9}$$

onde $z = \mathbf{w}^T \mathbf{x} + b$ é o valor de entrada da função logística, que possui um formato sigmoidal, ilustrado na figura 2.5. Para valores pequenos de z, $\sigma(z)$ tende a zero, enquanto para valores grandes, $\sigma(z)$ tende a um. Este valor pode ser interpretado como a $P(Y \mid x)$, isto é, a pobabilidade condicional do exemplar x pertencer a classe Y.

2.1.4 Redes Neurais

A hipótese da Equação 2.9, utilizada na regressão logística, é composta por duas partes: uma linear $z = \mathbf{w}^T \mathbf{x} + b$, a soma dos valores $\mathbf{x}$ ponderada pelos parâmetros $\mathbf{w}$; e

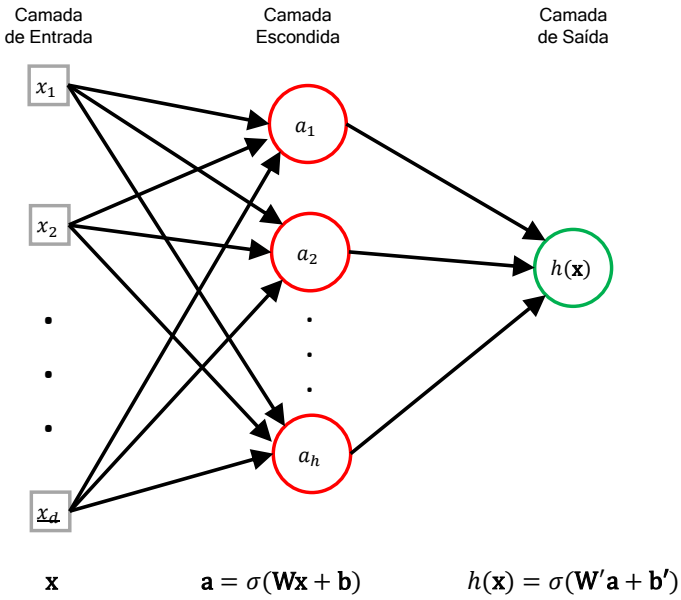

Figura 2.7: Exemplo de rede neural. A rede neural ilustrada calcula o valor da função $h(\mathbf{x})$ da Equação 2.13

.

uma não-linear, a função "sigmoidal" $\sigma(z)$. Esta equação é um modelo aproximado do funcionamento de um neurônio, ilustrado na Figura 2.6.

Identicamente à regressão logística, os parâmetros de um neurônio definem um hiperplano que divide o espaço de entrada em duas regiões. Intuitivamente, um neurônio pode ser entendido como um *detector de característica*. Um neurônio treinado para determinar se em alguma região de uma imagem há ou não uma figura de um nariz pode ser entendido como um detector da característica "nariz". Vários neurônios operando paralelamente em uma camada são capazes, portanto, de detectar diferentes características simultaneamente. Estas características podem ser utilizadas como valores de entrada de uma camada de neurônios superior, capaz de detectar a combinação de características extraídas da camada de entrada. Este processo pode ser repetido formando uma hierarquia capaz de detectar categorias complexas formadas por múltiplas partes.

Na Figura 2.7 é ilustrada uma rede neural composta por três camadas. A *camada de entrada* recebe os valores da variável $\mathbf{x} = (x_1, \ldots, x_d)$. A *camada escondida* é composta por h neurônios. Cada neurônio i da camada escondida está conectado à unidade j da camada de entrada através da conexão w_{ij}. Cada neurônio da camada escondida também está a conectada uma "unidade de viés" b_i, que desempenha o papel do parâmetro w_0 da Equação 2.3. O valor de saída de cada neurônio da camada escondida é dado por

$$a_i = \sigma\left(\sum_{j=1}^{d} w_{ij}x_j + b_i\right) \tag{2.10}$$

O único neurônio da camada de saída está conectado a cada neurônio da camada escondida através da conexão w'_k. A este neurônio também está associado uma unidade de viés b'. O valor total de saída calculado pela rede é dado por

$$h(\mathbf{x}) = \sigma\left(\sum_{k=1}^{h} w'_k a_k + b'\right) \tag{2.11}$$

que assume valores entre zero e um e pode ser interpretado como a probabilidade condicional de o vetor de entrada $\mathbf{x}$ pertencer à classe para a qual a rede foi treinada para reconhecer.

O valor de saída da rede e da camada escondida podem ser escritos simplificadamente com notação vetorial. Seja $\mathbf{W}$ a matriz dos parâmetros w_{ij}, com h linhas e d colunas, $\mathbf{b}$ um vetor coluna com os viéses das unidades da camada escondida, $\mathbf{w}'$ o vetor dos parâmetros w'_k, b' o viés da camada de saída, e considerando-se o vetor de entrada $\mathbf{x}$ como um vetor coluna, o valor de saída da rede é dado por

$$\mathbf{a} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2.12}$$

$$h(\mathbf{x}) = \sigma\left(\mathbf{w}'\mathbf{a} + b'\right) \tag{2.13}$$

onde $\mathbf{a} = (a_1, \ldots, a_h)$ é o vetor das ativações calculadas pela camada escondida. Em casos nos quais há mais de um neurônio na camada de saída, o vetor de parâmetros $\mathbf{w}'$ também pode ser representado através de uma matriz. Esta notação será utilizada na Seção 2.2.1, sobre *Denoising Autoencoders*.

O algoritmo mais utilizado no treinamento da redes neurais é o *backpropagation*, que não é mais que o algoritmo de descida de gradiente, explicado na seção 2.1.2, utilizando princípios de programação dinâmica para realizar o cálculo dos gradientes de forma eficiente, armazenando cálculos intermediários, que são utilizados em diferentes momentos durante a execução do algoritmo.

2.2 Aprendizado Não Supervisionado

Algoritmos de aprendizado não supervisionado têm como objetivo encontrar algum tipo de estrutura em um conjunto de dados não rotulados. O tipo de estrutura identificado

varia de acordo com o tipo de algoritmo. Nesta seção serão descritas duas tarefas de aprendizado não supervisionado utilizadas neste trabalho.

2.2.1 Aprendizado Não Supervisionado de Características

Até recentemente grande parte do esforço empregado na criação de um novo modelo de AM era gasto na engenharia de características. Os conjuntos de características criados manualmente por especialistas são não somente dispendiosos, como também comumente são de difícil generalização, aplicáveis somente a conjuntos restritos de tarefas para os quais foram desenvolvidos [Bengio et al. 2013]. Por exemplo, caso desejasse-se criar um classificador de espécies de besouros, era preciso se contratar especialistas para definirem com precisão as características proeminentes que distinguem cada uma das espécies. Em seguida, seria preciso criar programas para extração destas características, e, só então, o modelo supervisionado poderia ser treinado, utilizando os vetores de características como representação das imagens dos besouros.

O aprendizado não supervisionado de características (às vezes também chamado de *aprendizado de representações*) tem o objetivo de criar representações úteis para o treinamento de modelos de AM de forma automática. A seguir será explicado o DAE (Denoising Auto-Encoder), que foi a principal técnica de extração de características utilizada neste trabalho.

Denoising Auto-Encoder

Um autocodificador consiste em uma rede neural treinada para minimizar o erro de reconstrução dos dados de entrada. Para impedir que a rede obtenha uma reconstrução perfeita simplesmente através da mera "memorização" dos dados, é preciso se adicionar algum tipo de restrição ao que pode ser aprendido (*regularização*).

Há várias formas propostas de regularização. A utilização de um número de neurônios na camada escondida inferior ao número de neurônios na camada de entrada é o suficiente para obrigar a rede a identificar padrões recorrentes nos dados de entrada para assim poder representá-los de forma compactada na camada escondida. Este tipo de restrição na prática não é forte o suficiente para gerar representações muito boas.

Outros tipos de regularização obrigam a rede a identificar padrões mais interessantes. O DAE (*Denoising Auto-Encoder*), introduzido por [Vincent et al. 2008], consiste em um autocodificador treinado para reconstruir os dados de entrada a partir de uma versão dos dados corrompida por ruído, como ilustrado na Figura 2.8. Ao se adicionar ruído

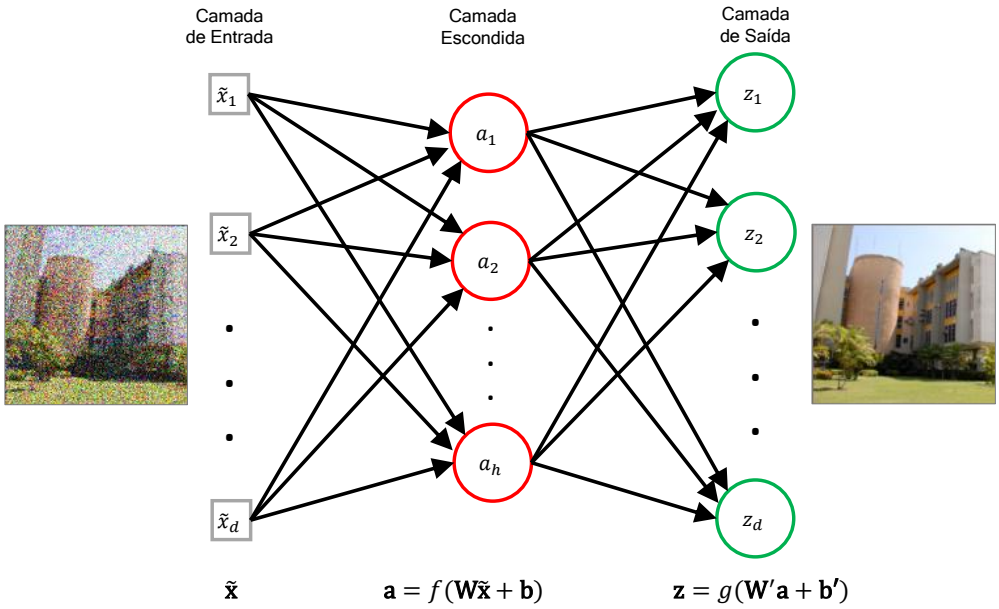

Figura 2.8: Exemplo de um DAE. A rede neural recebe uma versão dos dados de entrada corrompida por ruído e é treinada para minimizar o erro cometido ao se reconstruir os dados originais. Para isto, a rede é obrigada a identificar as regularidades frequentes no conjunto de dados de treino.

aos dados de entrada, a rede é obrigada a "preencher os buracos", e assim extrair as regularidades existentes, para assim poder reduzir o erro de reconstrução.

Mais formalmente, seja d o número de dimensões dos dados de entrada (e.g. 784 em imagens 28x28), $\tilde{\mathbf{x}} \in [0,1]^d$ uma versão corrompida do exemplar $\mathbf{x} \in [0,1]^d$, $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{W}', \mathbf{b}'\}$ os parâmetros do codificador e do decodificador, e n o número de exemplares nos dados de treino, a reconstrução calculada pela rede é dada por

$$\mathbf{z} = g(\mathbf{W}' f(\mathbf{W}\tilde{\mathbf{x}} + \mathbf{b}) + \mathbf{b}') \tag{2.14}$$

na qual f e g são as funções de ativação. Escolhas usuais destas funções são: a função logística $\sigma(z) = (1 + exp(-z))^{-1}$ para ambos, codificador e decodificador; e a função de ativação linear para o decodificador, quando os dados têm valores reais. Os parâmetros da rede são otimizados tal que

$$\theta^* = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} L\left(\mathbf{x}^{(\mathbf{i})}, \mathbf{z}^{(\mathbf{i})}\right) \tag{2.15}$$

onde L é uma *função objetivo*, usualmente o erro quadrático $L(\mathbf{x}, \mathbf{z}) = \|\mathbf{x} - \mathbf{z}\|^2$ para dados reais, e a *entropia cruzada* da reconstrução

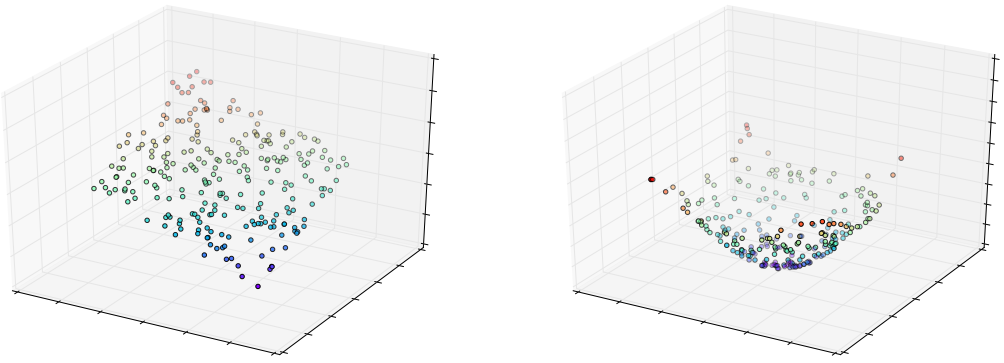

Figura 2.9: Comparação entre PCA e técnica de redução de dimensionalidade não linear. Embora os dados representados nos dois gráficos possuam três dimensões, eles se concentram em torno de *variedades*[1] bidimensionais. Na imagem da esquerda, é possível se achar as principais direções de variações dos dados utilizando PCA. Na imagem da direita, é preciso se utilizar uma técnica de redução de dimensionalidade não linear, como autocodificadores.

$$L_H(\mathbf{x},\mathbf{z}) = -\sum_{i=1}^{d} \mathbf{x_i} \log \mathbf{z_i} + (1-\mathbf{x_i}) \log (1-\mathbf{z_i}) \tag{2.16}$$

para dados binários (ou aproximadamente binários). Para se gerar a versão corrompida dos dados de entrada $\tilde{\mathbf{x}}$, uma opção comum é realizar uma operação XOR do valor $\mathbf{x}$ de entrada com uma máscara binária, na qual cada elemento tem probabilidade p de ser igual a um.

Principal Component Analysis

A técnica de redução de dimensionalidade PCA (*Principal Component Analysis*) é provavelmente a técnica mais antiga de extração de características [Bengio et al. 2013] e possui uma relação muito próxima com autocodificadores, descritos na seção anterior. Enquanto PCA é capaz somente de aprender uma transformação linear, que identifica as direções de maior variação presentes nos dados, autocodificadores são capazes de aprender transformações não lineares, como ilustrado na Figura 2.9. Autocodificadores podem ser considerados, portanto, uma generalização não linear de PCA [Hinton et al. 2006].

O poder superior de técnicas de redução de dimensionalidade não lineares, tal qual autocodificadores, decorre não somente da capacidade delas de identificar variedades não

[1] Enquanto em topologia o termo "variedade" (*manifold*, em inglês) designa de forma rigorosa uma generalização da ideia de superfície, em Aprendizado de Máquina o termo tem sido utilizado menos formalmente, com o significado aproximado de "sub-variedade" [Bengio et al. 2014].

lineares de baixa dimensionalidade escondidos nos dados, mas também por ser possível criar transformações de complexidade crescente através de múltiplas transformações em sequência. O resultado de duas transformações lineares seguidas é uma transformação linear, portanto a aplicação em camadas de PCA não é capaz de gerar resultados superiores aos da utilização de uma única camada. O mesmo não acontece com transformações não lineares, portanto a composição de múltiplas transformações é capaz de gerar funções de complexidades crescentes.

2.2.2 Algoritmos de Agrupamento

Algoritmos de agrupamento têm o objetivo de dividir os elementos de um conjunto de dados em subconjuntos compostos por elementos "similares". A similaridade é definida por uma *função de distância*, que mede o quão diferentes são dois elementos. Diferentes estratégias de agrupamentos conduzem a agrupamentos com formatos diversos. Em geral, contudo, os agrupamentos encontrados consistem em conjuntos de pontos vizinhos localizados em regiões de alta densidade.

O k-médias é um dos mais utilizados e mais antigos algoritmos de agrupamentos. O algoritmo é inicializado escolhendo-se k pontos aleatórios como centroides de k agrupamentos. Em cada passo do algoritmo, cada ponto do conjunto de dados é atribuído ao agrupamento com centroide mais próximo e, em seguida, novos centroides são recalculados. Este procedimento é repetido até que algum critério de detecção de convergência seja atingido. Mais detalhes sobre o algoritmo podem ser obtidos em [Rajaraman & Ullman 2014].

Um requisito importante para que algoritmos de agrupamento sejam empregados na abordagem proposta, é que a categoria mais frequente de cada agrupamento possa ser determinada a partir de um pequeno número de amostras. Algoritmos que dão origem a agrupamentos com formatos muito irregulares dificultam a escolha dos pontos cujas classes representam as classes mais frequentes. Algoritmos que geram agrupamentos com formatos regulares, tal qual o formato de células de Voronoi gerados pelo k-médias, facilitam essa escolha. Ao se utilizar o k-médias, a opção mais óbvia é escolher o ponto mais próximo ao centroide de cada agrupamento como amostra representativa da categoria mais frequente.

Outro aspecto importante que torna o algoritmo k-médias adequada para uso na abordagem proposta é sua eficiência[2]. Um algoritmo com complexidade espacial da ordem

[2]Embora o problema geral tratado pelo k-médias seja do tipo NP-difícil, na prática heurísticas baseadas no algoritmo de Lloyd encontram soluções quase ótimas de forma extremamente eficiente [Ostrovsky et al. 2006].

de $O(n^2)$ pode se tornar rapidamente inviável, mesmo para tamanhos moderados de n. Os experimentos realizados utilizaram o conjunto de dados MNIST, que possui 60.000 exemplares. Neste caso, o algoritmo de agrupamento hierárquico, por exemplo, se torna pouco viável na prática por exigir a criação de uma matriz de distâncias com cerca de 4 bilhões de elementos.

Inicialização do k-médias

A qualidade dos agrupamentos encontrados pelo k-médias pode variar bastante em cada execução, devido à natureza aleatória da escolha dos centroides iniciais. Várias técnicas foram desenvolvidas para se escolher estes pontos, de forma a maximizar as chances de se obter bons agrupamentos. O método de inicialização k-means++ [Arthur & Vassilvitskii 2007] foi utilizado nos experimentos descritos no Capítulo 4. Ele se difere de uma mera inicialização aleatória por escolher como centroides pontos bastante espalhados pelo conjunto de dados.

Mesmo ao se utilizar uma técnica de inicialização tal qual o k-means++, é possível ainda se obter melhores resultados ao se executar múltiplas vezes o algoritmo e em seguida escolher o agrupamento gerado pela execução que obteve o menor valor da função de distorção, dada por

$$\sum_{i=0}^{N} \|\mathbf{x_i} - \mu_{\mathbf{x_i}}\|^2 \tag{2.17}$$

onde N é o número de pontos e $\mu_{\mathbf{x_i}}$ é o centroide do agrupamento ao qual $\mathbf{x_i}$ pertence. O uso do método de inicialização k-means++ e a execução do algoritmo com múltiplas inicializações, utilizando a execução com menor valor da função de distorção, resulta em melhoras significativas no desempenho final da metodologia proposta, como será mostrado no Capítulo 5.

Embora o algoritmo k-médias tenha sido o principal algoritmo de agrupamentos utilizado nos experimentos, acredita-se que haja outras opções viáveis, que devem ser exploradas em trabalhos futuros. Algumas investigações preliminares foram realizadas com o k-médias esférico, que é similar ao k-medias, mas utiliza a distância cosseno como medida de dissimilaridade, ao invés da distância euclidiana. Mais considerações sobre a inadequação da distância euclidiana e utilização de outros algoritmos de agrupamento serão feitas no Capítulo 6.

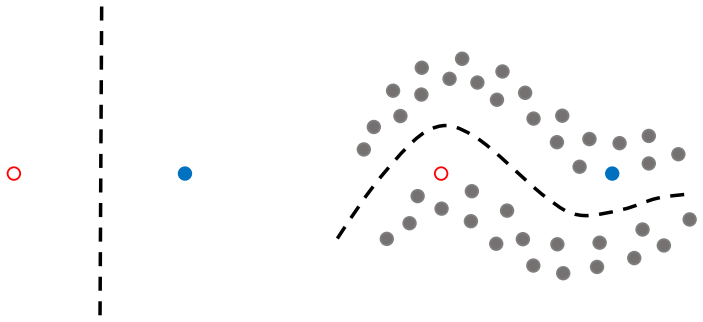

Figura 2.10: Exemplo de Aprendizado Semi-Supervisionado. Na imagem da esquerda um algoritmo de aprendizado supervisionado é treinado com apenas dois exemplos de treino. A hipótese $h(\mathbf{x})$ inferida consiste em uma linha reta. Na imagem da figura da direita, aos dois exemplos de treino rotulados é adicionado um conjunto de dados não rotulados (círculos em cinza), que permite ao algoritmo de aprendizado semi-supervisionado identificar a estrutura presente nos dados e assim inferir uma hipótese com um contorno mais complexo.

Avaliação de Agrupamentos

Enquanto a avaliação de algoritmos de classificação é relativamente simples, a avaliação de algoritmos de agrupamento impõe algumas dificuldades. Considerando que objetiva-se obter agrupamentos homogêneos (isto é, que possuem membros somente de uma classe) para posterior classificação de seus elementos, a métrica que melhor reflete este objetivo é a *pureza*, que é um *critério externo* (isto é, compara as atribuições a agrupamentos feitas pelo algoritmo com as classes reais dos dados), aplicável mesmo quando o número de agrupamentos identificados é diferente do número de classes presentes nos dados. Ela é dada por

$$Pureza = \frac{1}{N}\sum_{k}\max_{j} n_j^k \tag{2.18}$$

onde N é o número total de exemplares de treino e n_j^k é o número de pontos no agrupamento k pertencentes à classe j. Se a classe mais frequente em cada agrupamento for tratada como a classe atribuída por um modelo preditivo a cada elemento do mesmo agrupamento, essa métrica é equivalente à noção usual de acurácia de classificação. Como se está interessado principalmente na predição final, o termo acurácia será utilizado como sinônimo de pureza.

2.3 Aprendizado Semi-Supervisionado

Algoritmos de aprendizado semi-supervisionado, assim como algoritmos de aprendizado supervisionado, têm o objetivo de, a partir de um conjunto de dados de treino, inferir uma hipótese $h(\mathbf{x})$, que associa a cada exemplar $\mathbf{x}$ o respectivo valor de saída y. No aprendizado semi-supervisionado, no entanto, o conjunto de dados de treino é composto tanto por uma porção de exemplos de treino rotulados, formados pelos pares $(\mathbf{x}, y)$, quanto por uma outra porção de exemplos não rotulados (geralmente em grande quantidade).

Na maior parte dos problemas de interesse prático, os exemplos de treino não se distribuem de forma aleatória pelo espaço vetorial no qual residem, mas sim tendem a se concentrar em regiões de maior densidade. Em aprendizado semi-supevisionado, espera-se que a estrutura identificada nos dados não rotulados ajude na escolha da hipótese $h(\mathbf{x})$ mais adequada, como ilustrado na Figura 2.10.

Neste trabalho, investigou-se um tipo particular de aprendizado semi-supervisionado, denominado de aprendizado ativo.

2.3.1 Aprendizado Ativo

Diferentemente dos algoritmos usuais de aprendizado semi-supervisionado, algoritmos de aprendizado ativo não utilizam um conjunto pré-definido de dados rotulados, mas ao invés disso, têm acesso aos rótulos de exemplos de treino através da interação com um "oráculo" durante a execução do algoritmo. Este oráculo pode ser, por exemplo, um anotador humano. Desta forma, evita-se despender recursos com a rotulação de dados de treino pouco informativos. No contexto usual de aprendizado semi-supervisionado, ao se escolher de forma aleatória os dados que serão rotulados para o treino de um classificador, é possível se escolher ao acaso exemplares muito similares e portanto pouco informativos. Algoritmos de aprendizado ativo tentam mitigar este problema, escolhendo pontos mais "promissores" para serem rotulados. Esta abordagem é particularmente útil em problemas nos quais dados não rotulados são abundantes e a obtenção de dados rotulados é cara, como por exemplo no problema de classificação de proteínas [Weston et al. 2005].

Há duas estratégias principais de aprendizado ativo. A primeira se baseia na utilização de exemplares rotulados próximos à fronteira de decisão do classificador para direcionar a busca no espaço de hipóteses de uma forma eficiente. Essa abordagem assume que pontos vizinhos à fronteira de decisão são mais informativos e dão pouco importância a pontos mais distantes [Dasgupta 2011]. A segunda estratégia é baseada na hipótese da variedade (*manifold hypothesis*, em inglês), segundo a qual os pontos de dados reais

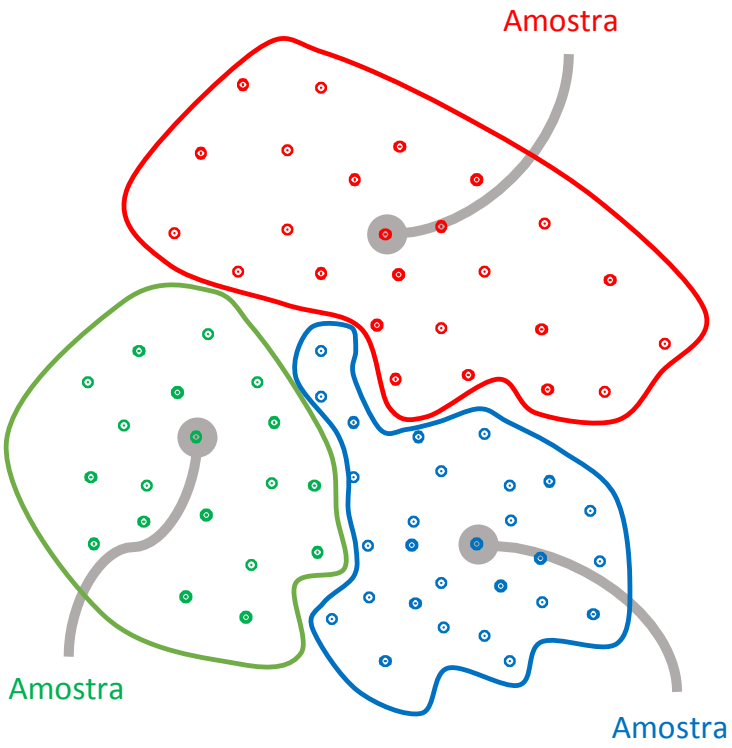

Figura 2.11: Exemplo de aprendizado ativo com agrupamentos ideais. Como o algoritmo de agrupamento identificou corretamente os agrupamentos presentes nos dados não rotulados, para se classificar corretamente todos os pontos, basta que se identifique a classe de uma única amostra por agrupamento.

(como imagens naturais ou áudio da fala) se concentram na vizinhança de uma variedade de baixa dimensionalidade imersa em um espaço de alta dimensionalidade [Nguyen & Smeulders 2004, Rifai & Dauphin 2011]. Em um cenário ideal, um algoritmo de agrupamento seria capaz de isolar os pontos de classes distintas em seus próprios agrupamentos e portanto seria suficiente se consultar o rótulo de somente um exemplar de cada agrupamento para se identificar corretamente as classes de todos os demais exemplares de treino [Dasgupta & Hsu 2008]. Este cenário é ilustrado na Figura 2.11.

Encontrar agrupamentos "puros", compostos somente por exemplares de uma única classe (como os exemplificados na Figura 2.11), é um problema extremamente difícil, ou mesmo impossível. Critérios diferentes utilizados para se julgar a similaridade (i.e. diferentes pesos dados a características diferentes) podem conduzir a agrupamentos distintos, que não correspondem necessariamente às classes que deseja-se encontrar. Por isso, em um cenário realista se obtém melhores resultados ao se encontrar um número maior de

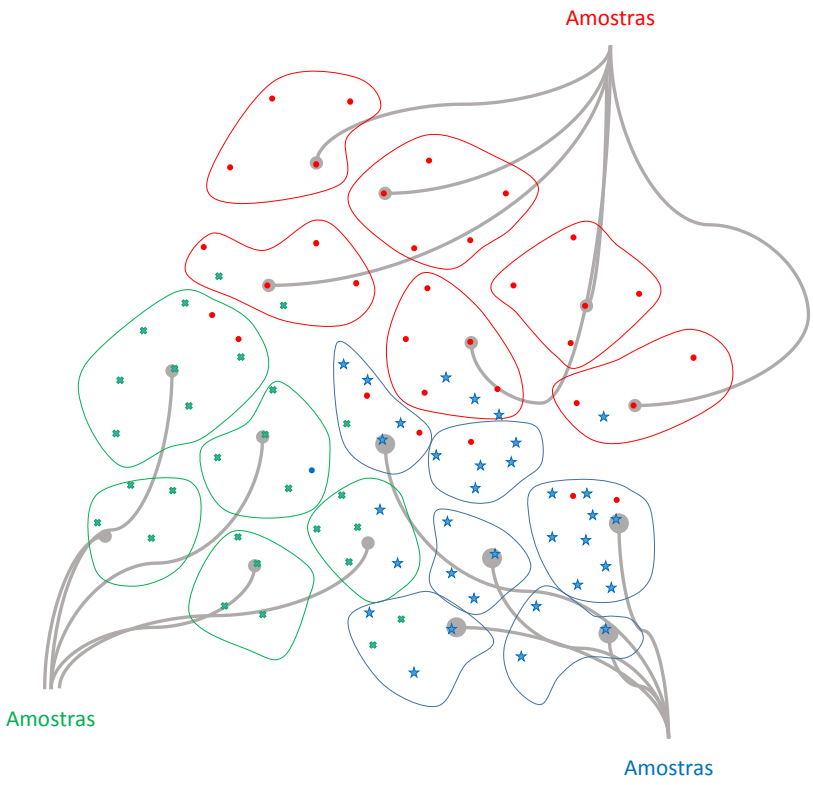

Figura 2.12: Exemplo de aprendizado ativo com agrupamentos reais. Na prática não é possível se encontrar apenas agrupamentos homogêneos, isolando os exemplares pertencem às mesmas classes. Portanto é preciso se encontrar um número maior de agrupamentos, para que se consiga achar agrupamentos tão homogêneos quanto possível e desta forma se reduzir o erro cometido ao se classificar cada ponto com a classe da amostra representativa do mesmo agrupamento.

agrupamentos, como ilustrado na Figura 2.12. Neste caso, no entanto, é preciso se utilizar um maior número de exemplos rotulados, pois para cada agrupamento é preciso se selecionar ao menos uma amostra representativa.

2.4 Deep Learning

O desenvolvimento de novas técnicas de aprendizado não supervisionado de características, como o DAE descrito na Seção 2.2.1, está fortemente ligado ao ressurgimento do interesse por redes neurais nos últimos anos.

Há fortes evidências de que arquiteturas com múltiplos níveis de representações possuem vantagens em comparação a arquiteturas razas. Resultados teóricos dos estudos da

complexidade de circuitos demonstram que qualquer função booleana pode ser representada por um circuito de portas lógicas de duas camadas [Mendelson 1997]. A implementação da maior parte destas funções requer um número exponencial de portas lógicas (em relação ao tamanho da entrada) [Wegener 1987]. Contudo, algumas destas funções podem ser calculadas de forma mais eficiente, usando um número polinomial de portas lógicas, quando permite-se que as portas lógicas sejam combinadas em múltiplas camadas, ao invés de apenas duas [Hastad 1986]. Se funções relativamente simples como funções booleanas podem ser representadas de forma mais eficiente através de representações em múltiplos níveis, pode-se argumentar *a fortiori* que funções mais complexas também o possam.

Até a publicação do trabalho pioneiro de [Hinton et al. 2006], poucos foram os casos de suceso no treino de redes neurais com mais de uma camada escondida. Em particular, ao se utilizar o algoritmo *backpropagation*, descrito brevemente na Seção 2.1.4, no treino de DNN (*Deep Neural Network*), não se conseguia obter resultados melhores do que os obtidos por um simples MLP com uma única camada escondida. Isto se dá em grande parte devido ao fenômeno dos "vanishing gradients" (vide nota de rodapé da página 12), explicado por [Hochreiter 1991].

Após 2006, várias técnicas foram desenvolvidas para treinar DNNs [Bengio et al. 2007, Hinton & Salakhutdinov 2006, Ranzato et al. 2007]. Estas técnicas utilizam um princípio comum: primeiro os parâmetros de cada camada são inicializados com uma técnica de aprendizado não supervisionado (pré-treino) e, em seguida, as camadas são treinadas simultaneamente de forma supervisionada (*fine-tuning*).

Atualmente a inicialização de parâmetros com pré-treino já não é uma técnica comumente utilizada em sistemas do estado da arte. A disponibilidade de quantidades moderadas de dados rotulados e de GPUs (*Graphics Processor Unit*) com alto poder de processamento a baixo custo possibilita o treino efetivo de redes neurais com múltiplas camadas com o algoritmo *backpropagation* sem a necessidade de realização de pré-treino. O pré-treino, contudo, ainda é aconselhável em casos nos quais a quantidade de dados não rotulados é muito superior à de rotulados [Paine et al. 2015].

Deep Learning é uma área de pesquisa muito ativa atualmente. Revisões com profundidade sobre o assunto podem ser encontradas em [Bengio et al. 2013, Bengio 2009, Deng & Yu 2013]. A seguir serão explicadas duas técnicas de *Deep Learning* utilizadas neste trabalho.

2.4.1 Stacked Denoising Autoencoders (SDAE)

A primeira técnica bem sucedida de treinamento de DNN utilizou RBM (*Restricted Boltzman Machine*) como técnica não supervisionada de inicialização dos parâmetros da rede [Hinton et al. 2006]. Embora RBM possa ser considerada como um tipo de rede neural, ela é mais facilmente compreendida (e normalmente descrita) como um tipo de rede bayesiana, estudada no contexto de Modelos Gráficos Probabilísticos. Logo após estes trabalhos inaugurais, Bengio et al. [Bengio et al. 2007] mostrou que autocodificadores eram uma alternativa viável à utilização de RBMs. No entanto, a taxa de acerto em problemas de classificaçao atingida por DNNs treinadas com "meros" autocodificadores (que não utilizavam restrições para o que poderia ser reconstruído além do número de neurônios nas camadas intermediárias) é inferior à atingida ao se utilizar RBM na inicialização de DNNs. Esta diferença de desempenho estimulou a criação de novos tipos de autocodificadores com restrições mais fortes. Um destes novos tipos de autocodificadores foi o DAE, descrito na Seção 2.2.1.

A utilização de DAE como bloco de construção de DNN é similar à utilização de RBM. Cada camada da rede é inicializada separadamente utilizando como valores de entrada o conjunto de dados de treino previamente transformados pelo DAE anterior. A rede neural criada por estas várias camadas de DAE recebe o nome de SDAE (*Stacked Denoising Autoencoder*). Após a inicialização dos parâmetros (pré-treino), todas as camadas são treinadas simultaneamente com o algoritmo *backpropagation* (*fine-tuning*).

A Figura 2.13, extraída de [Erhan et al. 2010], mostra visualmente os parâmetros das três camadas de um DNN treinado com o conjunto de dados InfiniteMNIST[3] antes (parte de cima) e após o *fine-tuning* (parte de baixo). Os neurônios da primeira camada se especializam no reconhecimento de padrões localizados de "traçados de caneta", os neurônios da segunda camada parecem se especializar na detecção de partes de dígitos, enquanto os neurônios da terceira camada se especializam na detecção de dígitos completos. Visualmente há muito pouca diferença entre os parâmetros da rede antes e após o *fine-tuning*. Este resultado sugere que o pré-treino inicializa os parâmetros da rede em uma região do espaço de parâmetros próxima de uma solução "boa" (na vizinha de um mínimo local com valor mais próximo ao do mínimo global), diferentemente do que ocorre com inicialização aleatória.

[3] O InfiniteMNIST é uma extensão do MNIST descrito na Seção 5.1 do qual se pode extrair um número quase ilimitado de imagens de dígitos através de transformações das imagens originais [Loosli et al. 2007]

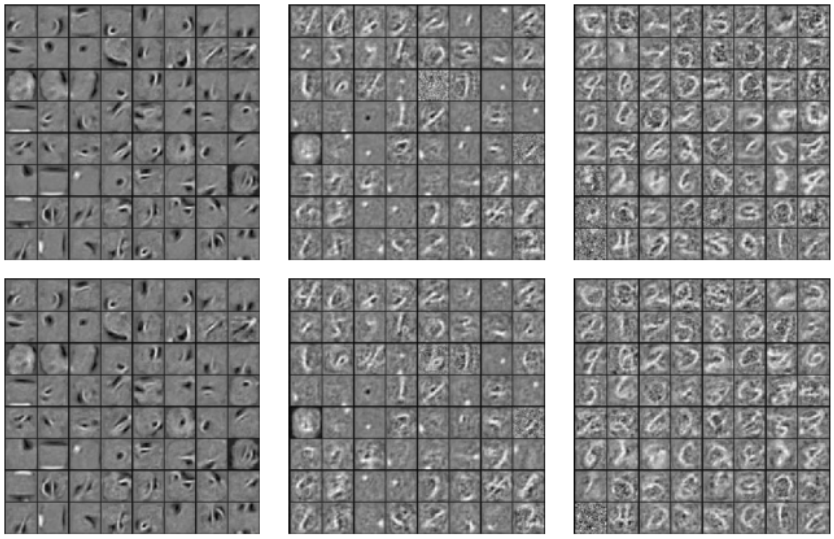

Figura 2.13: Visualização dos parâmetros de uma DNN. As figuras de cima foram geradas após o pré-treino, enquanto as de baixo foram geradas após o *fine-tuning*. As figuras da esquerda, do centro e da direita representam os parâmetros da primeira, segunda e terceira camada, respectivamente. Embora o pré-treino das camadas da DNN ilustada tenha sido feito com RBMs, os parâmetros gerados com um SDAE são qualitativamente similares. Fonte da Imagem: [Erhan et al. 2010].

2.4.2 Redes Neurais Convolucionais

Nas arquiteturas "convencionais" de redes neurais artificiais as unidades das camadas superiores à camada de entrada se conectam a todas as unidades da camada imediatamente inferior. Este padrão de conectividade completo não corresponde ao padrão encontrado no córtex cerebral. Hubel & Wiesel [1968] demostraram que os neurônios da região V1 do córtex respondem a estímulos luminosos somente de uma região restrita do campo visual, chamada de campo receptivo. Fukushima [1980] propôs um modelo de rede neural que tentava simular esta organização. Este modelo obteve alguns avanços em relação a modelos anteriores. Em particular, ele foi bem sucedido em atingir certa invariância quanto à posição dos padrões reconhecidos (isto é, o objeto reconhecido pode ocupar posições diferentes na imagem). LeCun [1998], inspirado nos trabalhos de Fukushima, desenvolveu a rede neural convolucional (ConvNet) — do inglês, *Convolutional Neural Network* —, que será descrita nesta seção. A implementação eficiente de redes neurais convolucionais profundas em GPU (*Graphical Processing Unit*) estabeleceu um dos marcos no uso de redes neurais no reconhecimento de imagens a partir da competição ILSVRC2010 (*Large Scale Visual Recognition Challenge 2010*), que foi vencida por uma rede neural convolucional profunda com larga margem em relação ao segundo colocado [Krizhevsky

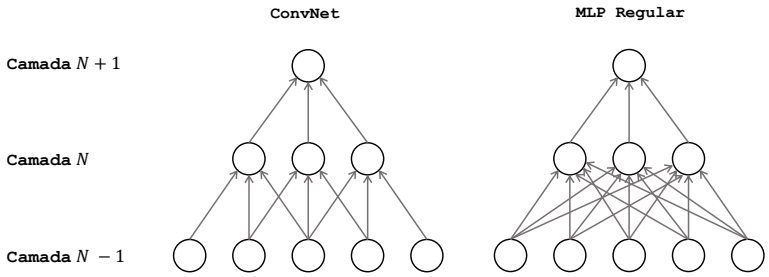

Figura 2.14: Padrão de Conectividade de redes convolucionais. Diferentemente de MLP regulares, em uma ConvNet os neurônios das camadas superiores se conectam apenas com grupos restritos de neurônios contíguos da camada inferior.

et al. 2012]. Desde então este tipo de modelo é o modelo predominante em tarefas de reconhecimento de imagens. Embora redes convolucionais não seja um dos elementos essenciais da abordagem proposta, na prática seu uso é importante para se atingir resultados competitivos em diversos *benchmarks* de AM, como os atingidos nos experimentos com reconhecimento de dígitos, descritos no Capítulo 5.

Conectividade Restrita

Conceitualmente, as redes neurais convolucionais se diferem de redes neurais convencionais primeiramente por requererem padrões de conectividade localizados, como ilustrado na Figura 2.14. Ao invés de cada neurônio se conectar com todos os neurônios das camadas inferiores, nas ConvNets os neurônios se conectam somente a grupos pequenos e contíguos de neurônios. A princípio esta limitação pode soar como uma desvantagem, contudo é fácil observar, ao menos em reconhecimento de imagens, que características relevantes de objetos tendem a abranger espaços relativamente pequenos e ser formadas por elementos conectados. Por exemplo, padrões formados por pixels distantes e não conectados dificilmente representam elementos distintivos de qualquer objeto. Tais padrões, caso identificados como "relevantes", são mais provavelmente ruídos, que levariam a rede ao sobreajuste, não sendo úteis para a identificação de imagens diferentes das utilizadas durante o treino.

Compartilhamento de Parâmetros

No reconhecimento de imagens, uma característica específica útil para a identificação de um objeto pode ocorrer em posições arbitrárias dentro do campo visual. Em camadas inferiores, por exemplo, um dado segmento de reta com uma orientação de 45º pode

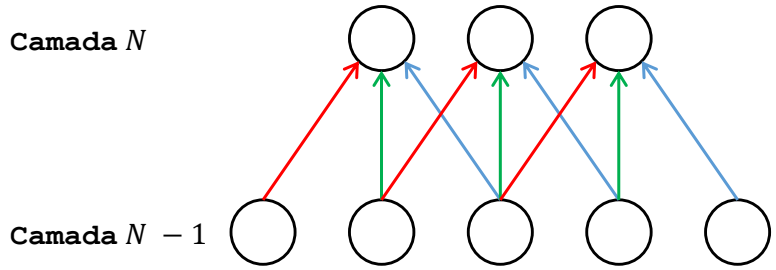

Figura 2.15: Exemplo de Compartilhamento de Parâmetros. As setas de mesma cor representam parâmetros com valores iguais. As unidades com os mesmos parâmetros identificam padrões idênticos, mas em regiões distintas da camada inferior.

ser útil em qualquer parte da imagem para compor um objeto. O mesmo se aplica a características de mais alto nível, que podem ser detectadas em posições arbitrárias na imagem. Considerando que a característica detectada por uma unidade é determinada pelos seus parâmetros, uma forma de se detectar uma mesma característica ocorrendo em qualquer posição do campo visual consiste em replicar cada unidade utilizando os mesmos parâmetros conectados a grupos distintos de unidades na camada inferior, como ilustrado na Figura 2.15.

Convoluções

O compartilhamento de parâmetros e a restrição de conectividade podem ser implementados de forma eficiente através de uma operação matemática denominada de convolução, comumente empregada em *análise funcional*. A convolução é uma operação entre duas funções e resulta em uma nova função. No caso unidimensional, ela é dada pela fórmula

$$\mathbf{y_k} = \sum_{n=0}^{N-1} \mathbf{h_n} \cdot \mathbf{x_{k-n}} \tag{2.19}$$

na qual **h** e **x** são vetores de entrada (as "funções" cujas variáveis são os índices) e **y** o vetor de saída (a "função" resultado da operação), chamado de *mapa de características*, no contexto de redes convolucionais. Enquanto **h** representa os dados de entrada, **x** representa o vetor de parâmetros replicado pelas sub-regiões do vetor de entrada (comumente chamado de *kernel de convolução*). Os valores de **x** com índices inexistentes, por convenção, costuma-se considerar como sendo iguais a zero. Abaixo, mostra-se um exemplo de cálculo de convolução para dois vetores

$$x = \begin{pmatrix} 1 & 2 & 1 & 3 \end{pmatrix}$$
$$h = \begin{pmatrix} 3 & 0 & 1 \end{pmatrix}$$

k	*0*	*0*	3	1	2	1	*0*	*0*	$\mathbf{y_k}$
0	3	0	1						$3 \cdot 1 = 3$
1		3	0	1					$1 \cdot 1 = 1$
2			3	0	1				$3 \cdot 3 + 2 \cdot 1 = 11$
3				3	0	1			$1 \cdot 3 + 1 \cdot 1 = 4$
4					3	0	1		$2 \cdot 3 = 6$
5						3	0	1	$1 \cdot 3 = 3$

No exemplo, a primeira coluna representa o valor do índice k do vetor de saída $\mathbf{y_k}$. O cabeçalho da segunda coluna representa o valor de $\mathbf{x}$ (com zeros adicionados à esquerda e à direita para suprir os índices não existentes). O restante da segunda coluna representa os valores de $\mathbf{x_{k-n}}$ na ordem invertida que "deslizam" da esquerda para a direita realizando um multiplicação ponto-a-ponto com o vetor $\mathbf{x}$. A última coluna mostra o resultado da operação, que gera o vetor (os dígitos de $\mathbf{y_k}$ são lidos de baixo para cima):

$$\begin{pmatrix} 3 & 6 & 4 & 11 & 1 & 3 \end{pmatrix}$$

Esta operação pode ser facilmente estendida para vetores bidimensionais (ou mesmo para um número maior de dimensões), como ilustrado na Figura 2.16, que mostra um exemplo de cálculo de um único elemento.

A Figura 2.17 ilustra a aplicação de dois kernels de convolução diferentes a uma mesma imagem e mostra a imagem resultante. Cada kernel salienta um aspecto importante da imagem. Alguns padrões podem ser detectados mais facilmente ao se manter somente as extremidades da imagem original, o que pode ser alcançado facilmente com um kernel com os valores ilustrados na figura. Portanto, a convolução pode ser compreendida tanto como uma forma de replicar um mesmo detector de característica para identificar características em quaisquer regiões dos dados de entrada, quanto pode ser compreendida como uma transformação dos dados de entrada para uma representação na qual seja mais fácil se identificar características relevantes.

Max Pooling

A replicação de características por todas as regiões do vetor de entrada tem o efeito colateral de aumentar muito o número de características utilizadas. Esse aumento pode

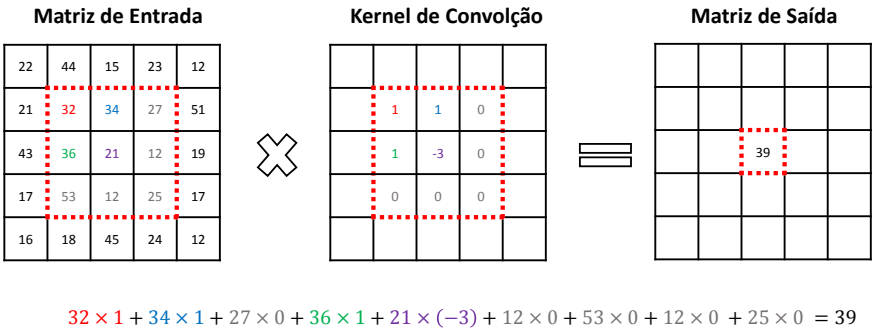

Figura 2.16: Exemplo de cálculo de um elemento em uma convolução bidimensional. A borda vermelha delimita uma área com tamanho igual ao tamanho do kernel de convolução (3x3). O elemento central na matriz de saída é resultado da multiplicação ponto-a-ponto dos elementos com posições correspondestes nas duas matrizes.

facilmente aumentar demasiadamente o custo computacional. Para contornar este problema, comumente aplica-se, após cada camada de convolução, uma operação de sub-amostragem. Essa sub-amostragem pode ser feita de diversas formas, contudo a mais comum é denominada de *max-pooling*, na qual a imagem de entrada é dividia em regiões disjuntas e cada região é substituída pelo pixel de maior valor dentro da região. Embora muita informação se perca neste processo, devido ao fato de haver uma correlação muito forte entre pixels adjacentes, a informação restante ainda é o suficiente para caracterizar a imagem original. Na prática esta operação tem mostrado bons resultados.

LeNet5

Para ilustrar a arquitetura de uma rede neural convolucional, descrevemos brevemente a arquitetura da LeNet5, desenvolvida por [LeCun et al. 1998] para o reconhecimento de caracteres. A arquitetura da rede é mostrada na Figura 2.18. A rede possui duas camadas iniciais de convoluções seguidas por camadas sub-amostragem (max-pooling) e uma etapa final com 3 camadas inteiramente conectadas, como em uma rede neural "tradicional", com a exceção de que no modelo original a última camada consistia em uma rede RBF (*Radial Basis Function*) ao invés de uma camada *Softmax*, que é mais comumente utilizada [Bishop 2007]. Um sistema baseado neste modelo foi empregado pelo serviço postal americano para o reconhecimento dos dígitos das caixas postais. A taxa de acertos deste modelo da década de 90 é até hoje similar às obtidas pelos modelos mais bem sucedidos já criados no reconhecimento de dígitos do MNIST.

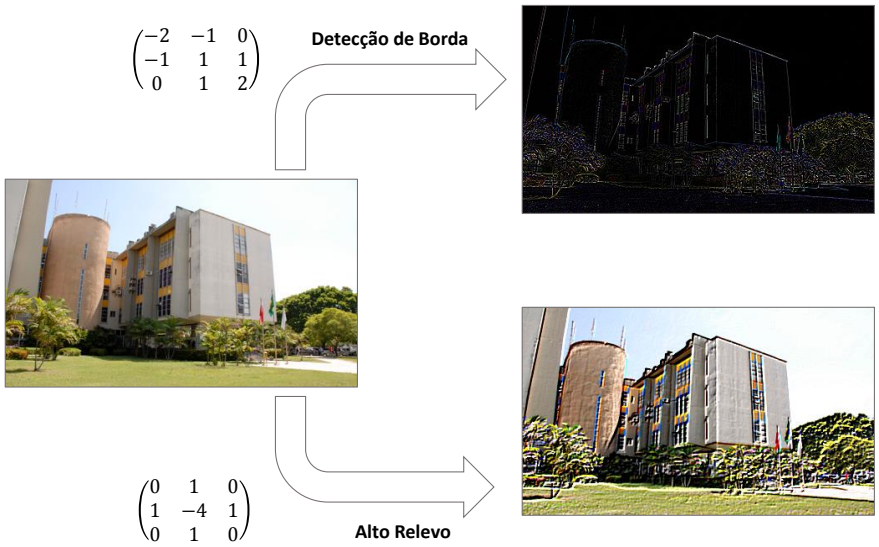

Figura 2.17: Exemplo de aplicação de kernels de convolução. A aplicação do kernel em uma imagem resulta na substituição de cada pixel por uma soma ponderada dos pixels ao seu redor, no qual os pesos são definidos pela matriz de convolução (kernel). No exemplo inferior o kernel gera um efeito de "alto relevo" enquanto no exemplo superior o efeito é de detecção de extremidades.

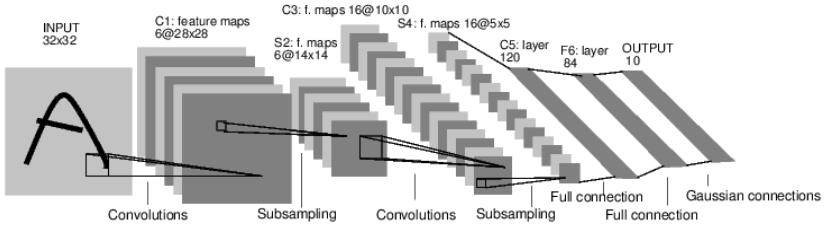

Figura 2.18: Arquitetura da rede neural convolucional LeNet5. Na primeira camada, são utilizados 6 detectores de características convolucionados que geram 6 mapas de características de tamanho 28x28 que em seguida são sub-amostrados para 14x14. Este processo é repedido mais uma vez com valores distintos e no final utiliza-se três camadas completamente conectadas, como as usualmente utilizadas em MLPs. Fonte da Imagem: [LeCun et al. 1998].

Capítulo 3

Trabalhos Relacionados

No Capítulo 2 fez-se uma revisão dos "componentes" utilizados na abordagem proposta. Neste capítulo, os trabalhos de aprendizado semi-supervisionado "concorrentes" são descritos, isto é, as demais técnicas de aprendizado ativo e de aprendizado semi-supervisionado que têm como objetivo a criação de modelos preditivos com o menor uso possível de dados rotulados.

3.1 Aprendizado Ativo

Como descrito na seção 2.3.1, há duas estratégias principais de aprendizado ativo atualmente investigadas: a que explora a *busca no espaço de hipóteses* direcionada pelos dados rotulados, e a que explora a *estrutura de agrupamentos* existentes nos dados (estratégia utilizada neste trabalho). Embora essas duas estratégias se baseiem em princípios comuns e compartilhem muitos elementos teóricos, os trabalhos de aprendizado ativo existentes podem facilmente se classificados como pertencentes a uma das duas.

3.1.1 Busca no Espaço de Hipóteses

Lewis & Gale [1994] criou uma das primeiras técnicas de aprendizado ativo: o *framework uncertainty sampling*. Nesse *framework* o treinamento do classificador é feito de forma iterativa. Em cada passo um novo modelo é treinado com os dados rotulados disponíveis. Cada novo modelo define uma nova fronteira de decisão, que separa as classes em regiões distintas. Os pontos não rotulados mais próximos a esta fronteira são justamente aqueles cujas classificações são as mais incertas, isto é, eles são os pontos com maior probabilidade de serem erroneamente classificados. Esses pontos são, portanto, escolhidos para anotação, pois restringem mais rapidamente o espaço de hipóteses válidas.

No treinamento de um classificador linear binário, por exemplo, as hipóteses válidas consistem em hiperplanos que dividem o espaço em duas regiões, separando os pontos pertencentes às duas classes. Para se determinar com mais precisão a posição do hiperplano (determinada pelos parâmetros do modelo), os pontos não-rotulados escolhidos para serem anotados em cada passo são aqueles mais próximos ao hiperplano, para os quais o classificador atribui menor probabilidade de pertencimento a uma das duas classes. Um problema desta abordagem, em tarefas de classificação com mais de duas classes, é que ela utiliza somente a informação de pertencimento à classe mais provável para determinar se um ponto é considerado com alta probabilidade ou não. Pontos que recebem probabilidades altas de duas classes, embora intuitivamente sugiram alta indecisão, são considerados como pontos bastante "certos". Uma variação desse *framework* que reduz esse problema foi proposta por [Scheffer et al. 2001], na qual o ponto considerado o "mais incerto" é aquele com a maior diferença entre as probabilidades de pertencimento às duas classes mais prováveis[1]. Princípios similares foram utilizados por [Tong & Koller 2001] no treinamento de SVMs (*Support Vector Machine*), cujo modelo de aprendizado ativo utiliza justamente os pontos mais próximos à fronteira de decisão durante cada iteração do algoritmo de aprendizado.

Embora essas abordagens de aprendizado ativo tenham obtido bons resultados, os autores deste trabalho acreditam que elas estão fortemente sujeitas à "maldição da dimensionalidade". Em dados de alta dimensionalidade, à medida que o número de dimensões aumenta, o volume do espaço cresce exponencialmente, de tal forma que mesmo uma quantidade "grande" de dados não rotulados não seria capaz de abranger uma fração significativa dele. Pontos distantes da fronteira de decisão podem na verdade ser muito mais informativos do que aparentam à primeira vista. A abordagem proposta neste trabalho é menos sujeita a este problema, pois a conversão dos dados para o espaço de características tem como resultado projetá-los para a vizinhança de uma variedade de baixa dimensionalidade, e, portanto, uma quantidade menor de pontos é necessária para se obter uma amostra significativa da totalidade do espaço efetivamente ocupado.

3.1.2 Estrutura de Agrupamentos

A linha de pesquisa em aprendizado ativo que se baseia na exploração da estrutura de agrupamentos tem sido relativamente pouco investigada. À título de exemplo, a revisão de literatura de Settles [2010] faz poucas menções a ela. Um dos primeiros trabalhos

[1]Este mesmo critério de determinação dos pontos "mais incertos" foi proposto independentemente na seção 4.3, sem que os autores tivessem conhecimento do trabalho citado.

nesta área foi feito por [Xu et al. 2003], que utilizou o k-médias para colher amostras representativas dos dados não rotulados, que eram usados para aumentar a velocidade de convergência do treinamento de SVMs. O trabalho de Zhu et al. [2003] explorou a estrutura de variedade (*manifold structure*) presente nos dados através da modelagem da distribuição de probabilidade dos pontos pertencerem às classes como um GRF (*Gaussian Random Field*) construído sobre um grafo cujos vértices representam os exemplos de treino e as arestas codificam as distâncias entre eles. Nguyen et al. [2004] tratou de alguns problemas dos dois trabalhos anteriores através da adoção de medidas para evitar rotular repetidamente amostras dos mesmos agrupamentos. Eles utilizaram estas amostras representativas para treinar um classificador discriminativo para obter os rótulos dos dados não rotulados restantes (diferentemente deste trabalho, que utiliza um classificador baseado em distância para propagar os rótulos). A ideia de utilizar SDAE antes da aplicação de um algoritmo de agrupamento foi explorada por Lefakis & Wiring [2007]. Esse trabalho focou, contudo, no uso de representações sub-completas (com camadas escondidas com número de unidades inferior ao número de dimensões do vetor de entrada) e autocodificadores "regulares" (sem a utilização de regularização adicional, como a inserção de ruído do DAE), que usualmente aprendem características menos robustas.

3.2 Aprendizado Semi-Supervisionado

Os algoritmos "tradicionais" de aprendizado semi-supervisionado (isto é, algoritmos não "ativos") em muitos cenários são alternativas viáveis à abordagem proposta. Em problemas nos quais já se possui previamente quantidades modestas de dados rotulados, em adição a grandes quantidades de dados não rotulados, ambas alternativas podem ser promissoras. Além disto, as duas abordagens compartilham diversos conceitos e premissas.

Uma das técnicas mais antigas de aprendizado semi-supervisionado é o "auto-aprendizado" (do inglês, *self-learning*), no qual o algoritmo incialmente treina um modelo preditivo utilizando somente os dados rotulados disponíveis e em seguida utiliza este modelo para classificar os dados não rotulados. O conjunto de dados rotulados obtido assim é utilizado no treinamento de um novo classificador, que em seguida é utilizado para reclassificar os dados. O processo é repetido iterativamente até que algum critério de parada seja atingido. Variações desse algoritmo foram propostas por [Scudder 1965] e [Fralick 1967].

O trabalho de Vapnick & Chervonenkis [1974], em russo, introduziu o princípio da inferência transdutiva (ou princípio de Vapnik) — conforme consta em [Chapelle et al. 2006] —, segundo o qual "quando se está tentando resolver um problema, não se deve tentar resolver um problema mais complexo como um passo intermediário".

Este princípio defende que em um contexto de aprendizado supervisionado ou semi-supervisinoado, quando se possui previamente os dados de teste que se deseja classificar, o algoritmo de classificação deve classifica-los diretamente, sem para isso criar um modelo preditivo genérico que deva funcionar para dados de testes arbitrários. Este princípio é utilizado na TSVM (*Transductive Support Vector Machine*), introduzido por [Vapnik 1998].

Um número grande de abordagens de aprendizado-supervisionado foi proposto nas últimas décadas. Mais recentemente uma nova tendência parece ter surgido com advento de algoritmos de deep learning e aprendizado de características não-supervisionado (tendência da qual este trabalho faz parte). Rifai [2011] combinou vários princípios de aprendizado semi-supervisionado em uma nova técnica de treinamento de redes neurais de múltiplas camadas utilizando audocodificadores contrativos de ordem superior (*high-order contractive auto-encoder*) como técnica de extração de características e "pré-treino", obtendo bons resultados. Weston et al. [2012] propuseram o treino de arquiteturas profundas utilizando simultaneamente alguma técnica de aprendizado não-supervisionado "qualquer" como tarefa auxiliar. A ideia principal desta abordagem é treinar o modelo de forma supervisionada com os dados rotulados disponíveis enquanto ao mesmo tempo se realiza o "pré-treino" com os dados não rotulados. Lee [2013] propôs uma forma alternativa de utilizar dados rotulados e não rotulados simultaneamente durante o treino de redes neurais profundas. Em cada passo do algoritmo de otimização (descida de gradiente) o modelo obtido no passo anterior é utilizado na classificação dos dados não rotulados. Os dados assim obtidos são utilizados no próximo passo como se seus rótulos fossem verdadeiros. Esse trabalho, assim como a descrita nesta pesquisa, também utiliza SDAE como técnica de aprendizado de características.

Capítulo 4

Materiais e Métodos

A distância euclidiana é uma métrica comumente utilizada em algoritmos de AM, principalmente como *medida de dissimilaridade* em algoritmos de agrupamento e parte da função objetivo em algoritmos supervisionados. Ela é dada por

$$d(\mathbf{x}, \mathbf{p}) = \sqrt{\sum_{i=1}^{N} (\mathbf{x_i} - \mathbf{p_i})} \tag{4.1}$$

onde $\mathbf{x}$ e $\mathbf{p}$ são dois vetores de N dimensões. É fácil constatar que a Equação 4.1 nem sempre captura corretamente a nossa noção intuitiva de dissimilaridade entre dois objetos. É ilustrado na Figura 4.1 um exemplo de imagem de um dígito deslocado ligeiramente para a direita. Um pequeno deslocamento como este é o suficiente para que a imagem deixe de compartilhar a maior parte dos pixels em comum com a imagem original. Embora as imagens continuem a retratar o mesmo objeto, a distância medida entre elas pela Equação 4.1 aumenta bastante.

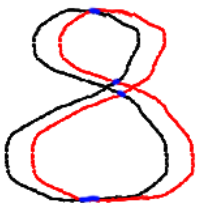

Figura 4.1: Efeito do deslocamento de um dígito na distância euclidiana. O dígito ao ser deslocado ligeiramente para a direita deixa de compartilhar grande parte dos pixels com a imagem original. A distância euclidiana das duas figuras aumenta bastante, embora ainda representem o mesmo objeto.

Neste trabalho, explora-se o uso de aprendizado não supervisionado de características para se converter os dados do espaço original para um espaço de características, no qual a distância euclidiana representa mais fielmente nossa noção intuitiva de dissimilaridade. Ao se comparar as imagens de dois objetos, é mais fácil dizer se eles são similares ou não através da comparação das características presentes (ou ausentes) em ambos do que através da comparação de seus pixels brutos (espaço original). Após a conversão para o espaço de características, espera-se consequentemente que os agrupamentos encontrados sejam mais homogêneos. Em um caso extremo, ao se achar agrupamentos puros (com elementos de uma única classe), o rótulo de somente uma amostra de cada agrupamento seria suficiente para se classificar corretamente todo o conjunto de dados.

Com base nesta constatação, neste trabalho propõe-se a estratégia de aprendizado ativo resumida nos seguintes passos:

1. Executar aprendizado não supervisionado de características em conjunto de dados não rotulados e convertê-los para o espaço de características.
2. Executar algoritmo de agrupamentos no espaço de características.
3. Classificar as amostras representativas de cada agrupamento (por exemplo, através da interação com um anotador humano).
4. Classificar os demais exemplos do conjunto de dados não rotulados utilizando as amostras representativas como "dados de treino" de um classificador baseado em distância.
5. Determinar a probabilidade de classificação do passo anterior e remover os dados classificados com baixa probabilidade (passo opcional).
6. Treinar um modelo supervisionado utilizando os dados classificados no passo 4 como dados de treino.

Esta estratégia foi inspirada no mecanismo pelo qual seres humanos identificam e aprendem novos conceitos. O cérebro parece ter mecanismos eficientes especializados no reconhecimento de padrões de forma não supervisionada. Via de regra, quando um novo conceito é aprendido, primeiramente as características fundamentais que o constituem são identificadas, e somente depois é que o nome do conceito é aprendido. A partir da identificação de um único exemplar da nova categoria aprendida, o ser humano é capaz de identificar outros representantes do mesmo conceito através da mera comparação de características comuns.

Neste capítulo são descritos com detalhes da abordagem proposta, junto com o experimentos realizados para validar os resultados intermediários obtidos em cada passo. Todos os experimentos foram realizados com o conjunto de dados MNIST, descrito na Seção

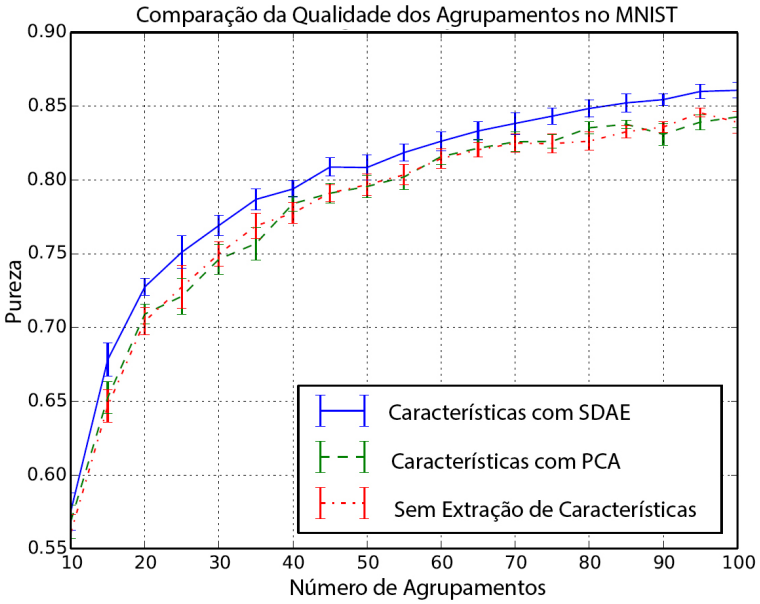

Figura 4.2: Comparação da pureza dos agrupamentos de acordo com as características usadas. A pureza dos agrupamentos encontrados pelo k-médias, executado com 10.000 exemplos do conjunto de treino do MNIST escolhidos ao acaso, aumenta à medida que número de agrupamentos k aumenta. A extração de características com PCA não gerou resultados melhores que a utilização dos dados brutos, enquanto que a extração de características com SDAE conduziu de forma consistente a agrupamentos mais homogêneos. Cada ponto no gráfico representa a média de 10 execuções do algoritmo k-médias com inicialização aleatória. A barra de erro mostra o intervalo de confianaça de 95%.

5.1. As implementações dos algoritmos de AM utilizadas foram as disponibilizadas nas bibliotecas *pylearn2* [Goodfellow et al. 2013] e *scikit-learn* [Pedregosa et al. 2011], disponíveis para a linguagem de programação Python.

4.1 Conversão para Espaço de Características

Como técnica de aprendizado não supervisionado de características, utilizou-se o SDAE, descrito na Seção 2.4.1. Para determinar se a conversão dos dados para o espaço de características com SDAE resulta em aumento da pureza dos agrupamentos encontrado pelo k-médias, foram realizados experimentos utilizando como *grupo de controle* os dados brutos (dados no espaço original) e os dados convertidos com PCA.

A Figura 4.2 mostra uma comparação da acurácia obtida na execução do k-médias ao se utilizar as imagens dos dígitos do MNIST em suas formas brutas (pixels), com a extração de características utilizando PCA e através da extração de características com

um SDAE.

O algoritmo k-médias foi executado com valores crescentes do número de agrupamentos (k). Após cada execução, todos os pontos pertencentes a um mesmo agrupamento foram classificados com a classe mais frequente dentro do agrupamento. Para valores maiores de k, a acurácia total cresce, independentemente das características utilizadas (dados brutos, PCA ou SDAE), contudo a acurácia obtida com SDAE é consistentemente superior para todos os valores de k, mostrando que a extração não supervisionada de características é capaz de aumentar a pureza dos algoritmos de agrupamento. O gráfico da Figura 4.2 mostra os resultados deste experimento.

Para determinar um número ótimo de componentes no PCA, realizou-se experimentos adicionais com o número de componentes variando de 50 a 500. Não foram encontradas diferenças significativas nos resultados. Portanto, por convenção optou-se por utilizar em todos os experimentos PCA com 300 componentes.

Considerando que foi assumida a premissa de que dados rotulados são escassos ou difíceis de se obter, caso houvesse dados adicionais para otimização de hiperparâmetros, estes seriam melhor empregados no treinamento do modelo em si. Portanto, durante o treinamento do SDAE foi utilizada uma quantidade reduzida de hiperparâmetros com o intuito de evitar sobreajuste (uma ou duas camadas; níveis de corrupção 0.1 e 0.2; larguras de camada 300 e 500). Em todos os experimentos, utilizou-se a configuração de parâmetros que gerou melhores resultados dentre o espaço de parâmetros explorado (SDAE com duas camadas, nível de corrupção de 0.2 e 500 unidades por camada).

4.2 Determinação de Amostras Representativas

Na Seção 4.1 mostrou-se que a extração de características com SDAE é capaz de aumentar a pureza dos agrupamentos encontrados pelo algoritmo k-médias. Para isso, aferiu-se a pureza dos agrupamentos ao se utilizar os rótulos verdadeiros de cada imagem (identificando a classe mais frequente em cada agrupamento). Contudo, objetiva-se a obtenção de um classificador em um cenário no qual estes rótulos não estão disponíveis. Como se pode determinar a classe mais frequente de cada agrupamento quando não se tem acesso a estes dados rotulados? Ao se utilizar o algoritmo de agrupamento k-médias, é natural esperar que os pontos mais centrais de cada agrupamento sejam uma boa amostra dos demais pontos em seus entornos, como ilustrado na Figura 4.3.

Para determinar o erro cometido ao se classificar os pontos de cada agrupamento com a classe do ponto mais central, foram realizados experimentos com um subconjunto de 10 mil imagens extraídas aleatoriamente do conjunto de treino do MNIST. O resultado é

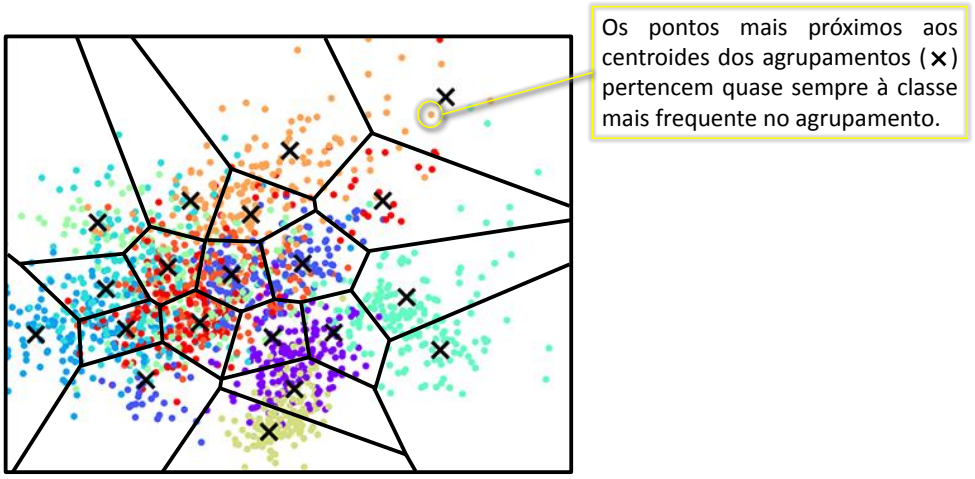

Figura 4.3: Determinação da classe mais frequente do agrupamento. A se utilizar o algoritmo k-médias em uma porção do MNIST (utilizando os dois principais componente extraídos por PCA), pode-se visualizar claramente como os exemplares mais centrais dos agrupamentos (mais próximos dos centroides) quase sempre pertencem às classes mais frequentes dentro dos agrupamentos.

mostrado na Figura 4.4. Ao se utilizar 10 agrupamentos, a diferença entre os dois procedimentos (usar a classe mais frequente e usar a classe do ponto mais central) é em média inferior a 1%. Ao se utilizar mais de 15 agrupamentos a diferença cai rapidamente para menos de 0,005%. Estes experimentos mostram que o rótulo do exemplo de treino mais central em cada agrupamento encontrado pelo k-médias é uma excelente aproximação do rótulo mais frequente no agrupamento.

4.3 Estimação de Probabilidade de Classificação

A classificação dos pontos de cada agrupamento com a classe do ponto mais central, como descrito na Seção 4.1, é equivalente à classificação dos dados com um algoritmo KNN (K-Nearest Neighbors) se utilizando esses pontos como os dados de treino. Uma desvantagem desta abordagem é que o algoritmo KNN não determina a probabilidade de pertencimento a cada classe. Em diversos contextos obter tais probabilidades pode ser desejável, por exemplo, em um cenário no qual se deseje obter um grande número de exemplares de determinadas classes, mas a tolerância a falsos positivos seja pequena. Nesta situação, poderia-se eliminar os pontos classificados com baixa probabilidade e manter os demais.

Considerando que o ponto central de cada agrupamento é um bom representativo da

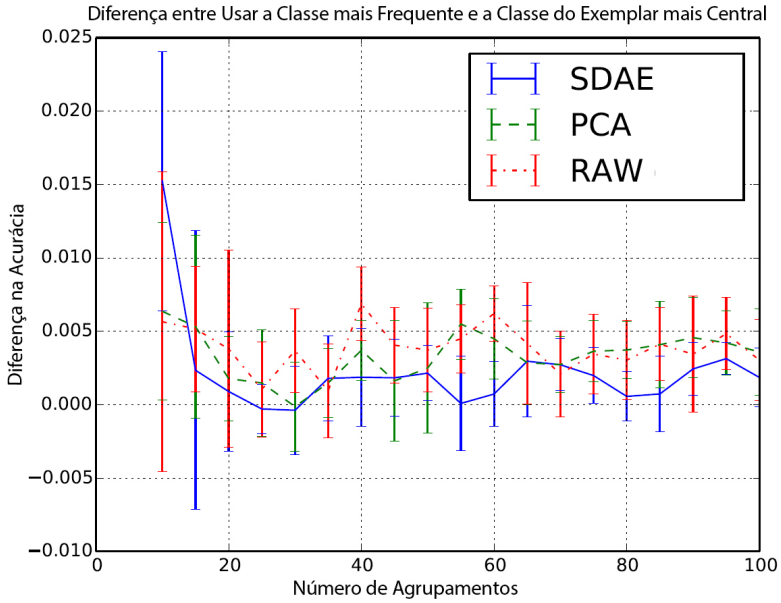

Figura 4.4: Comparação entre uso de classe mais frequente por agrupamento com o uso da classe de ponto mais próximo ao centroide. Para um número de agrupamentos um pouco maior que o número de classes (10) a diferença é próxima de zero. A diferença tende a ser ainda um pouco menor quando se utiliza SDAE para extração de características.

classe mais prevalente do agrupamento, pode-se assumir que os pontos mais centrais devam ser classificados com maior probabilidade que os pontos mais periféricos. No entanto, pontos na periferia de dois agrupamentos de mesma classe, embora estejam na periferia, também devem ter alta probabilidade de serem classificados corretamente, pois a existência de dois agrupamentos vizinhos de uma mesma classe sugere a existência de um agrupamento maior, no qual ambos estão contidos.

Dois modelos para se calcular a probabilidade condicional de um ponto pertencer a uma classe[1]foram propostos. Sendo $d(\mathbf{x},\mathbf{p})$ a distância euclidiana de $\mathbf{x}$ a $\mathbf{p}$, dada pela Equação 4.1, e C_k o conjunto de centroides cujo vizinho mais próximo pertence à classe k. Define-se a probabilidade do ponto $\mathbf{x}$ pertencer à classe j como o inverso da distância de x ao ponto mais próximo da classe k, normalizado pelo somatório deste mesmo fator calculado em relação a x e todas as outras classes. Este modelo é calculado pela equação 4.2

[1]Embora estes modelos tenham sido desenvolvidos no escopo desta pesquisa, nada impede que eles sejam utilizados em outros contextos como uma forma de se calcular probabilidades em modelos baseados em distância.

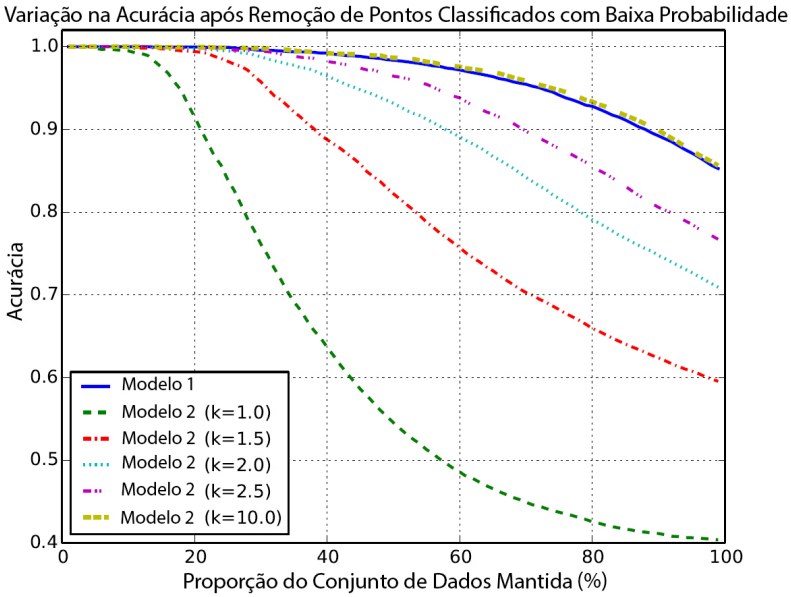

Figura 4.5: Comparação entre modelos Probabilísticos. Após a classificação, os pontos foram ordenados de acordo com a diferença entre as duas maiores probabilidades atribuídas. Com ambos os modelos pode-se obter próximo de 100% de acurácia ao se manter cerca de 40% dos pontos classificados com menores incertezas. O modelo 2 é pior que o modelo 1 para valores baixos de K. Para $K = 10$, ambos os modelos são quase equivalentes.

$$p(j \mid \mathbf{x}) = \frac{\left(\min_{\mathbf{p} \in C_j} d(\mathbf{x}, \mathbf{p})\right)^{-1}}{\sum_i \left(\min_{\mathbf{p} \in C_i} d(\mathbf{x}, \mathbf{p})\right)^{-1}} \tag{4.2}$$

que utiliza como evidência em favor do pertencimento a uma classe apenas a distância do ponto ao exemplar mais próximo desta classe. O segundo modelo utiliza como evidência a distância a todos os pontos. Neste modelo, a probabilidade é dada por

$$p(j \mid \mathbf{x}) = \frac{\sum_{\mathbf{p} \in C_j} d(\mathbf{x}, \mathbf{p})^{-K}}{\sum_i \sum_{\mathbf{p} \in C_i} d(\mathbf{x}, \mathbf{p})^{-K}} \tag{4.3}$$

onde K é um parâmetro que determina o quão rápido a influência de um exemplar deve decair com a distância. A Figura 4.5 mostra como a acurácia ("pureza") da classificação é afetada ao se remover os pontos classificados com maior incerteza (pontos cujas diferenças entre as duas maiores probabilidades atribuídas é pequena). Para valores grandes de K a influência do ponto mais próximo tende a dominar sobre a influência de pontos mais distantes e os resultados obtidos pelos dois modelos são quase idênticos. Para va-

lores pequenos de K o modelo 1 tem resultados superiores. Esta diferença ocorre porque as estruturas dos agrupamentos não são confinadas a regiões convexas do espaço. A informação de pontos distantes não é tão significativa quanto a informação de pontos mais próximos e, portanto, em geral considerar apenas os pontos mais próximos gera melhores resultados.

Outro uso possível para estes modelos consiste em determinar os pontos classificados com muita incerteza e removê-los na próxima etapa da abordagem proposta, pois a utilização de dados de treino com rótulos incorretos pode degradar a performance do classificador. Embora em alguns experimentos este procedimento pareça ter melhorado a acurácia final, em alguns casos o resultado obtido foi inferior. Isso mostra que a diversidade dos dados de treino é tão importante quanto a quantidade, pois ao se eliminar os pontos de menor certeza dos dados de treino, reduz-se a diversidade do conjunto de treino.

4.4 Treinamento de Modelo Supervisionado

Após a classificação inicial dos dados com a classe da amostra representativa de cada agrupamento, deve-se utilizar estes dados rotulados para treinar um classificador que possa ser utilizado em novos dados não vistos até então. Uma abordagem imediata seria utilizar somente as amostras representativas (cujas classes foram determinadas com precisão por um anotador externo) como conjunto de dados de treino de um modelo preditivo qualquer. Embora essa abordagem gere resultados melhores do que os obtidos ao se utilizar uma mesma quantidade de casos de treino escolhidos de forma aleatória, pode-se obter resultados ainda melhores ao se utilizar o conjunto total dos dados como conjunto de treino de um classificador. Um classificador “poderoso”, tal qual uma rede neural, é capaz de capturar os padrões essenciais que caracterizam cada categoria mesmo ao se utilizar um conjunto de dados “impuros” (com uma grande quantidade de rótulos errados).

Capítulo 5

Resultados

Alguns dos resultados intermediários obtidos ao longo deste trabalho foram descritos junto com a abordagem descrita no Capítulo 4. Para se ter uma ideia geral da eficácia da estratégia proposta, foram realizados experimentos de classificação com o *benchmark* MNIST e os resultados obtidos foram comparados com outras técnicas de aprendizado semi-supervisionado no estado da arte.

O modelo utilizado nos experimentos foi treinado com os seguintes passos (ilustrados na Figura 1.1):

1. Treinamento de SDAE com os dados de treino do MNIST e conversão dos dados de treino e teste para o espaço de características.
2. Execução do algoritmo k-médias nos dados de treino com valor de k igual a 100, 600 e 1.000 (números de exemplos rotulados utilizados em cada experimento), já no espaço de características (dados convertidos pelo SDAE).
3. Obtenção das classes dos exemplares mais próximos dos centroides em cada agrupamento (obviamente, como os rótulos verdadeiros das imagens estão disponíveis, não é preciso se recorrer a um "anotador" externo para se obter estas classes).
4. Classificação de cada exemplar do conjunto de treino com a classe do centroide do agrupamento ao qual o exemplar pertence. Após este passo o conjunto de treino já se encontra todo rotulado (embora nem todos os rótulos sejam corretos).
5. Treinamento de rede neural utilizando o conjunto de treino obtido no passo anterior (já rotulado) como conjunto de dados de treino.

Em alguns testes, os modelos probabilísticos descritos na Seção 4.2 foram utilizados antes do treinamento da rede neural com o intuito de "limpar" o conjunto de dados de treino, removendo o excesso de exemplares classificados com baixa probabilidade no

Figura 5.1: Exemplos de dígitos do MNIST (Fonte: [LeCun et al. 1998]).

passo 4. Este procedimento, contudo, não pareceu gerar resultados melhores. Aparentemente ao se remover os dados classificados com baixa probabilidade, a diversidade do conjunto de dados de treino é reduzida. Isto mostra que um bom conjunto de dados de treino não precisa ser somente grande e livre de ruídos, mas também precisa ter exemplares com alto nível de variabilidade. Experimentos sistemáticos seriam necessários para se chegar a conclusões mais sólidas. Nos resultados descritos a seguir esta etapa não foi realizada.

5.1 MNIST

O MNIST é um conjunto de imagens de dígitos em escala de cinza, de tamanho 28 por 28 pixels, desenvolvida por [LeCun et al. 1998]. O conjunto de dados foi obtido a partir da compilação de imagens de dois outros conjunto de imagens de dígitos, originalmente em preto e branco. As imagens foram redimensionadas para ocupar o espaço de uma caixa de 20 por 20 pixels. Como efeito da técnica de antisserrilhamento utilizada, as imagens

Tabela 5.1: Comparação de Taxa de Erro no MNIST (em %)

Método	100	600	1000
Supervised Neural Network	25.13	11.21	9.84
Manifold Tangent Classifier [Rifai & Dauphin 2011]	12.6	5.13	3.64
Semi-Supervised Embedding [Weston et al. 2012]	**7.75**	3.82	**2.73**
Pseudo-label [Lee 2013]	10.49	4.01	3.46
AL + MLP	8.17	4.57	3.86
AL + ConvNet	7.98	**3.73**	3.32

resultantes contém valores em escala de cinza.

O conjunto de dados é dividido e duas porções: dados de treino, composto por 60 mil exemplares, e conjunto de teste, composto por 10 mil exemplares. É comum que 10 mil imagens do conjunto de treino sejam utilizadas como conjunto de "validação", utilizado na otimização de hiper-parâmetros.

O MNIST é provavelmente o *benchmark* mais utilizado atualmente na avaliação de técnicas de reconhecimento de imagens e de AM. Alguns dígitos são ilustrados na Figura 5.1.

5.2 Desempenho da Abordagem

Para comparar o desempenho da abordagem proposta com outras de aprendizado semi-supervisionado, foram realizados experimentos com o conjunto de dados MNIST, reportando-se os resultados da classificação realizada com o conjunto de testes padrão de 10.000 dígitos. Duas arquiteturas de redes neurais foram treinadas: *Multilayer Perceptron* (MLP) e *Convolutional Neural Network* (ConvNet) como última etapa de estratégia de aprendizado ativo. Na Tabela 5.1 é comparada a taxa de erro da abordagem proposta com outros modelos de aprendizado semi-supervisionado ao se utilizar 100, 600 e 1.000 exemplares rotulados (e o restante do conjunto de treino como dados não rotulados). Para representar o desempenho esperado em um contexto realista, a etapa do algoritmo de agrupamento foi executada 30 vezes (com inicializações aleatórias) e foi selecionado o modelo com a acurácia mediana para prover os rótulos utilizados no treinamento das redes neurais. Optou-se por não se utilizar dados adicionais para otimização de hiper-parâmetros e, ao invés disso, utilizou-se como conjunto de dados de validação 10.000 amostras extraídas aleatoriamente dos dados de treino classificados na etapa anterior com o modelo baseado em distância utilizando os centróides dos agrupamentos como dados

de treino.

A abordagem proposta obteve a taxa de erro mais baixa reportada ao se utilizar 600 exemplos de treino. Para os outros valores, os resultados obtidos foram somente um pouco inferiores aos alcançados pela técnica *Semi-Supervised Embedding*, que é brevemente descrita no no Capítulo 3 (junto com as demais técnicas comparadas).

5.3 Resultados Intermediários

Embora o objetivo final deste trabalho tenha sido criar uma abordagem de aprendizado ativo, foi necessária a realização de experimentos para comprovar a eficácia dos passos executados, que foram concebidos baseados principalmente em intuições geométricas e deduções a partir de premissas relativamente pouco sólidas. Estes resultados em grande parte podem ser também aplicados por si sós em contextos diversos, portanto, são listados aqui como contribuições adicionais da pesquisa.

5.3.1 Melhores agrupamentos utilizando SDAE

Através dos experimentos resumidos na Figura 4.2, demonstrou-se que é possível se achar agrupamentos mais homogêneos através da extração prévia de características de forma não supervisionada com SDAE. O mesmo não se mostrou possível com PCA, que não é capaz de extrair características de maior complexidade.

É provável, contudo, que o SDAE seja útil somente em contextos nos quais os dados possam ser representados como hierarquias de características de complexidade crescente, como por exemplo no reconhecimento de imagem e fala. Em problemas nos quais haja menos estrutura nos dados, é possível que técnicas lineares, como PCA, possam gerar resultados igualmente bons, ou até melhores, por não serem menos dependentes do ajuste de hiperparâmetros.

5.3.2 Amostra representativa da categoria mais prevalente no agrupamento

Demonstrou-se experimentalmente que a classe mais prevalente em cada agrupamento encontrado pelo algoritmo k-médias é quase sempre a mesma classe do ponto mais próximo do centroide do respectivo agrupamento. Este resultado foi ilustrado na Figura 4.3. Esta aproximação é um resultado importante que viabiliza esta abordagem. Caso fosse necessário classificar vários pontos dentro de cada agrupamento para se determinar a classe

mais prevalente, a abordagem proposta se tornaria rapidamente inviável, pois seria necessário um número elevado de rótulos, contrariando o propósito de redução da dependência de dados rotulados.

5.3.3 Influência da inicialização do k-médias

Há uma correlação negativa modesta ($r = -0.26$, $p < 0.001$) entre a acurácia medida após a execução do k-médias e a função de distorção dada pela Equação 2.17. A utilização do método de inicialização k-means++ também é relevante, resultando em acurácias mais altas, como medido por um teste T de amostras independentes ($t(58) = 3.66$, $p < 0.001$). Embora esses resultados fossem de certa forma previsíveis, optou-se por listá-los aqui, pois servem como evidência experimental para reforçar a relevância do procedimento de inicialização para obtenção de melhores resultados de agrupamentos.

5.3.4 Robustez a ruídos nos rótulos do conjunto de treino de Redes Neurais

Após a realização do agrupamento e classificação de todos os exemplares não rotulados com os rótulos das amostras representativas de seus agrupamentos obtém-se um conjunto de dados de treino com erros (cerca de 12% ao se utilizar 100 agrupamentos). Mesmo se utilizando estes dados "impuros" como conjunto de treino de uma rede neural, conseguiu-se obter resultados com taxas de erro abaixo de 8% no conjunto de teste do MNIST, demonstrando empiricamente que os modelos de redes neurais utilizados têm certa robustez a presença de ruídos nos rótulos do conjunto de treino.

5.3.5 Modelos probabilísticos para classificadores baseados em distância

Propomos dois modelos probabilísticos sem equivalentes na literatura da área (até onde foi possível verificar) que podem ser utilizados para determinação da probabilidade de classificação em modelos baseados em distância. Embora estes modelos acabaram tendo um papel secundário na estratégia de aprendizado ativo proposta (pois a remoção dos pontos classificados com baixa probabilidade não resultou aparentemente em melhorias na acurácia final), eles podem ser utilizados de forma independente sempre que se utilizar um modelo baseado em distância. A versão descrita para o KNN quando k é igual a 1 pode ser facilmente estendida para outros valores de k.

Capítulo 6

Conclusões e Trabalhos Futuros

Os avanços recentemente obtidos por redes neurais artificiais na execução de diversas tarefas (até pouco tempo consideradas de domínio exclusivo de "cérebros") têm reforçado as evidências de que a percepção de conceitos complexos é realizada através da perceção de conceitos mais simples de forma hierárquica.

Inspirado nestes avanços, este trabalho investigou a viabilidade de se utilizar técnicas de aprendizado não supervisionado de características como mecanismo hábil à geração de representações de mais alto nível mais adequadas à execução de algoritmos de agrupamento, e, consequentemente, se este artifício pode ser explorado na elaboração de uma nova estratégia de aprendizado ativo, na qual todos os pontos pertencentes a um mesmo agrupamento são classificados com a classe atribuída por um anotador externo a uma única amostra do agrupamento.

Os experimentos realizados no *benchmark* MNIST demonstraram que as premissas assumidas estavam corretas:

- A conversão dos dados brutos do MNIST para um espaço de características utilizando SDAE resultou em agrupamentos mais puros encontrados pelo k-médias (algo que não foi possível com PCA).
- A classe mais prevalente em cada agrupamento encontrado pelo k-médias pode ser aproximada com erro quase nulo pela classe do elemento mais próximo do centroide do respectivo agrupamento.
- As redes neurais treinadas com o conjunto de dados de treino do MNIST utilizando os rótulos obtidos após a execução do k-médias obteve acurácia de classificação nos dados de teste similar (ou até superiores) às obtidas por técnicas de aprendizado semi-supervisionado no estado da arte. Como estes dados de treino continham muitos rótulos errados, este resultado era longe de óbvio.

A pesquisa ainda se somou às evidências empíricas já existentes de que melhores resultados no k-médias podem ser alcançados com a utilização do método k-means++ como técnica de inicialização dos centroides e com a realização de múltiplas execuções do algoritmo com escolha da execução que obteve menor valor da função de distorção.

Por último, durante a pesquisa foram elaborados dois modelos probabilísticos para estimação da probabilidade de classificação em modelos baseados em distância. Embora estes modelos não tenham desempenhado um papel importante na proposta de aprendizado ativo final, eles podem ser utilizados em outros contextos, como na obtenção de grandes quantidades de instâncias de uma categoria específica a partir de uma grande quantidade de dados rotulados.

Os resultados obtidos geram vários questionamentos para orientar futuras investigações. A seguir são descritos quatro desdobramentos que se acredita merecerem atenção futura.

6.1 Aprendizado Autodidata

Um problema das abordagens de aprendizado semi-supervisionado é que elas requerem a disponibilidade de uma grande quantidade de dados não rotulados originados da mesma distribuição generativa dos dados rotulados. Por exemplo, caso deseje-se criar um classificador capaz de distinguir rinocerontes de elefantes, para se aplicar as técnicas tradicionais de aprendizado semi-supervisionado, é preciso que se tenha acesso a uma grande quantidade de imagens não rotuladas destes animais. É difícil de se imaginar um processo de coleta destas imagens sem que seja necessário realizar a própria classificação. Para resolver este problema Raina et al. [2007] propuseram um novo *framework* de aprendizado semi-supervisionado, o *Self-Taught Learning*, no qual se utiliza dados oriundos de uma distribuição generativa diferente da dos dados para os quais se deseja criar um classificador. Essa abordagem se baseia na observação de que mesmo imagens baixadas aleatoriamente da internet conterão padrões (como contornos, extremidades, texturas, partes de objetos, etc.) que sejam similares aos padrões encontrados nas imagens de elefantes e rinocerontes.

Acredita-se que este paradigma se encaixe bem com a abordagem proposta neste trabalho. Antes de aprenderem a reconhecer dígitos (como os do MNIST, objeto dos experimentos realizados), seres humanos interagem com o mundo e aprendem a extrair características úteis para a representação de uma vasta quantidade de objetos. Em especial, no sistema visual, a região V1 do córtex aprende a reconhecer segmentos de retas orientados, que poderiam ser úteis como características para composição de dígitos. O aprendizado

de padrões utilizando somente imagens de dígitos não é capaz de gerar padrões como estes e portanto reconhecer características tão básicas como estas.

6.2 Métricas e Algoritmos de Agrupamento Alternativos

O algoritmo k-médias utilizado nos experimentos está intimamente ligado à distância euclidiana (equação 4.1), que é reescrita aqui em uma forma alternativa

$$d(\mathbf{x},\mathbf{p}) = \sqrt{\sum_{i=1}^{N} (\mathbf{x_i^2} - 2\mathbf{x_i}\mathbf{p_i} + \mathbf{p_i^2})} \tag{6.1}$$

Há um problema fundamental em usar a distância euclidiana como métrica para comparação entre dois objetos. Consideremos por exemplo o comportamento da métrica ao se comparar dois objetos cujas características são dadas por funções indicadoras (i.e. que possuem o valor zero para a ausência de uma característica e o valor um para indicar sua presença). Para cada característica a equação adiciona o termo $(\mathbf{x_i^2} - 2\mathbf{x_i}\mathbf{p_i} + \mathbf{p_i^2})$ à somatória dentro da raiz. A tabela abaixo resume os valores que esse termo pode assumir de acordo com a presença ou ausência das características (onde 0 indica a ausência da característica e 1 a sua presença).

Tabela 6.1: Influência de cada característica na Distância Euclidiana

$\mathbf{x_i}$	$\mathbf{p_i}$	$(\mathbf{x_i^2} - 2\mathbf{x_i}\mathbf{p_i} + \mathbf{p_i^2})$
0	0	0
0	1	1
1	0	1
1	1	0

No primeiro caso, quando ambos os objetos não possuem uma determinada característica (e os valores são, portanto, iguais a 0), não se pode dizer que eles são mais ou menos diferentes. Por exemplo, ao se comprar uma imagem qualquer com a imagem de um elefante. Supondo que a imagem qualquer não possui a característica “asa” (tampouco a do elefante), a ausência desta característica em ambas as imagens não nos permite falar nada sobre elas. Nos dois casos seguintes, a distância entre os dois objetos é aumentada em uma unidade toda vez que um objeto possui uma característica e o outro não, como é de se esperar. Por exemplo, a imagem do elefante possui a característica “tromba”; caso a

outra imagem não possua esta característica, é plausível que a distância entre as imagens deva ser considerada maior. No último caso, quando ambos os objetos possuem uma característica em comum, a distância entre eles também não aumenta. No entatno, ela não deveria diminuir? O conhecimento de que duas imagens compartilham a característica "tromba" aumenta bastante a probabilidade de que elas representem o mesmo objeto (um elefante). A distância euclidiana simplesmente ignora este fato, tratando características coincidentes da mesma forma que a ausência de características em ambos objetos. Na tabela a seguir são apresentados os valores que possivelmente representam melhor a noção intuitiva de distância entre dois objetos.

Tabela 6.2: Influência de cada característica na Métrica Proposta

$\mathbf{x_i}$	$\mathbf{y_i}$	Métrica Proposta
0	0	0
0	1	1
1	0	1
1	1	-1

Nesta tabela, diferentemente da anterior, quando dois objetos tem uma característica em comum, sua distância decresce. A equação que nos permite obter estes valores é a seguinte

$$d(\mathbf{x},\mathbf{p}) = \sqrt{\sum_{i=1}^{N} (\mathbf{x_i^2} - 3\mathbf{x_i}\mathbf{p_i} + \mathbf{p_i^2})} \tag{6.2}$$

A diferença desta equação para a distância euclidiana é unicamente o fator 2, que passa a ser um 3, e consequentemente resulta em um decréscimo na distância quando dois fatores são comuns a ambos os objetos. Um problema "grave" com esta equação é que ela não respeita os axiomas dos espaços métricos, o que impossibilita seu uso em diversos contextos. Acredita-se, contudo, que seja possível utilizá-la como função de distância em alguns algoritmos de agrupamento, como o k-medoides por exemplo, que pode ser visto como uma generalização do k-médias, no qual há maior flexibilidade na escolha da métrica.

Alguns experimentos foram realizados utilizando o algoritmo k-médias esférico [Dhillon 2004]), que é baseado na distância cosseno, contudo os testes preliminares não demonstraram melhorias em comparação ao k-médias. Aparentemente a distância cosseno sofre dos mesmos problemas atribuídos à distância euclidiana, portanto seu uso não nos parece

promissor.

6.3 Compressão de Modelo e "Dark Knowledge"

Bucilua et al. (2006) criaram um método para transferir o conhecimento armazenado em *ensembles* compostos por muitos classificadores para modelos mais simples baseados em redes neurais. Recentemente este método inspirou novos trabalhos focados na transferência do conhecimento armazenado em modelos complexos para modelos mais simples [Ba & Caruana 2013, Hinton et al. 2014]. Uma observação importante feita por estes trabalhos é que grande parte da informação armazenada em um classificador está armazenada nas probabilidades atribuídas a classes erradas. Por exemplo, um classificador ao classificar a imagem de uma BMW pode atribuir uma probabilidade muito pequena a categoria "Caminhão de Lixo", contudo esta probabilidade pequena é ainda assim muito superior à probabilidade atribuída à categoria "cenoura". Esse conhecimento contido nestas pequenas probabilidades foi apelidado por Hinton de "*dark knowledge*".

Na abordagem proposta neste trabalho também efetuou-se uma transferência de conhecimento de um modelo para outro, contudo o primeiro modelo foi utilizado na classificação do conjunto de dados não rotulado para uso no treinamento de uma rede neural. A informação das probabilidades atribuídas às classes erradas é completamente ignorada. Esta informação poderia ser explorada com uma das técnicas propostas nos trabalhos mencionados, resultando em um modelo com maior acurácia.

Referências Bibliográficas

Arthur, David & Sergei Vassilvitskii (2007), k-means ++ : The Advantages of Careful Seeding, *em* 'Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms', Vol. 8, pp. 1027–1035.

Ba, Lei & Rich Caruana (2013), 'Do Deep Nets Really Need to be Deep ?', *arXiv preprint arXiv:1312.6184* **2014**, 1–6.

Bengio, Yoshua (2009), 'Learning Deep Architectures for AI', *Foundations and Trends® in Machine Learning* **2**(1), 1–127.

Bengio, Yoshua, Aaron Courville & Pascal Vincent (2013), 'Representation learning: A review and new perspectives', *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **35**(8), 1798–1828.

Bengio, Yoshua, Ian J. Goodfellow & Aaron Courville (2014), *Deep Learning*. Book in preparation for MIT Press.

Bengio, Yoshua, Pascal Lamblin, Dan Popovici & Hugo Larochelle (2007), 'Greedy layer-wise training of deep networks', *Advances in neural information processing systems* **19**(1), 153–160.

Bishop, Christopher M (2007), *Pattern Recognition and Machine Learning*, Springer, New York.

Chapelle, Olivier, Bernhard Schölkopf & Alexander Zien (2006), *Semi-Supervised Learning*, Vol. 1.

Ciresan, Dan, Ueli Meier, Jonathan Masci & Jürgen Schmidhuber (2012), 'Multi-column deep neural network for traffic sign classification', *Neural Networks* **32**, 333–338.

Dasgupta, Sanjoy (2011), 'Two faces of active learning', *Theoretical computer science* **412**(19), 1767–1781.

Dasgupta, Sanjoy & Daniel Hsu (2008), Hierarchical sampling for active learning, *em* 'Proceedings of the 25th international conference on Machine learning', pp. 208–215.

Deng, Li & Dong Yu (2013), 'Deep Learning: Methods and Applications', *Foundations and Trends in Signal Processing* **7**, 197–387.

Dhillon, Inderjit S (2004), 'Concept Decompositions for Large Sparse Text Data using Clustering', *Machine Learning* **42**(1-2), 143–175.

Erhan, Dumitru, Aaron Courville & Pascal Vincent (2010), 'Why Does Unsupervised Pre-training Help Deep Learning ?', *Journal of Machine Learning Research* **11**, 625–660.

Fralick, S. (1967), 'Learning to recognize patterns without a teacher', *IEEE Transactions on Information Theory* **13**.

Goodfellow, Ian J., David Warde-Farley, Pascal Lamblin, Vincent Dumoulin, Mehdi Mirza, Razvan Pascanu, James Bergstra, Frédéric Bastien & Yoshua Bengio (2013), 'Pylearn2: a machine learning research library', *arXiv preprint arXiv:1308.4214* .

Hastad, Johan (1986), 'Almost Optimal Lower Bounds for Small Depth Circuits', *STOC '86 Proceedings of the eighteenth annual ACM symposium on Theory of computing* pp. 6–20.

Hastie, Trevor, Robert Tibshirani & Jerome Friedman (2009), *The Elements of Statistical Learning*, Vol. 18.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren & Jian Sun (2015), 'Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification', *arXiv preprint arXiv: 1502.01852* .

Hebb, D. O. (1949), *The Organization of Behaviour*, John Wiley & Sons, New York.

Hinton, Geoffrey, Oriol Vinyals & Jeffrey Dean (2014), Distilling the Knowledge in a Neural Network, *em* 'NIPS 2014 Deep Learning Workshop', pp. 1–9.

Hinton, Geoffrey & RR Salakhutdinov (2006), 'Reducing the dimensionality of data with neural networks', *Science* **313**(July), 504–507.

Hinton, Geoffrey, Simon Osindero & Yee-Whye Teh (2006), 'A fast learning algorithm for deep belief nets.', *Neural computation* **18**(7), 1527–1554.

Hochreiter, S (1991), 'Untersuchungen zu dynamischen neuronalen Netzen', *Master's thesis, Institut fur Informatik, Technische Universitat, Munchen* .

Hubel, DH & TN Wiesel (1959), 'Receptive fields of single neurones in the cat's striate cortex', *The Journal of physiology* pp. 574–591.

Ito, Minami & Hidehiko Komatsu (2004), 'Representation of angles embedded within contour stimuli in area V2 of macaque monkeys.', *The Journal of neuroscience : the official journal of the Society for Neuroscience* **24**(13), 3313–24.

Krizhevsky, Alex, Ilya Sutskever & Geoffrey E Hinton (2012), 'ImageNet Classification with Deep Convolutional Neural Networks', *Advances In Neural Information Processing Systems* pp. 1–9.

Kunihiko Fukushima (1980), 'Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position', *Biological Cybernetic* **36**, 193–202.

Le, QV (2013), 'Building high-level features using large scale unsupervised learning', *Acoustics, Speech and Signal Processing (ICASSP)* .

LeCun, Yann (1985), Une procédure d'apprentissage pour réseau a seuil asymmetrique (a Learning Scheme for Asymmetric Threshold Networks), *em* 'Cognitiva 85', pp. 599–604.

LeCun, Yann (2015), 'Facebook AI Director Yann LeCun on His Quest to Unleash Deep Learning and Make Machines Smarter'.

LeCun, Yann, Léon Bottou, Yoshua Bengio & Patrick Haffner (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**, 2278–2323.

Lee, Dong-Hyun (2013), Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks, *em* 'Workshop on Challenges in Representation Learning, ICML'.

Lee, Honglak, Chaitanya Ekanadham & Andrew Y Ng (2008), Sparse deep belief net model for visual area V2, *em* 'Advances in Neural Information Processing Systems 20', pp. 873–880.

Lefakis, Leonidas & MA Wiering (2007), 'Semi-Supervised Methods for Handwritten Character Recognition using Active Learning', *Proceedings of the Belgium-Netherlands Conference on Artificial Intelligence* pp. 205–212.

Lewis, D.D. & Jason Catlett (1994), 'Heterogeneous uncertainty sampling for supervised learning', *Proceedings of the eleventh international conference on machine learning* pp. 148–156.

Loosli, Gaëlle, Stéphane Canu & Léon Bottou (2007), Training Invariant Support Vector Machines using Selective Sampling, *em* L.Bottou, O.Chapelle, D.DeCoste & J.Weston, eds., 'Large Scale Kernel Machines', MIT Press, Cambridge, MA, pp. 301–320.

Martens, James (2010), Deep learning via Hessian-free optimization, *em* 'Proceedings of the 27th International Conference on Machine Learning (ICML-10)', pp. 735–742.

McCulloch, Warren S. & Walter Pitts (1943), 'A logical calculus of the ideas immanent in nervous activity', *The Bulletin of Mathematical Biophysics* **5**, 115–133.

Mendelson, Elliott (1997), 'Introduction to Mathematical Logic', pp. 1–440.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra & Martin Riedmiller (2013), 'Playing Atari with Deep Reinforcement Learning', *arXiv preprint arXiv: 1312.5602* .

Newborn, Monty (1996), *Kasparov Versus Deep Blue: Computer Chess Comes of Age*, Springer-Verlag New York, Inc., New York, NY, USA.

Newell, Allen & Herbert A.Simon (1956), 'The Logic Theory Machine: A Complex Information Processing System'.

Nguyen, Hieu T & Arnold Smeulders (2004), Active learning using pre-clustering, *em* 'Proceedings of the twenty-first international conference on Machine learning', ACM, p. 79.

Olshausen, B A & D J Field (1996), 'Emergence of simple-cell receptive field properties by learning a sparse code for natural images.', *Nature* **381**, 607–609.

Ostrovsky, Rafail, Yuval Rabani, Leonard Schulman & Chaitanya Swamy (2006), The Effectiveness of Lloyd-type Methods for the k-Means Problem, *em* 'Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on', pp. 165–176.

Paine, Tom Le, Pooya Khorrami, Wei Han & Thomas S. Huang (2015), An Analysis of Unsupervised Pre-training in Light of Recent Advances, *em* 'Workshop at ICLR2015', pp. 1–10.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg et al. (2011), 'Scikit-learn: Machine learning in python', *The Journal of Machine Learning Research* **12**, 2825–2830.

Raina, Rajat, A Battle, H Lee, B Packer & AY Ng (2007), Self-taught learning: transfer learning from unlabeled data, *em* 'Proceedings of the 24th international conference on Machine learning', pp. 759–766.

Rajaraman, Anand & Jeffrey D Ullman (2014), *Mining of Massive Datasets*, Cambridge University Press.

Ranzato, Marc, Y-Lan Boureau & Yan LeCun (2007), 'Sparse feature learning for deep belief networks', *NIPS* pp. 1–8.

Rifai, Salah & YN Dauphin (2011), 'The manifold tangent classifier', *Advances in Neural Information Processing Systems* pp. 2294–2302.

Rosenblatt, Frank. (1962), *Principles of neurodynamics; perceptrons and the theory of brain mechanisms*, Washington.

Rumelhart, David E., Geoffrey E. Hinton & Ronald J. Williams (1986), 'Learning representations by back-propagating errors', *Nature* **323**, 533–536.

Russell, Stuart & Peter Norvig (2009), *Artificial Intelligence: A Modern Approach, 3rd edition*.

Scheffer, Tobias, Christian Decomain & Stefan Wrobel (2001), 'Active Hidden Markov Models for Information Extraction', *IDA '01: Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis* **01**, 309–318.

Scudder, H., Iii (1965), 'Probability of error of some adaptive pattern-recognition machines', *IEEE Transactions on Information Theory* **11**.

Selfridge, O. G. (1959), Pandemonium: A paradigm for learning, *em* 'Proceedings of the Symposium on Mechanisation of Thought Processes', Vol. 1, HMSO, pp. 511–529.

Settles, Burr (2010), 'Active learning literature survey', *University of Wisconsin, Madison* .

Tom Mitchel (1997), *Machine Learning*, McGraw-Hill, Boston, MA.

Tong, Simon & Daphne Koller (2001), 'Support Vector machine Active Learning with Applications to Text Classification', *Journal of Machine Learning Research* pp. 45–66.

Vapnik, Vladimir N. & A. Ja Chervonenkis (1974), 'Theory of pattern recognition [in Russian]'.

Vapnik, Vladimir Naumovich (1998), *Statistical Learning Theory*, New York.

Vincent, Pascal, Hugo Larochelle, Yoshua Bengio & Pierre-Antoine Manzagol (2008), Extracting and composing robust features with denoising autoencoders, *em* 'Proceedings of the 25th international conference on Machine learning', ACM, pp. 1096–1103.

Wegener, Ingo (1987), 'Complexity of Boolean Functions WARNING', *Complexity* **7**(1), 1–22.

Werbos, Paul (1974), Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, Tese de doutorado, Harvard University.

Weston, Jason, Christina Leslie, Eugene Ie, Dengyong Zhou, Andre Elisseeff & William Stafford Noble (2005), 'Semi-supervised protein classification using cluster kernels.', *Bioinformatics (Oxford, England)* **21**(15), 3241–7.

Weston, Jason, Frédéric Ratle, Hossein Mobahi & Ronan Collobert (2012), Deep learning via semi-supervised embedding, *em* 'Neural Networks: Tricks of the Trade', Springer, pp. 639–655.

Wiesel, T N (1968), 'Receptive Fields and Functional Architecture of Monkey Striate Cortex', *J. Physiol.* **195**, 215–243.

Xu, Zhao, Kai Yu, Volker Tresp, Xiao Wei Xu & Jizhi Wang (2003), *Representative Sampling for Text Classification Using Support Vector Machines*, Springer Berlin Heidelberg.

Zhu, Xiaojin, Zoubin Ghahramani & John Lafferty (2003), 'Semi-supervised learning using gaussian fields and harmonic functions', *ICML* **3**, 912–919.